SKIN LESION DETECTION APP

Nathan Philip Bulla

5th December 2023

Abstract: Deep Learning-Based Skin Lesion Detection App for Multiclass Classification of Dermatological Conditions

This project presents a user-friendly app utilising deep learning to analyse skin lesion images and classify multiple dermatological conditions, including melanoma, BCC, keratoses, vascular lesions, etc. Trained on a diverse dataset, the app employs Deep learning models for accurate identification. The interface allows seamless image uploads, producing informative outputs with probability scores. Emphasising high sensitivity and specificity, the app serves as a preliminary assessment tool for early detection.

1.0 Introduction

Skin cancer is a prevalent global health concern, necessitating effective early detection methods. It is evident that advancements in technology, particularly in the field of deep learning and image analysis, offer promising solutions for timely skin lesion detection. Skin cancer, including melanoma, basal cell carcinoma (BCC), and various keratoses, poses a substantial threat, with rising incidence rates globally. And Skin cancer being the most common condition in the United States of America alongside many more countries. Traditional diagnostic methods often require expert consultation, leading to delays in diagnosis and treatment. The purpose of this project is to address this critical gap by developing a user-friendly skin lesion detection app leveraging deep learning techniques.

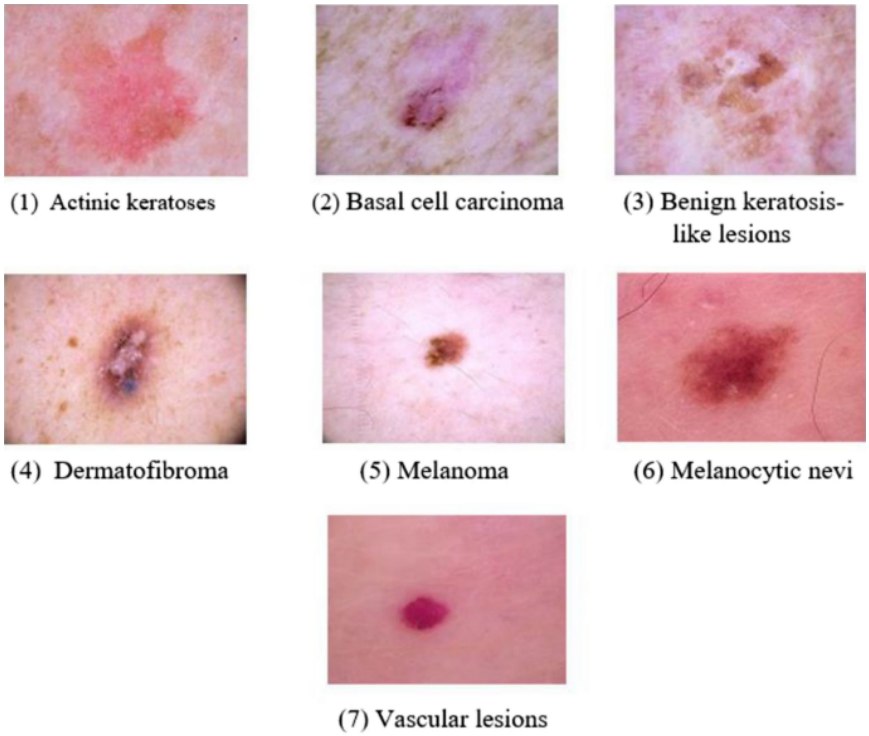


Fig. 1 Showcasing samples of Skin lesions denoting the different identifiable Skin diseases and cancers.

The skin lesion detection app aims to provide users with a preliminary assessment of dermatological conditions based on uploaded images. While the app cannot replace professional medical consultation, it serves as an accessible tool for early detection, potentially reducing delays in seeking medical advice.

The objectives for the project include to:

- Develop a user-friendly interface for easy image uploads and result interpretation with the trained DL model at the backend.
- Train the deep learning model on a diverse dataset for accurate multiclass classification of skin lesions.

- Emphasise high sensitivity and specificity in detecting various dermatological conditions.
- Provide clear and informative outputs, including probability scores for each identified condition.
- Conduct rigorous testing and validation, comparing app performance with existing diagnostic methods.
- Highlight the app's role as a preliminary assessment tool and emphasise the importance of professional medical consultation for comprehensive diagnosis and guidance.

