Final Report

Skin Cancer Detection

DESIGN DOCUMENT

07
Dr. Gaffar Ashraf
Evan Hanson, Mishar Alhyarbi, Ziyad Alqahtani, Wonjun Choi
Anirudh Ambore, Abdelrahman Mohamed
sdmay24-07@iastate.edu
https://sdmay24-07.sd.ece.iastate.edu/

4/27/2024

1 Introduction

1.1 Problem Statement

The project aims to address the problem of improving skin cancer detection using artificial intelligence (AI) and cloud computing. The primary problem is the need for more accurate and efficient methods to detect skin cancer. Currently, skin cancer diagnosis relies heavily on manual examination by medical professionals, which can be time-consuming and subject to human error. By leveraging AI, the project intends to enhance the accuracy, convenience, and speed of skin cancer detection, ultimately benefiting both patients and healthcare providers.

1.2 Intended Users

Our project targets individuals and organizations involved in skin cancer detection and diagnosis. Given the complexity of the system, we have identified specific users and their needs to tailor our approach effectively. The intended users include:

Medical Professionals:

Needs: Medical professionals require reliable tools for accurate skin cancer diagnosis to enhance patient outcomes.

Objectives: To reduce misdiagnosis rates and improve the efficiency of skin cancer detection processes.

Requirements: The AI model should provide precise assessments of skin lesions and integrate seamlessly into existing healthcare workflows.

Patients:

Needs: Patients seek timely and accurate identification of potential cancerous moles and spots to ensure early diagnosis and treatment.

Objectives: To empower individuals to proactively monitor their skin health and seek medical assistance promptly when needed.

Requirements: The user interface should be intuitive, allowing for easy uploading of skin images and providing clear and understandable diagnostic results.

By focusing on these specific user groups, we aim to develop tailored requirements and use cases that address their unique needs. Moreover, our project serves as a platform for training and skill development:

Medical Institutes:

Needs: Some medical institutions are interested in implementing AI-based solutions for skin cancer detection to enhance their diagnostic capabilities.

Objectives: To leverage the experiences and insights gained from our project to develop and deploy similar systems in their own institutions.

Requirements: The project documentation should provide clear guidelines and best practices for implementing AI-based skin cancer detection systems in diverse medical settings.

Students:

Needs: Students seek opportunities to gain practical experience in AI and cloud computing technologies to prepare for future careers.

Objectives: To acquire hands-on experience in developing and deploying AI models for real-world applications.

Requirements: The project should offer opportunities for students to actively participate in AI model development, deployment, and real-world application, providing them with valuable experiential learning opportunities.

1.3 Contextual Placement within Related Products and Literature:

Our project builds upon existing research and technologies in the field of medical imaging and artificial intelligence:

Literature Review: We conducted a review of relevant literature on skin cancer detection methods, AI applications in healthcare, and cloud-based diagnostic tools. By synthesizing prior research findings, we identified opportunities for innovation in the domain.

Some of the literature and website we reviewed were:
Applied Sciences | Free Full-Text | Skin Cancer Disease Detection Using Transfer Learning Technique (mdpi.com)
Optimal Skin Cancer Detection Model Using Transfer Learning and Dynamic-Opposite Hunger Games Search - PMC (nih.gov)

Related Products: Our project aligns with and extends upon existing AI-based diagnostic tools, for various medical conditions, showcasing advancements in machine learning algorithms tailored specifically for skin cancer detection. An example of an application we referred to is:
AI dermatologist: Skin scanner (ai-derm.com)

Industry Standards and Best Practices: We adhere to industry standards and best practices in medical imaging, data privacy, and AI model development, ensuring the reliability and ethical integrity of our solution. We use the datasets provided by ISIC as they are widely used in the industry to train models used for skin cancer detection applications:
ISIC Challenge (isic-archive.com)

2 Reviewed Design

2.1 Requirements (functional & non-functional)

Functional Requirements

1. **Image Upload and Processing:**

Requirement: Users should be able to upload images of skin lesions for analysis.

Satisfaction: The system allows users to upload images through a user-friendly interface. The AI model then processes these images, ensuring this primary functionality is met.

2. **Skin Cancer Diagnosis:**

Requirement: The system should provide an accurate assessment of whether the uploaded image is indicative of skin cancer.

Satisfaction: The AI model, trained on a comprehensive dataset of skin cancer images, assesses the uploaded images. While the initial design ensures this functionality, continuous improvement in model training will enhance accuracy over time.

3. **Utilize AI/ML models trained over skin cancer datasets:**

Requirement: The system should use AI/ML models we ourselves trained on a wide range of skin cancer examples.

Satisfaction: We ensure this by employing AI models trained on accurate datasets that cover various types and stages of skin cancer.

Non-Functional Requirements

1. **Scalability:**

Requirement: The system should be able to handle a growing number of users without performance degradation.

Satisfaction: By doing cloud deployment, the system is inherently scalable, ensuring that it can accommodate an increasing user base while maintaining performance.

2. **Accessibility:**

Requirement: The system should be accessible from anywhere, without geographical constraints.

Satisfaction: The cloud-based approach ensures that users can access the system from anywhere, satisfying this requirement.

3. **Usability:**

Requirement: The system should be easy to use, catering to both healthcare professionals and patients.

Satisfaction: The Flask web app user interface leveraging the Bootstrap library is intuitive, simplifying the process of uploading images and viewing results, which enhances the system's usability.

