PROJECT TITLE : SKIN FIT

Vityarthi Project

ADITYA AGRAWAL
25BAI11428

- # Introduction

I designed and implemented a machine-learning based system capable of distinguishing between vitiligo/leucoderma-affected skin and healthy skin with high accuracy.

The objective of this project is to classify skin images into 'vitiligo' or 'normal' using a Convolutional Neural Network. The early detection can support dermatological assessment for swift and effective intervention.

- # Problem Statement

Vitiligo is a depigmentation skin disorder. Manual diagnosis involves much time and subjectivity. There exists a need for the development of an automated system to classify skin images that will support clinicians and improve early screening.

- # Functional Requirements

1. Load image datasets from directories
2. Preprocess and augment images for model generalization.
3. Train a binary classification CNN
4. Validate accuracy during training
5. Predict class from a new image (vitiligo/normal)

- # Non-Functional Requirements

1. Fast inference time on new images
2. Usable by non-experts - simple input/output
3. Compatibility with Google Drive for dataset storage
4. Model persistence for use later

- # System Architecture

1. High-Level Flow
2. Data loading and preprocessing
3. Model training & validation
4. Model saving and prediction

- # Diagrams Design

1. Use Case Diagram: It depicts the actors that interact with the system, and what the system does for those actors. That is, a user who uploads an image, and in response, gets results back.
2. Workflow Diagram: The linear flow is Image Loading → Preprocessing → Model Training → Evaluation → Prediction.
3. Sequence Diagram: User provides image → System loads & resizes image → CNN predicts label → Result shown
4. Class/Component Diagram: Key components/ classes: ImageDataGenerator, Model, Train/Validation Data
5. ER Diagram: If using database, represents the images table (id, path, label); not needed for in-memory/flat file workflows

- # Design Decisions & Rationale

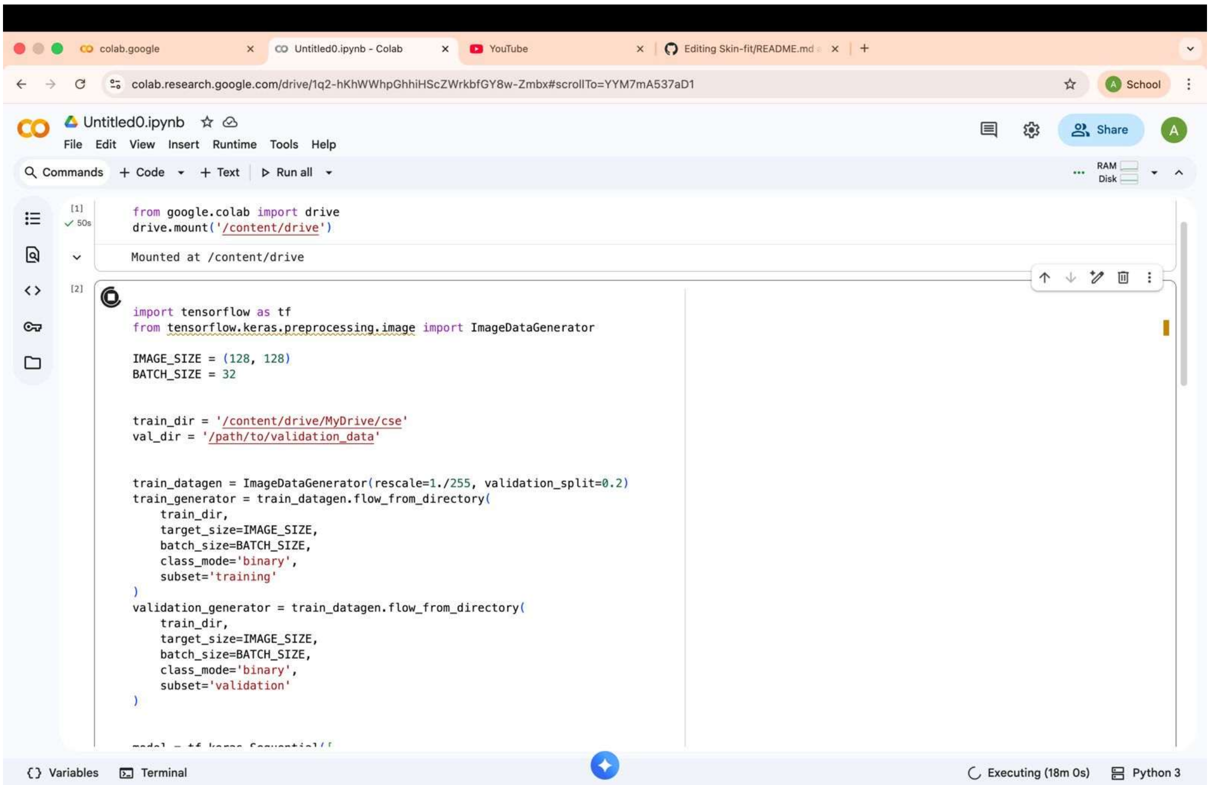
1. Chose CNN for image feature extraction and classification accuracy
2. Used Image Data Generator for augmentation to address overfitting.
3. Rescale the pixel values to normalize the input and enhance convergence.
4. Adopted binary cross-entropy as loss for two-class problem