1.1 Initial Needs Statement (Times New Roman, 14, left)

This app should allow users to upload skin lesion images for a preliminary assessment of dermatological conditions. The image must be evaluated by the trained model and predict the class of skin disease detected from the patient's image. The development of such a tool aligns with the broader objective of enhancing early detection and facilitating prompt medical intervention.

2.0 Customer Needs Assessment

Table 1. Initial Customer Needs List

1	**User-Friendly Interface |** Easy navigation and intuitive design for image uploads and result analysis
2	**Accurate Classification |** Reliable identification of various skin conditions.
3	**Rapid Processing |** Quick analysis of uploaded images for timely results.
4	**Educational Output |** Informative outputs explaining identified conditions.
5	**Privacy Assurance |** Stringent measures to ensure user data confidentiality.

3.0 Revised Needs Statement and Target Specifications

3.0.0 User Interface:
Objective: Develop an intuitive and user-friendly interface.
Design Criteria: Navigation should be straightforward, allowing users to easily upload images for analysis.

3.0.1 Classification Accuracy:
Objective: Ensure accurate classification of skin lesions.
Design Criteria: The app should demonstrate high sensitivity and specificity in identifying various dermatological conditions.

3.0.2 Rapid Image Processing:
Objective: Implement rapid image processing for timely results.
Design Criteria: The app should analyse uploaded images swiftly, providing results within an acceptable time frame.

3.0.3 Educational Output:
Objective: Generate informative outputs explaining identified conditions.
Design Criteria: The app should present clear and concise information about detected skin conditions, aiding users in understanding the results.

3.0.4 Privacy Measures:
Objective: Assure user privacy and data confidentiality.
Design Criteria: The app should adhere to stringent privacy standards, ensuring secure handling of user data.

4.0 External Search

4.1 Benchmarking

4.1.1 ‘Miiskin’
Miiskin is a premium (or paid) app. It takes pictures of the majority of your body using high-resolution photography. The application enables the user to track changes over time by comparing individual moles.

4.1.2 ‘MoleMapper’
The creation of MoleMapper stemmed from a cancer biologist's attempts to support his spouse. This software was developed in partnership with Apple and Sage Bionetworks by Oregon Health & Science University. It is freely accessible. With no monthly in-person visits, OHSU provides physicians with guidance on how to monitor worrisome lesions.

4.1.3 ‘MoleScope’
Customers need to buy an accessory for their smartphone. A dermatologist receives photos taken with the attachment and provides an online opinion.

4.1.4 ‘SkinVision’
This premium app was created by a board of dermatologists. The software analyses your mole using a deep learning algorithm

4.2 Applicable Standards

4.3.1 Health and Safety Standards:

ISO 13485 - Medical devices - Quality management systems: This standard outlines the requirements for a quality management system in the design and manufacturing of medical devices.

IEC 62304 - Medical device software - Software life cycle processes: This standard provides guidelines for the software development life cycle processes for medical device software.

4.3.2 Data Privacy and Security Regulations:

General Data Protection Regulation (GDPR): Applicable as our project involves handling personal data of individuals who reside within the European Union. It sets out rules for data protection and privacy.
Health Insurance Portability and Accountability Act (HIPAA): Relevant as the app may deal with health information in the United States. It establishes standards for the secure handling of protected health information.

4.3.3 Medical Device Regulations:

FDA Guidance for Software as a Medical Device (SaMD): Is applicable to our project, as the U.S. Food and Drug Administration provides guidance for software that is intended to be used for medical purposes.

4.3.4 Accessibility Standards:

Web Content Accessibility Guidelines (WCAG): Ensures that digital content is accessible to people with disabilities, which is important for inclusive app design.

4.3 Applicable Constraints

4.4.1 Internal Constraints:

Expertise and Skill Sets:
Impact: The availability of skilled personnel in areas such as deep learning, medical software development, and user interface design is critical. A lack of expertise may affect the quality and functionality of the app.