4. **Security:**

Requirement: User data like medical images should not be saved, and privacy should be maintained.

Satisfaction: To incorporate security measures, the application destroys images after using them to process diagnostic results, complying with legal and ethical requirements like HIPAA.

5. **Performance:**

Requirement: The system should provide quick response times, especially when generating skin cancer diagnoses.

Satisfaction: The cloud architecture is designed for robust performance, and the AI model's efficiency ensures quick diagnoses within a few seconds, meeting the performance requirements.

6. **Reliability:**

Requirement: The system should provide accurate and reliable diagnoses.

Satisfaction: The AI model's accuracy is central to the system's reliability. Transfer learning and model training over 11,000+ images from medical institutes aims to provide and maintain a level of reliability that, while not perfect, is still reasonably accurate.

2.2 Engineering Standards

Software Development Practices: Agile and scrum methodologies for project management and development.

Data Privacy Regulations: Compliance with data privacy laws, like HIPAA, when handling medical data.

Cloud Computing Standards: Following best practices for cloud-based infrastructure and data security.

Machine Learning and AI Ethics: Adhering to ethical guidelines and standards for AI model development and deployment.

Educational Standards: Ensuring that the educational program meets relevant teaching and curriculum standards.

2.3 Security concerns and countermeasures

Concern: A major security concern in this project would be the saving of images, which constitutes sensitive information, uploaded by the users for skin cancer prediction. While we need to process these images, storing them can pose risks related to data privacy, unauthorized access, and potential breaches.

Countermeasure: To avoid this, we carry out the following:

In-memory processing - We use file.stream, which provides a temporary, in-memory representation of the uploaded image during the request/response cycle. It allows us to process the image directly without saving it. By avoiding storage, we minimize the risk of data exposure and unauthorized access.

No User Login Information Required - To further protect privacy, users are not required to enter login information as the application aims to simply process images and predict the likelihood of cancer, and not store the images and results. HIPAA stresses on data minimization (i.e. collecting only necessary data).

2.4 Description of how your design has evolved since 491

AI/ML:

Our previous AI/ML model was quite simple. It followed the structure of a straightforward Tensorflow/Keras example for image classification. It relied on DenseNet201, a convolutional neural network requiring substantial computing resources and a large amount of labeled data. Since our dataset is not extremely large or diverse, it led to overfitting, meaning it fitted too closely to the training data and resulted in a model that could not make accurate predictions on unseen data.

For our current design, we utilize MobileNetV2. It is lightweight, allowing for efficient deployment. It also achieves competitive accuracy compared to larger, more computationally expensive models. The small size of the model speeds up inference times, making it suitable for real-time applications. We also employ transfer learning and fine-tuning, allowing us to capitalize on features learned by a pre-trained model on a large dataset. By starting with these pre-existing features, we jumpstart our model's ability to recognize patterns relevant to our specific task.

Training the model on the cloud:

One of the requirements of the project is to train the model on the Cloud. In 491, we trained our model locally and on Kaggle notebook, which leverages cloud-based resources to train deep learning models.

In our current approach, we wrote code and trained our model on both AWS and GCP. Gathering the metrics from each cloud platform, we have carried out benchmarking to compare their performance.

User Interface:

Our initial plan was to use Flutter to create a simple, user-friendly interface for our application. However, we have switched to Flask due to its compatibility with cloud technologies and machine learning.

3. Implementation details

3.1 Detailed design

AI/ML Model:

Data Preprocessing:

- The load_dataset function loads image data from a directory of skin cancer lesion images,
- It then splits them into training and validation sets with a specified size and batch size.
- It also performs some preprocessing steps like prefetching and splitting the validation dataset further into a test set to test the accuracy of the final model.
- Finally, it returns the processed datasets along with class names.

Model Creation and Training:

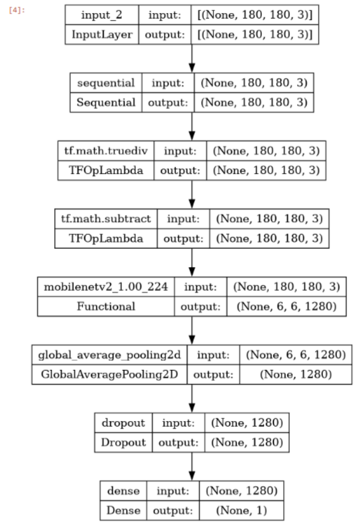
- This function creates a convolutional neural network (CNN) model using transfer learning with MobileNetV2 as the base model.
- It freezes the layers of the base model (except the last few) to prevent them from being updated during training, allowing the model to leverage pre-learned features while focusing on learning task-specific patterns and preventing overfitting.
- It compiles and trains the model on the training dataset for a specified number of epochs and evaluates its performance on the validation set.
- After initial training, it fine-tunes the model by unfreezing some layers of the base model and continuing training with a lower learning rate.
- The function returns the trained model.

Model Evaluation:

- This function evaluates the trained model on a test dataset.
- The function prints test accuracy and displays predicted labels along with actual labels.

Model Deployment:

- The preprocess_img function prepares an image for prediction by resizing it to the required input size and converting it to a format suitable for the model.
- The predict_result function takes a preprocessed image as input, passes it through the trained model and generates a human-readable prediction.



Our trained AI/ML model summary

Cloud Training:

For benchmarking, we deployed our model training code on both AWS and GCP.

