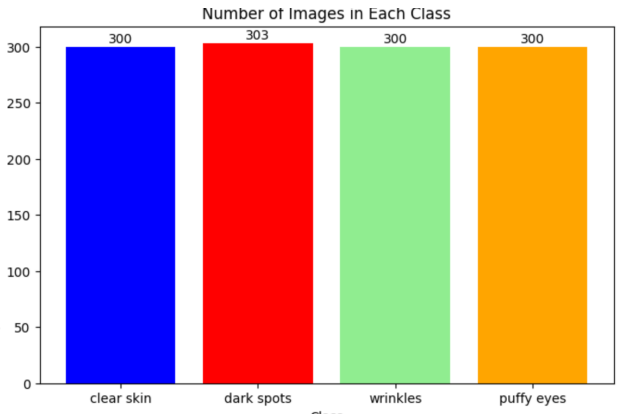
AI DermalScan Project Documentation

Introduction

This document provides a structured explanation of the preprocessing, visualization, augmentation, and dataset preparation steps used in the AI DermalScan project. The goal of this phase is to load the raw dataset, analyze class distribution, preprocess images, apply data augmentation, prepare train-validation-test splits, and verify dataset integrity.

1. Dataset Loading and Class Distribution

The raw dataset contains four classes stored in separate folders. The process starts by reading the dataset directory and counting the number of images in each class. These counts are stored in a tabular format for analysis and later visualized using bar charts. This step helps validate that each class contains a balanced number of samples before model training.



2. Image Preprocessing

A preprocessing function is created to:

- Read an image from a given path

- Convert the image from BGR to RGB format

- Resize the image to a fixed dimension of 224 by 224 pixels

- Normalize pixel values by scaling them between 0 and 1

A sample image is displayed after preprocessing to confirm that the transformation has been applied correctly.

3. Data Augmentation

A data augmentation pipeline is implemented using TensorFlow. It includes random horizontal flips, rotations, zooms, and contrast adjustments. These transformations help increase dataset variability and improve model generalization.

A function is created to generate multiple augmented versions of the same image. The original and augmented images are displayed together for visual confirmation.

4. Dataset Loading with TensorFlow

A dataset loader is implemented using the image_dataset_from_directory utility. It:

- Reads images from class folders

- Resizes them to 224 by 224 pixels

- Converts labels to one-hot encoding

- Normalizes pixel values using a rescaling layer

This dataset loader produces batches of images for model training.

5. Train, Validation, and Test Split Creation

A custom function is implemented to split the dataset into train, validation, and test sets. It performs the following steps:

- Identifies class folders in the raw dataset

- Creates new folders for the three splits

- Collects file paths of all images in each class

- Uses scikit-learn to split them into train, validation, and test portions

- Copies the images into corresponding new directories

The split is confirmed with a success message.

6. Counting Images in Each Split

A verification step is added to count images in each split and each class. This prevents issues such as missing files or incorrect folder placement. The results are stored in a table and used to generate a grouped bar chart. The chart visually confirms that all splits maintain the correct distribution.

7. Visualization

Bar charts are used to represent:

- Number of images per class in the raw dataset

- Number of images in each split across all classes

These visualizations provide clarity and help ensure dataset correctness before proceeding to model development.