Time Constraints:
Impact: Project timelines can influence the depth of research, testing, and iterative design processes. Balancing the need for a timely release with thorough development is crucial.

Regulatory Compliance:
Impact: Adhering to various regulations, including data privacy laws and medical device regulations, can impose constraints on development timelines and features.

4.4.2 External Constraints:

Market Competition:
Impact: The competitive landscape can influence feature prioritisation and marketing strategies. Understanding market trends and user expectations is vital for successful execution.

Health and Safety Regulations:
Impact: Adherence to health and safety standards, such as regulations for medical software, is mandatory. Non-compliance can lead to legal issues and hinder market access.

Data Privacy Laws:
Impact: Strict data privacy laws (e.g., GDPR, HIPAA) impact how user data is handled. Failure to comply can result in legal consequences and damage the app's reputation.

Technological Advances:
Impact: Rapid advancements in technology may present challenges in keeping the app up-to-date. Continuous monitoring of technological trends is necessary for long-term relevance.
Like updating the algorithm and model to iteratively improve its accuracy and reliability across time.

5.0 Concept Generation

Alternatively, this software could be marketed in other forms towards medical professionals rather than common users, as the app can not under most circumstances replace professional diagnosis, but can supplement and aid it. The following sections detail the alternatives based on the core feasibility criteria with respect to the current proposed design to be kept in mind

5.0.1 Feasibility Screening Criteria:

Technical Feasibility: Assessing whether the proposed design could be technically implemented given the available resources and expertise.

Compliance with Standards: Ensuring that the conceptual designs aligned with relevant health and safety, data privacy, and medical device standards.

Budget and Resource Requirements: Evaluating the estimated budget and resource requirements for each design option, considering the project's financial constraints.

User Acceptance: Considering whether the design aligns with user expectations, preferences, and usability principles, as validated through feedback and customer input.

5.0.2 Feasible Alternative Conceptual Designs:

AI-Driven Mobile App:
Description: Utilising advanced deep learning algorithms for image analysis, this design focuses on a mobile application with real-time skin lesion detection capabilities.
Feasibility Assessment: Technically feasible with the availability of AI expertise. Compliance with standards requires rigorous testing for accuracy and safety. Moderate budget and resource requirements.

Cloud-Based Processing:
Description: Offloading image processing to a cloud-based platform to enhance rapid analysis. Users upload images via the app, and results are delivered swiftly.
Feasibility Assessment: Technical feasibility depends on robust cloud infrastructure. Compliance with data privacy laws is critical. Potentially higher budget requirements for cloud services.

Hybrid Model with Dermatologist Review:
Description: Combining automated analysis with a feature allowing users to request a dermatologist review for certain cases, providing an extra layer of expert scrutiny.
Feasibility Assessment: Technical feasibility with a hybrid approach. Compliance with medical device regulations for the dermatologist-reviewed cases. Moderate budget requirements.

Educational Chatbot Integration:
Description: Introducing a chatbot feature that interacts with users, providing educational information about detected conditions and guiding them on the next steps.
Feasibility Assessment: Technically feasible with natural language processing capabilities. Compliance with data privacy laws is essential. Moderate budget requirements for chatbot development.