In AWS, we trained the model on an Amazon EC2 instance of type t2.medium with 2 vCPUs and 4.0 GB RAM.

In GCP, we trained the model on a Google Compute Engine VM instance of type e2-medium with 2 vCPUs and 4.0 GB RAM.

The code keeps track of time for certain tasks like data preprocessing and model training, and stores them in a text file. Metrics like CPU utilization were recorded on GCP and AWS consoles. GCP had a higher CPU utilization percentage.

In terms of training and evaluation time (in seconds), we found that GCP had better overall metrics

AWS: Training time before fine-tuning: 2511.556998729706

Training time after fine-tuning: 5079.302985668182

Total training time: 7590.8599844

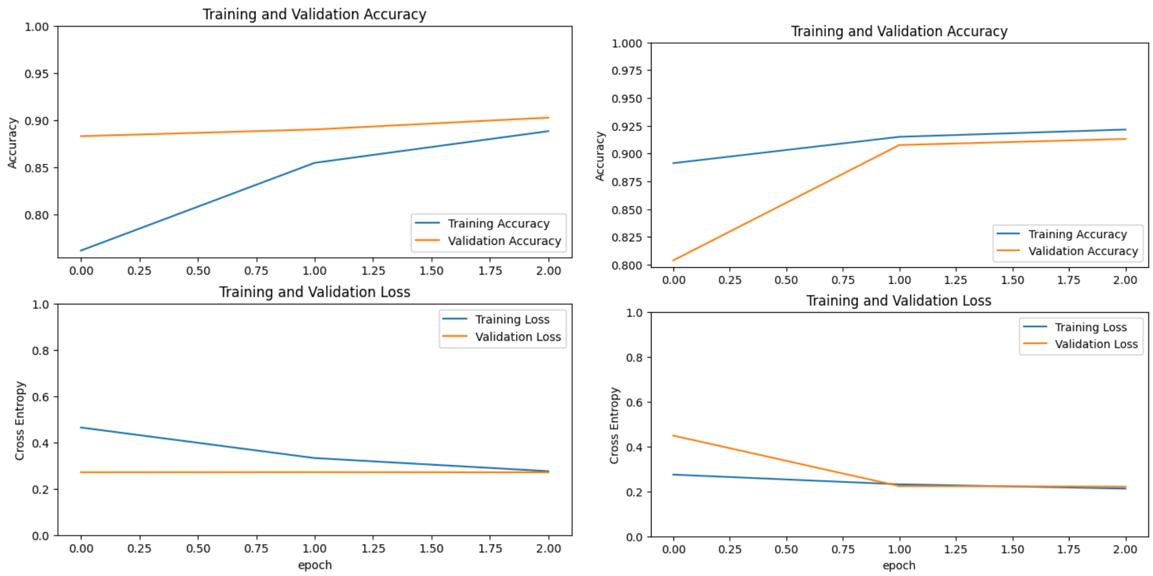
Evaluation time: 125.09700036048889

GCP: Training time before fine-tuning: 3035.6095237731934

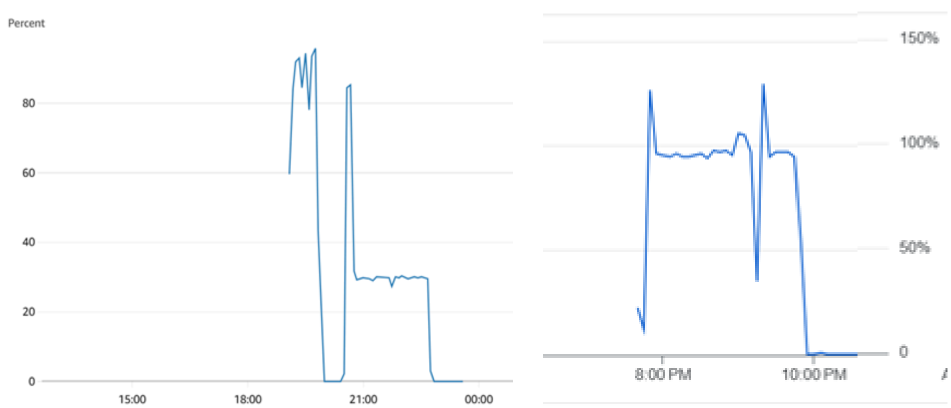
Training time after fine-tuning: 4226.838294267654

Total training time: 7262.44781804

Evaluation Time: 33.44007158279419



Accuracy and loss charts on AWS (Left)and GCP (Right)



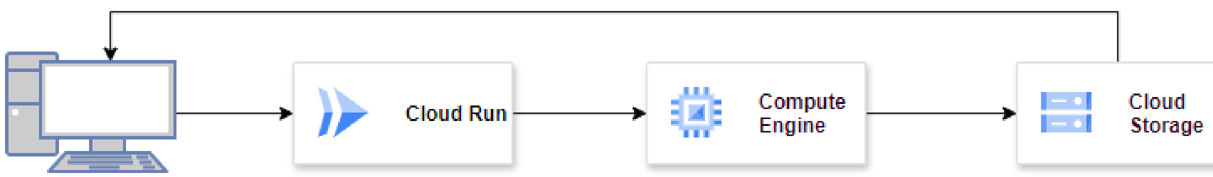
CPU Utilization (%) in AWS (Left) and GCP (Right)

Web App Development and Deployment:

We developed a web application using the Flask framework to provide a user-friendly interface for uploading skin images. It provides basic functionality like uploading images, displaying predictions, and providing information about the project. We utilized HTML, JS, and CSS.