- # Implementation Details

1. Kera's Sequential model for modularity and simplicity

2. Directory structure: Different folders for 'vitiligo' and 'normal'
3. Output: trained model ('vitiligo_detector_model.h5')
4. Code compatible with Google Colab & Google Drive

- # Screenshots / Results



colab.google | Untitled0.ipynb - Colab | YouTube | Editing Skin-fit/README.md

colab.research.google.com/drive/1q2-hKhWWhpGhhiHScZWrkbfGY8w-Zmbx#scrollTo=YYM7mA537aD1

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all

```
    class_mode='binary',
    subset='training'
)
validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)


model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=IMAGE_SIZE + (3,)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # for binary classification
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)


model.save('vitiligo detector model.h5')
```

Variables Terminal Executing (18m 7s) Python 3

colab.google | Untitled0.ipynb - Colab | Editing Skin-fit/README.md

colab.research.google.com/drive/1q2-hKhWWhpGhhiHScZWrkbfGY8w-Zmbx#scrollTo=YYM7mA537aD1

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all

```
    epochs=10,
    validation_data=validation_generator
)


model.save('vitiligo_detector_model.h5')
```

```
Found 2869 images belonging to 2 classes.
Found 716 images belonging to 2 classes.
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `su
  self._warn_if_super_not_called()
Epoch 1/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 1014s 11s/step - accuracy: 0.6152 - loss: 0.7219 - val_accuracy: 0.9288 - val_loss: 0.2319
Epoch 2/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 115s 1s/step - accuracy: 0.8444 - loss: 0.3723 - val_accuracy: 0.8450 - val_loss: 0.3682
Epoch 3/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 145s 1s/step - accuracy: 0.8534 - loss: 0.3406 - val_accuracy: 0.9064 - val_loss: 0.2566
Epoch 4/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 151s 1s/step - accuracy: 0.8798 - loss: 0.2974 - val_accuracy: 0.9330 - val_loss: 0.2500
Epoch 5/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 118s 1s/step - accuracy: 0.8821 - loss: 0.2771 - val_accuracy: 0.8757 - val_loss: 0.3074
Epoch 6/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 117s 1s/step - accuracy: 0.8949 - loss: 0.2480 - val_accuracy: 0.9330 - val_loss: 0.2115
Epoch 7/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 117s 1s/step - accuracy: 0.9206 - loss: 0.1931 - val_accuracy: 0.9190 - val_loss: 0.2481
Epoch 8/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 116s 1s/step - accuracy: 0.9173 - loss: 0.2074 - val_accuracy: 0.9539 - val_loss: 0.1571
Epoch 9/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 120s 1s/step - accuracy: 0.9401 - loss: 0.1581 - val_accuracy: 0.9092 - val_loss: 0.2600
Epoch 10/10
90/90 ━━━━━━━━━━━━━━━━━━━━ 120s 1s/step - accuracy: 0.9564 - loss: 0.1243 - val_accuracy: 0.9120 - val_loss: 0.2870
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. W
```

Variables Terminal ✓ 4:07 PM Python 3

- # Testing Approach

1. Used validation split in ImageDataGenerator for in-training evaluation

2. Test with unseen images to measure real world performance.

- # Challenges Faced

Limited and imbalanced training data

Tuning hyperparameters for best accuracy Managing image augmentation and directory formats Learnings & Key Takeaways Importance of clean, labeled data CNNs are able to model image classification problems in an efficient manner. Validation split helps to monitor overfitting. Future Enhancements Add multiclass classification for more skin conditions Integrate with databases for scalable storage. Develop web-mobile UIs for wider accessibility References Keras Documentation TensorFlow Tutorials Imaging for Diagnosis of Vitiligo: A Review

- # References

1. Keras Documentation
2. TensorFlow Tutorials
3. Research on Vitiligo Diagnosis via Imaging
4. Kaggle : https://www.kaggle.com/datasets/shinynose/vitiligo