7.0 Final Design

7.1 How does it work?

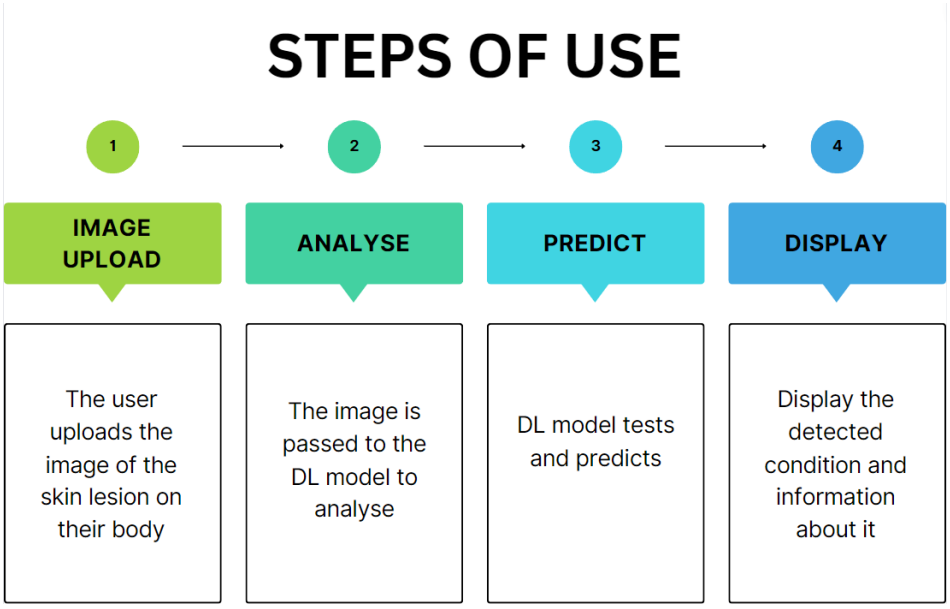


Fig. 2 Flow of the application

7.1.1 Getting Started:

Download and Install:
Download the Skin Lesion Detection App from your device's app store.
Install the app and launch it.

User Registration:
Register for an account to access the app's features.
Follow the on-screen prompts to provide necessary information.

7.1.2 Image Upload and Analysis:

Navigate to Image Upload:
Open the app and select the "Upload" option from the main menu.

Capture or Select Image:
Capture a photo using your device's camera or choose an existing image from your gallery.

Review and Confirm:
Review the selected image and confirm your choice.

Wait for Analysis:
The app will process the image using advanced deep learning algorithms. The analysis may take a few moments.

View Results:
Once the analysis is complete, the app will display the results indicating the likelihood of various skin conditions. Results are presented in a clear and informative format.

7.1.3 Educational Features:

Condition Details:
Explore detailed information about the identified skin condition, including potential causes, symptoms, and recommended next steps.

Chatbot Assistance:
Utilise the chatbot feature for interactive assistance and additional information. The chatbot is designed to provide educational content and answer queries.
Maintenance and Service:

7.1.4 App Updates:

Regularly check for updates in the app store to ensure you have the latest features and improvements.
Troubleshooting:

If you encounter any issues, refer to the app's "Help" section for troubleshooting tips or contact customer support for assistance.

7.1.5 User Privacy and Security:

Data Confidentiality:
User data is handled with utmost confidentiality and is encrypted to protect privacy.

Secure Transactions:
Any transactions or data exchanges within the app are conducted securely to prevent unauthorised access.

7.1.6 Customer Assistance:

Customer Support:
For any questions or concerns, reach out to customer support through the app's "Contact Us" feature.

Feedback:
The app values user feedback. Use the feedback option to share your thoughts and help us improve.

7.2 Data Sources

Some of the largest dermatological datasets are found within and as part of the International Skin Imaging Collaboration (ISIC) Archive, with over 76,108 public images, and 240,655 total images, encompassing multiple public datasets.

As a viable proposition, we may also collect user data in accordance with data privacy laws to enhance our dataset.

7.4 Algorithms, Frameworks and Software

Algorithms with deep architectures are ideal, such as ResNets and its derivatives, but for the application to be highly responsive and efficient, we may even consider EfficientNet or YOLO classification algorithms.

7.3 How is it manufactured and assembled, and what does it cost?

Manufacturability:

7.3.1 Software Development:

7.3.2 Cloud-Based Infrastructure:

The cloud-based infrastructure supporting image processing can be easily scalable based on demand, ensuring efficient handling of varying workloads. Services such

7.3.3 AI processing

Collaborating with AI and deep learning experts during development ensures the development and implementation of robust algorithms and models for the task of detection and diagnosis of skin lesion images received from users of the app, trained upon the large datasets of data, from sources such as ISIC.

7.3.4 Quality Assurance Testing:

Rigorous testing protocols are essential for ensuring the quality and reliability of the app. Automated testing tools can be employed to streamline this process.

7.3.5 User Interface Design:

The user interface design involves creating an intuitive and user-friendly layout.

Collaboration with UI/UX designers is crucial for manufacturability, as it influences user adoption.