GCP Deployment

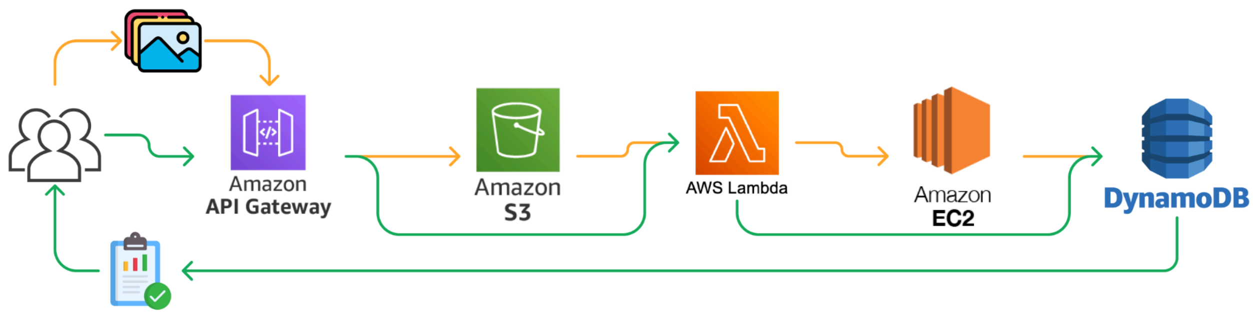
Utilizing Google Cloud services for deploying the Flask app involves harnessing the power of Cloud Run, Compute Engine, and Cloud Storage. Cloud Run efficiently manages containerized application deployment, leveraging the default Compute Engine service with comprehensive IAM permissions to ensure seamless execution. Meanwhile, Cloud Storage serves as a reliable repository for storing essential app resources, including images, prediction percentages, and software packages, ensuring accessibility and scalability as per demand.



AWS Deployment

The server configuration image below shows the setup for running an efficient web application. Here, NGINX handles incoming requests from clients in parallel and forwards them to the uWSGI and web application server. The uWSGI runs with four processes, improving the server's performance and the accuracy of request processing. By adjusting the number of processes, the server can respond to each request more quickly and is better able to handle increased load reliably and accurately. Flask is a web framework written in Python that is used to translate requests processed by uWSGI into appropriate web responses.

The AWS cloud architecture below uses AWS Lambda serverless computing to efficiently organize an image processing workflow for a machine learning model. The image data received from the client is securely stored in S3 and utilizes a pre-signed URL when sent to the ML model on EC2 via a Lambda function. The use of pre-signed URLs provides added security and protects the data from unauthorized access. EC2 uses the ML model to process the images and store the results in Amazon DynamoDB. When storing data in DynamoDB, it ensures the uniqueness of the data by assigning a UUID to each item. Images and data stored in S3 (including EC2 memory) are configured to be automatically deleted after all processing is complete, which further enhances data protection and storage space efficiency. This reduces the risk of potential privacy breaches. Incorporate ALB settings to cope with large volumes of traffic. ALB automatically balances inbound traffic and delivers it evenly across multiple EC2 instances so it can maintain reliable service even during peak-time traffic spikes. This ensures high availability and resiliency and provides users with a consistent, lag-free service experience.



3.2 Description of functionality

Image Upload: Users can upload skin lesion images through the web interface.

Image Processing: Uploaded images undergo preprocessing, including resizing and normalization, to prepare them for input into the AI model.

Diagnosis: The AI model analyzes the uploaded images and provides a diagnosis indicating the probability of skin cancer presence.

3.3 Notes on implementation

Security: Uploaded images are not saved to ensure compliance with healthcare data privacy regulations such as HIPAA.

4. Testing

4.1 Process

Model Selection

To test the accuracy of our models, we tested multiple models in the beginning steps (Inception, MobileNet, EfficientNet, Xception); from the results of validation/testing data sets, we focused on MobileNet/Inception. From here, we further trained our models on different datasets to view how the performance is affected. Finally, we fine-tuned our models based on the results of testing/validation to ensure that we were not overfitting or underfitting. Finally, once we had properly fitted our models and completed all preliminary testing, we began to look at the results of testing on the same dataset (Provided by ISIC | International Skin Imaging Collaboration) giving us the results of InceptionV3 having an average testing accuracy of 87% and the average of MobileNetV2 being 92%. Though testing on different datasets can affect the overall accuracy, results for our model MobileNet performed better overall, leading us to deploy it as our model. Testing the models involved setting aside a portion of the dataset for two central portions: training and validating. The first portion takes a segment of the dataset, runs the images (Labeled) through the model, and adjust weights based on the accuracy of the model; once this step is completed, we run the trained model on a validation set that compares the result from the model's classification versus the actual label. This step gives us our model's accuracy for the specific dataset.

Test Accuracy

While our model used a validation dataset to evaluate the accuracy as it is trained, we also set aside images from the dataset to serve as a test dataset. The test dataset comprises images that the model did not train on. We used our resulting model to classify the images on the unseen data several times.