Cost Considerations:

7.3.6 Development Costs:
Development costs include expenses related to software development, Deep learning algorithm creation, collaboration with experts.
Such as cost for the development platform, maintaining the cloud platform, the infrastructure used for running the deep learning task, et cetera.

7.3.7 Cloud Service Expenses:
Cloud-based image processing incurs ongoing expenses. Choosing scalable and cost-effective cloud services helps manage this aspect efficiently.

7.3.8 Data Privacy Compliance:
Ensuring compliance with data privacy regulations may involve costs related to legal consultations, data encryption technologies, and security measures.

7.3.9 User Interface Design:
Designing an intuitive user interface may involve costs related to hiring UI/UX designers. Investing in a well-designed interface contributes to user satisfaction and adoption.

7.3.10 Quality Assurance:

Quality assurance testing is a critical aspect of cost considerations. Allocating resources for thorough testing helps identify and address issues before the app is released.

7.3.11 Updates and Maintenance:
Ongoing costs for app updates, maintenance, and customer support should be factored into the budget. Regular updates ensure the app stays relevant and secure.
Steps for Cost Management:

8.0 Small Scale Code Implementation

Link to the full code:
https://github.com/NathanPhilipB/Skin_Cancer_Classification_NATHANPHILIPB

Link to Google Colaboratory:
https://colab.research.google.com/drive/1dqUP6Hmd6leHrTQVUa6lOyRedLcOlbr8?usp=sharing

The following is a small scale snippet of the code to analyse skin lesions and classify the skin diseases detected.

CLAHE

```python
import os
import cv2

# Define the path to the training dataset folder
train_folder = "path/to/train_folder"

# Function to apply CLAHE to an image
def apply_clahe(image_path):
    # Read the image using OpenCV
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Create a CLAHE object
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

    # Apply CLAHE to the image
    clahe_img = clahe.apply(img)

    return clahe_img

# Function to apply CLAHE to all images in a folder
def apply_clahe_to_folder(folder_path):
    if not os.path.exists(folder_path):
        print(f"Folder {folder_path} does not exist.")
        return

    # List all image files in the folder
    image_files = [f for f in os.listdir(folder_path) if f.endswith((".png", ".j
```

BLACK HAT FILTERING

```python
import os
import cv2

def morph(image):
    grayScale = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY )
    kernel = cv2.getStructuringElement(1,(17,17))
    blackhat = cv2.morphologyEx(grayScale, cv2.MORPH_BLACKHAT, kernel)
    ret,thresh2 = cv2.threshold(blackhat,10,255,cv2.THRESH_BINARY)
    dst = cv2.inpaint(image, thresh2,1,cv2.INPAINT_TELEA)
    return dst

def process_dataset(dataset_dir, output_dir):
    #for split in ['train', 'val', 'test']:
    for split in ['train', 'val']:
        split_dir = os.path.join(dataset_dir, split)

        for class_name in os.listdir(split_dir):
            class_dir = os.path.join(split_dir, class_name)
            output_class_dir = os.path.join(output_dir, split, class_name)

            os.makedirs(output_class_dir, exist_ok=True)

            for image_name in os.listdir(class_dir):
                image_path = os.path.join(class_dir, image_name)
                output_image_path = os.path.join(output_class_dir, image_name)
                image = cv2.imread(image_path)
                hair_removed = morph(image)
                cv2.imwrite(output_image_path, hair_removed)

```

```python
    for image_file in image_files:
        image_path = os.path.join(folder_path, image_file)
        clahe_img = apply_clahe(image_path)

        # Save the CLAHE-enhanced image back to the same location
        cv2.imwrite(image_path, clahe_img)

# Apply CLAHE to the training dataset
apply_clahe_to_folder(train_folder)

```

```python

dataset_dir = '/content/ISIC - 2019/'
output_dir = '/content/ISIC - 2019_No-hair/'
#dataset_dir = '/content/small_data/'
#output_dir = '/content/small_data_blackhat/'
process_dataset(dataset_dir, output_dir)
```