4.2 Results

We ran the model several times on the test dataset to classify the various unseen skin cancer lesion images. We obtained an accuracy of approximately 90.6% on the new, unseen data.

```
229/229 [==============================] - 21s 89ms/step - loss: 0.2416 - accuracy: 0.9061
Test accuracy : 0.9061135649681091
```

5. Broader context

5.1 Discuss each of the four broader context areas considering your final design

Public health and Well-being

- The app directly contributes to public health by assisting in the early detection of skin cancer.
- Detecting cancerous tissue accurately can lead to timely medical intervention, better treatment outcomes, and potentially saving lives.
- By prioritizing accuracy and performance, the app enhances overall health by empowering users to take proactive steps toward their well-being.

Global, Cultural, and Social Impact

- Skin cancer affects people worldwide.
- Our app provides a tool for skin health assessment.
- It promotes awareness about skin cancer, irrespective of language or cultural differences.

Environmental Impact

- While not directly related to environmental conservation, our app indirectly impacts the environment.
- Early detection reduces the need for extensive treatments, which can have environmental implications (such as reduced medical waste).
- Additionally, raising awareness about skin health contributes to a more informed and health-conscious global population.

Economic Impact

- Economically, our app can lead to cost savings in healthcare systems.
- Early detection avoids expensive late-stage treatments.
- Moreover, it empowers individuals to make informed decisions about seeking medical attention, potentially reducing healthcare costs.

6. Conclusions

6.1 Review progress

Originally, we believed that our application would have the ability to run on GCP/AWS. We were able to obtain this goal. In the AI portion, initially, we discussed the possibility of creating a simple image recognition model. After attempting to train and build a model, our team shifted focus to transfer learning as our solution to the image recognition portion. This allowed us to have the ability to focus on the cloud computing aspect of our project and also create benchmarks that will enable us to compare the efficiency and performance of the two cloud providers when it comes to training models. Additionally, it allowed us to focus on our finished product and create a well-rounded UI interface and fully functioning application capable of giving an accurate reading of an image and displaying whether or not it may be cancerous.

6.2 Discuss value your design provides toward the problem and for users

We decided to deploy our application on the Cloud to allow for a broader user base. Our target audience is in the medical field, in which, historically, equipment in terms of computers/servers is farther behind most industries. Due to this, we wanted to make our application something that is not computationally intensive on the user's device. Deploying to AWS/GCP makes it so that hospitals can use our application regardless of their devices. We also wanted to ensure that our application would be legally usable by the medical community, which is why we decided not to save user data; this makes sure that we do not break guidelines such as HIPAA.

6.3 Potential future steps (for technical development AND providing value for users/society)

Potential future steps can include the following: Increased testing of different cloud-based services to determine what is the most cost-effective model currently. We have tested the free tier options of both AWS and GCP, but there may be tiers that cost money that can lead to increased efficiency for model training. Secondly, we can have further improvements on the model. Our model is built and trained with transfer learning and fine-tuning, but due to our limited knowledge of machine learning, there is room for improvement for our model, so anyone furthering this project may want to focus their efforts there. It would also be beneficial to find and use datasets with diverse skin colors as the ISIC datasets have skin lesion images of light-skinned individuals.

Appendix 1 – Operation Manual

1. **Accessing the Application:**

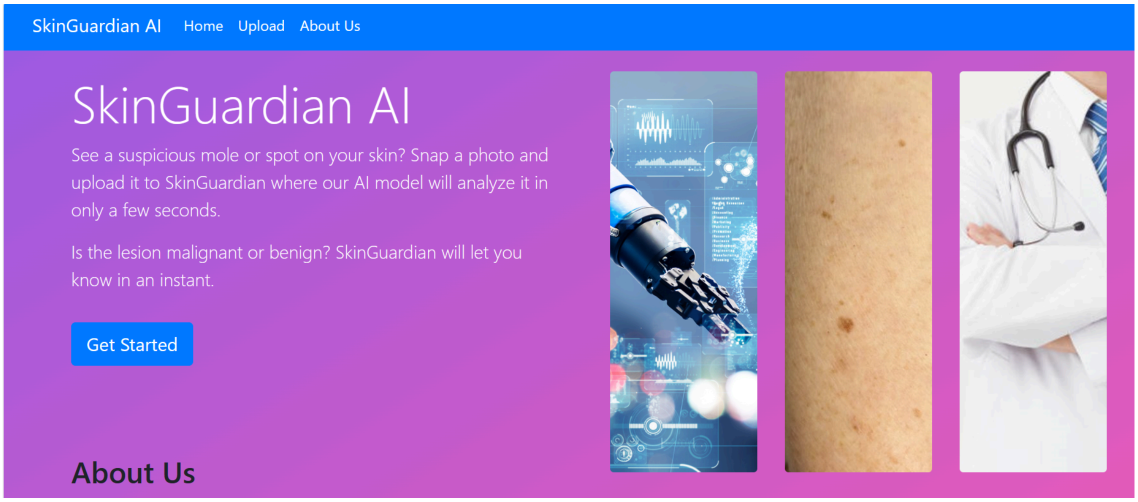
Our Skin Cancer Detection Application can be accessed through either of the following links:

GCP:

Home - Skin Cancer Detection (skincancerprediction-okqsgc3hrq-uc.a.run.app)