Fig. 3.0 Code snippets of the Preprocessing

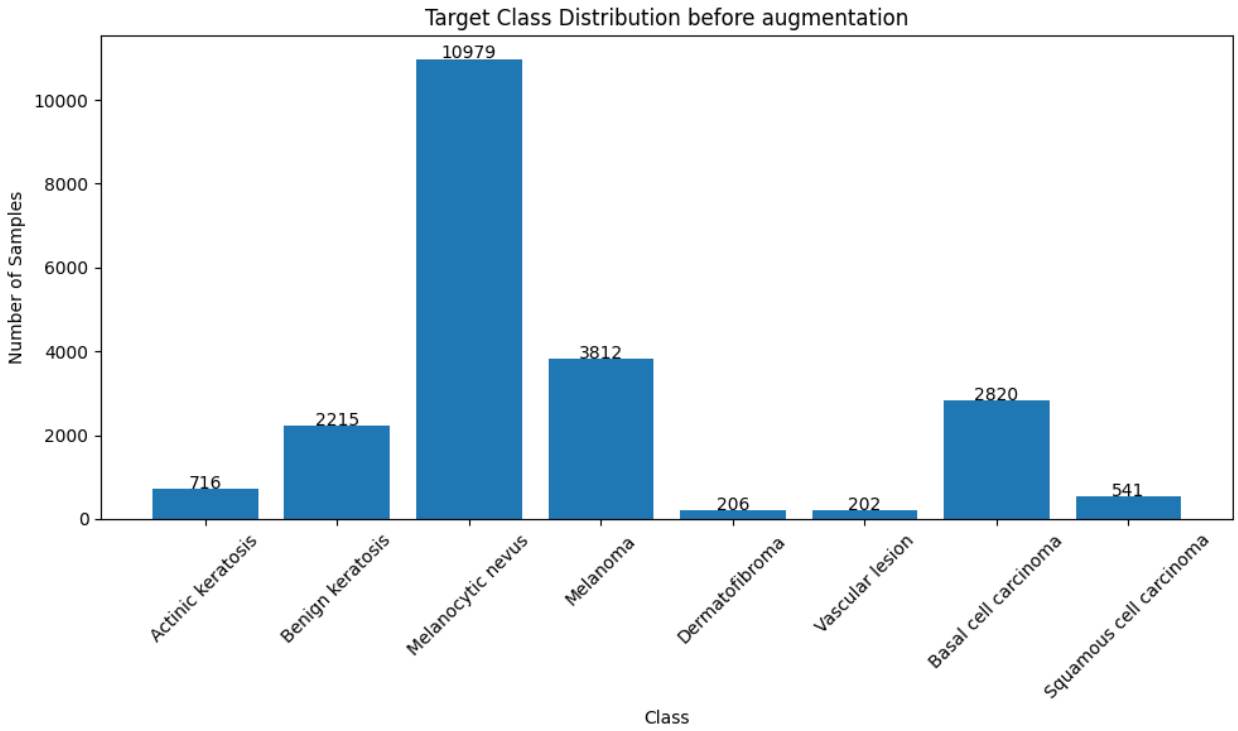
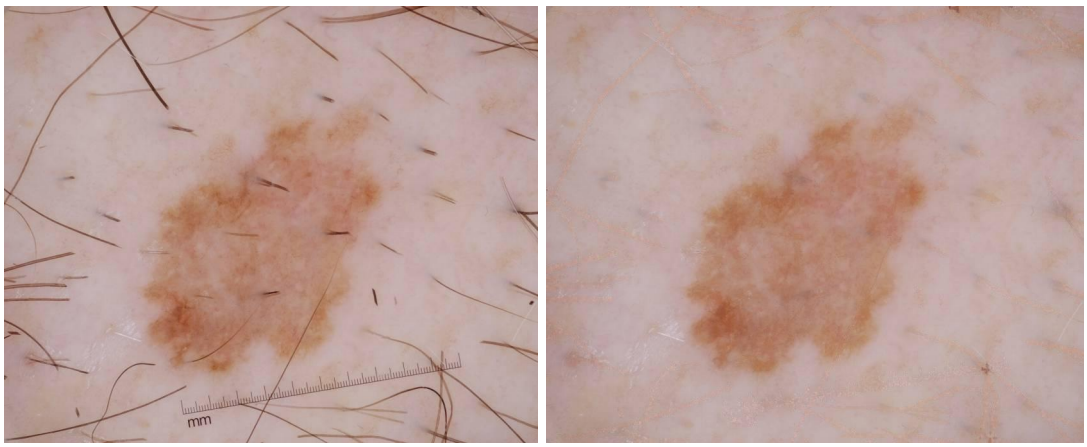


Fig. 3.1 Class distribution across a small subset of the ISIC 2019 challenge dataset



(a) (b)

Fig. 4 (a) Image before preprocessing (hair removal) (b) Image after hair removal

```
1 import os
2
3 from ultralytics import YOLO
4 model = YOLO("yolov8m-cls.pt")
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8m-cls.pt to 'yolov8m-cls.pt'...
100%|██████████| 32.7M/32.7M [00:00<00:00, 163MB/s]
```

```
[ ]  1 results = model.train(data='/content/ISIC - 2019', epochs=10, imgsz=64)
```

```
Ultralytics YOLOv8.0.145 🚀 Python-3.10.6 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=classify, mode=train, model=yolov8m-cls.pt, data=/content/ISIC - 2019, epochs=10, patience=50, batch=16, imgsz=64, save=True, save_period=-1, c
Overriding model.yaml nc=1000 with nc=8

                   from  n    params  module                                       arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f             [96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f             [192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f             [384, 384, 4, True]
  7                  -1  1   2655744  ultralytics.nn.modules.conv.Conv             [384, 768, 3, 2]
  8                  -1  2   7084032  ultralytics.nn.modules.block.C2f             [768, 768, 2, True]
  9                  -1  1    995848  ultralytics.nn.modules.head.Classify         [768, 8]
YOLOv8m-cls summary: 141 layers, 15782584 parameters, 15782584 gradients, 41.9 GFLOPs
Transferred 228/230 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/classify/train3', view at http://localhost:6006/
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
AMP: checks passed ✅
albumentations: RandomResizedCrop(p=1.0, height=64, width=64, scale=(0.5, 1.0), ratio=(0.75, 1.3333333333333333), interpolation=1), HorizontalFlip(p=0.5), ColorJitt
optimizer: AdamW(lr=0.000714, momentum=0.9) with parameter groups 38 weight(decay=0.0), 39 weight(decay=0.0005), 39 bias(decay=0.0)
Image sizes 64 train, 64 val
Using 2 dataloader workers
Logging results to runs/classify/train3
Starting training for 10 epochs...
Closing dataloader mosaic

      Epoch    GPU_mem       loss  Instances       Size
```