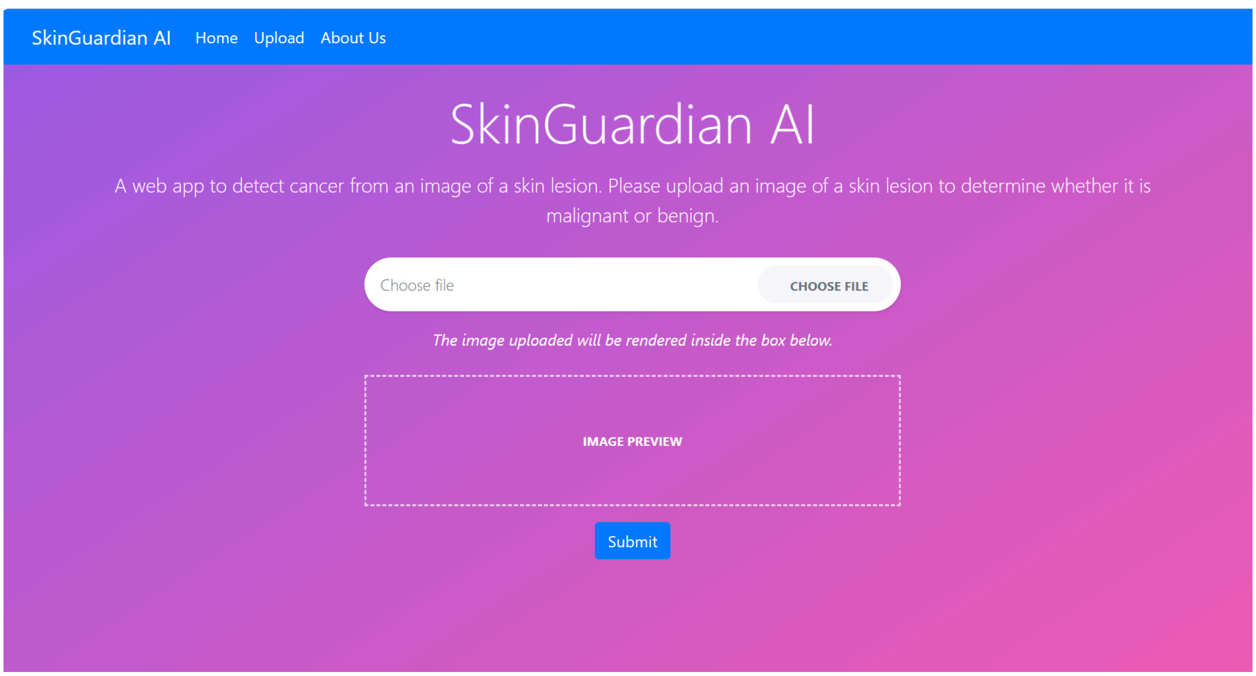
AI:

Skin Cancer Detection (skincancerdiag-sdmay24-07.com)



2. **Navigating to Upload:**

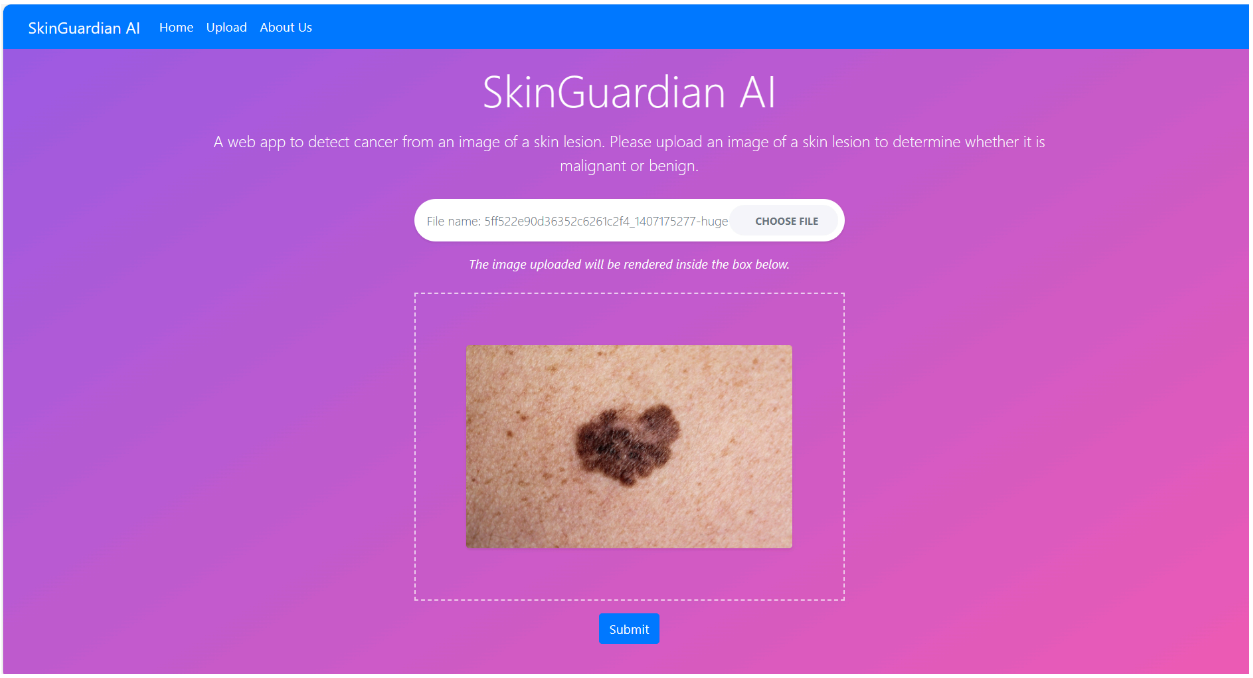
The Upload page can be accessed by clicking 'Upload' on the navigation bar.



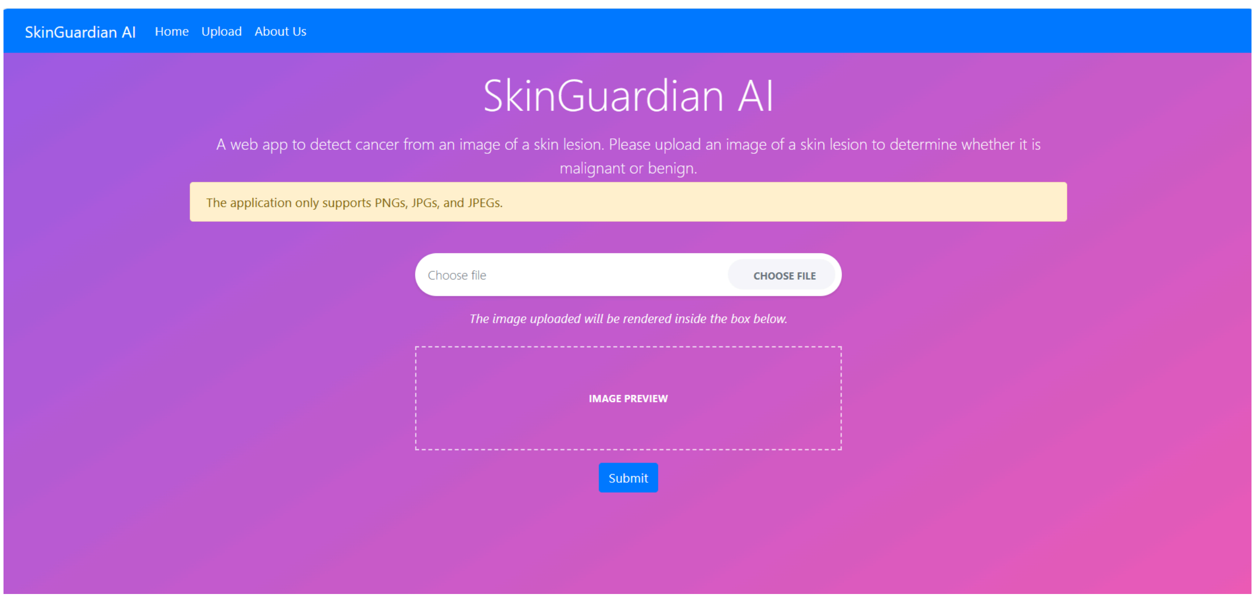
3. **Uploading an Image:**

You can click the 'Choose File' button to upload a close-up image of a suspicious mole or spot on your skin. Then click 'Submit.'

Valid Image Upload:

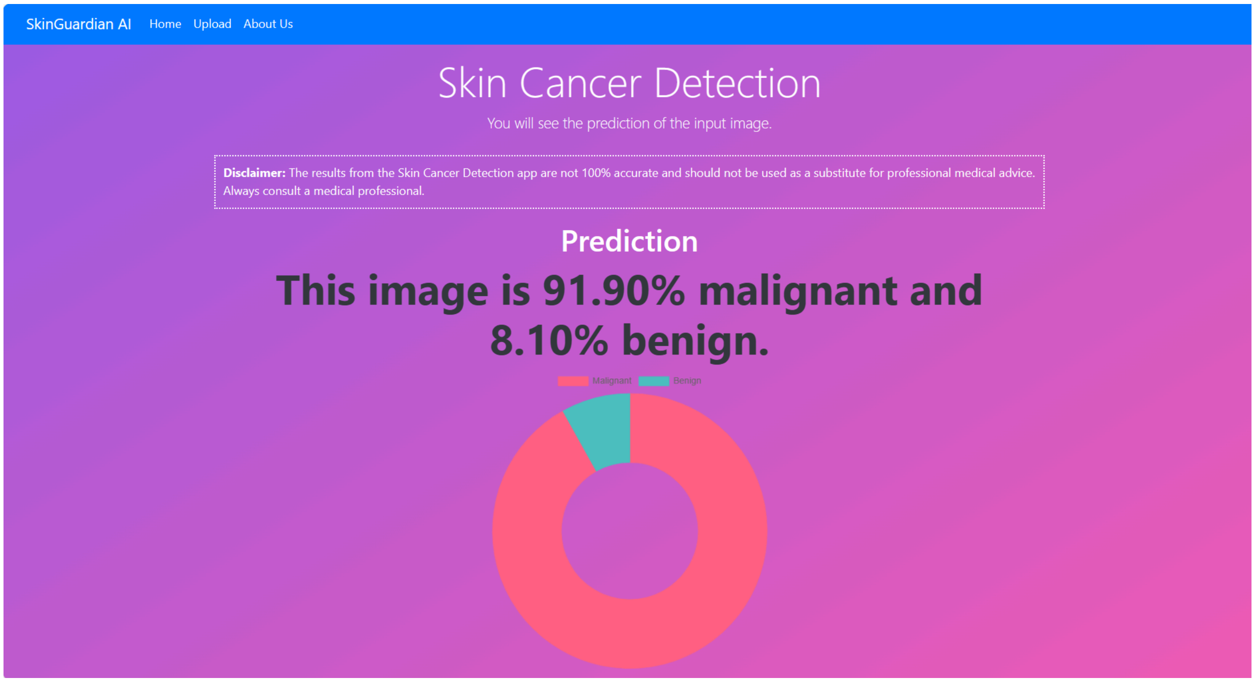


Invalid Image Upload after clicking 'Submit':