Fig. 5 Code implementation for a simple YOLOv8 Classification model

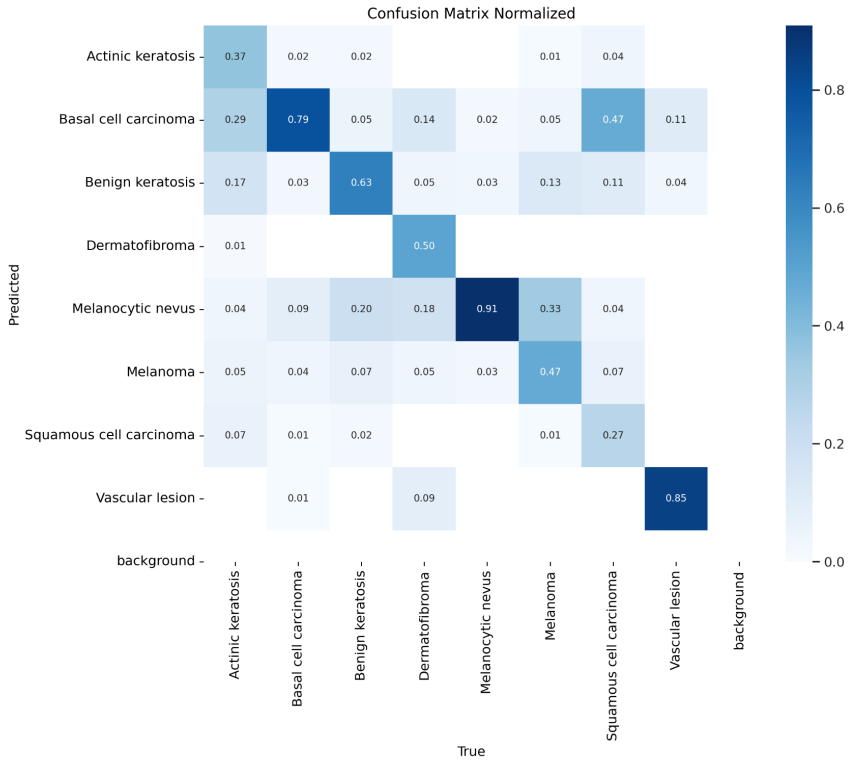


Fig. 6 Confusion Matrix of a simple small-scale YOLOv8 classification model

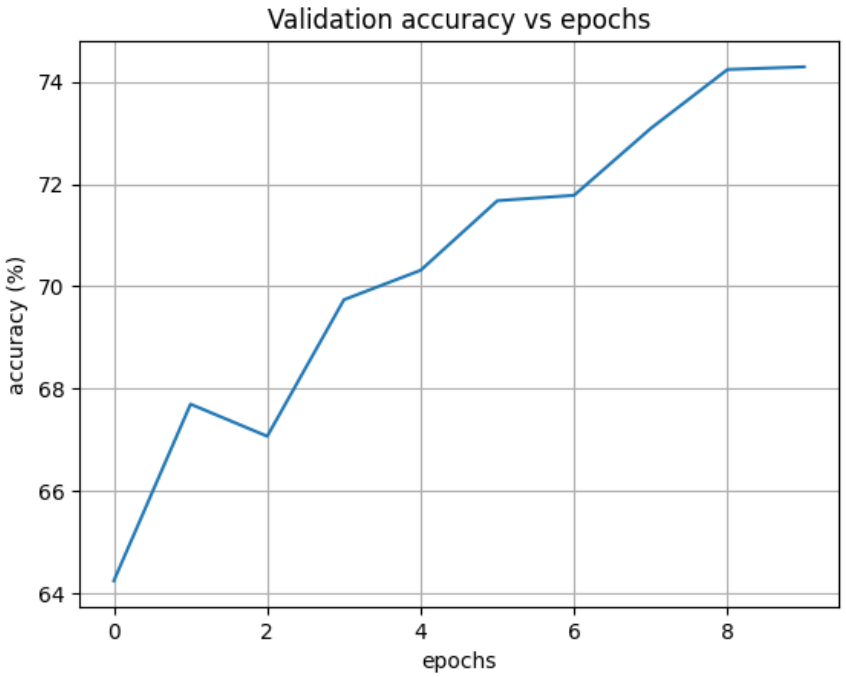


Fig.7 Accuracy graph of a small-scale simple YOLOv8 Classification model for Skin disease classification from skin lesion images.

9.0 Conclusions

In summary, the conceptualization of the Skin Lesion Detection App represents a promising avenue for advancing accessible and early skin health assessments. Through the envisioned integration of cutting-edge deep learning algorithms, cloud-based processing, and an intuitive user interface, the proposed app aims to offer users a reliable tool for identifying potential dermatological conditions. The iterative design approach, shaped by ongoing customer feedback and collaborative team efforts, forms the foundation for a prospective product that aligns with both technical standards and user expectations, fostering trust and empowerment within the user community.

As we look towards the future, the success of this potential project hinges on the dedication of a multidisciplinary team, including AI experts, software developers, and UI/UX designers. The seamless integration of technical innovation, adherence to regulatory compliance, and a commitment to user-centric design principles will be critical in transforming this concept into a tangible and impactful solution. The envisioned Skin Lesion Detection App holds the promise of bridging technological advancements with user-centric healthcare needs, pending further development, testing, and validation in the journey from concept to reality.

References

Kaggle ISIC 2019 Competition
[https://www.kaggle.com/competitions/siim-isic-melanoma-classification/data]

Google Patents

Other Medical/Dermatological apps
[https://www.oregoncancer.com/blog/apps-to-help-detect-skin-cancer]