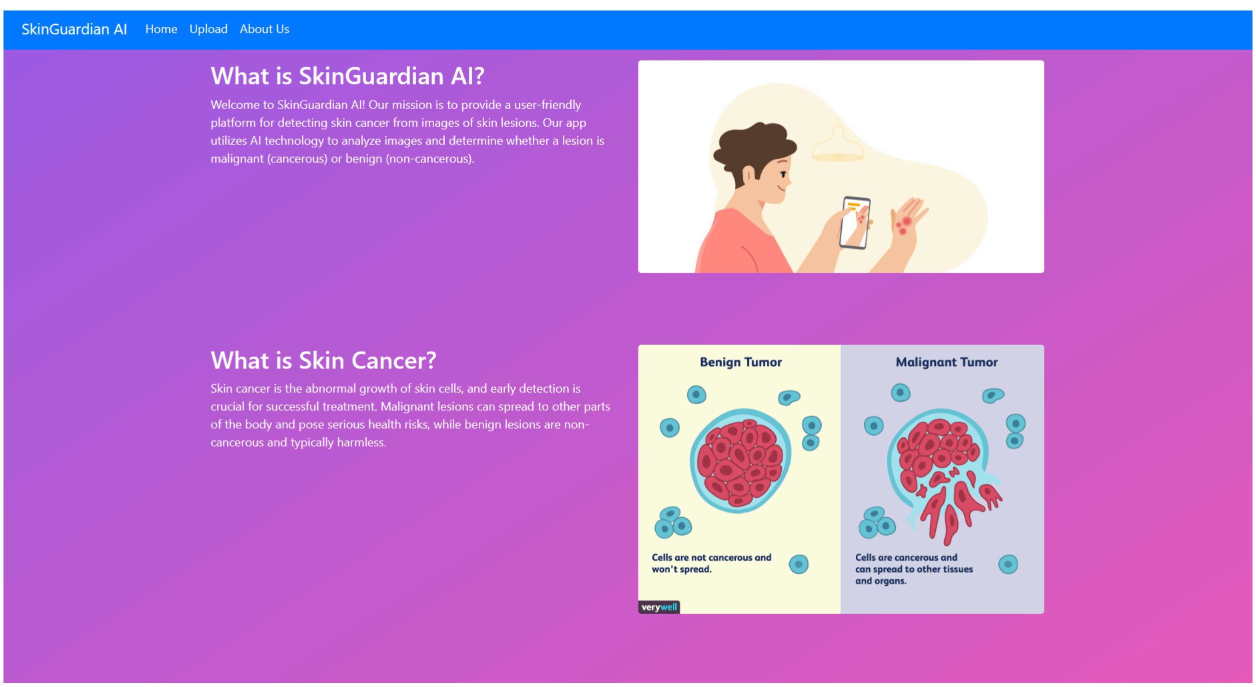
4. **Viewing Results:**

If the image upload is successful, it will take you to the Result page, where the model outputs its prediction as a percentage (malignant being cancerous and benign being non-cancerous). If the file uploaded is invalid, it displays an error message instead.



5. **Other Navigation:**

The app also has an 'About Us' section with information about the app, HIPAA, and basic cancer terminology.



Appendix 2 – Alternative/initial version of design

- Versions considered before client's specifications have changed

Originally, we had planned to use a simple model and manually train layers. After receiving feedback about feasibility and how this is out of the scope for the project, we settled for transfer learning to train a model. We also thought of hosting the model locally which was extremely computationally intensive on the devices. Later, we moved our project to the cloud.

- Versions considered before learning more about the project

Originally, we believed we would be able to create a model that performed well enough to be utilized to analyze images and define them as malignant or benign. We soon realized that this is a much more complicated task than we were able to tackle and led us to finding different avenues to complete our goal of image recognition (transfer learning).

Appendix 3 – Other considerations

- Any miscellany you deem important, what you learned, anything funny, anecdotes from your project experience
- Working as a team has taught us the importance of effective communication, coordination, and collaboration. We have learned to leverage each team member's strengths and expertise to achieve our goals.
- Learning to adapt to new requirements and feedback has been a valuable lesson since we encountered changes and challenges throughout the project.
- A project of this scale allowed us to explore new concepts, technologies, and methodologies, which has been an enriching experience for all of us.
- Each accomplishment from reaching milestones to overcoming technical challenges has strengthened our team spirit and bond.
- We are grateful for the opportunity to work on a project that has not only allowed us to apply our knowledge and skills in a real world context but also fostered our personal and professional growth. Also, this project provided practical experience that will undoubtedly benefit us in the future.

Appendix 4 – Code

Our code can be viewed here: [sd / sdmay24-07 · GitLab (iastate.edu)]

- app.py and model.py are used for the Flask application
- Model-training.ipynb is a Jupyter notebook with our initial AI/ML code before modifying it to train on the Cloud.
- The templates contain HTML files for each webpage in the application
- The static folder contains JS and CSS code for the application
- The model-training-on-cloud folder contains the code used to train the model on AWS and GCP