Fair Melanoma Detection Technical Documentation

Table of Contents

Table of Figures

1. Introduction

This document presents the technical implementation of our melanoma classification pipeline, developed for the LUMEN Data Science 2024/25 competition. The goal of the challenge was to build a machine learning model capable of accurately detecting melanoma in dermoscopic images, while ensuring fairness across different skin tones.

Our solution adopts a two-stage architecture:

- **Segmentation**: We use a U-Net model with a **ResNet34** backbone to localize skin lesions within the images. Accurate segmentation allows the model to focus on the most relevant regions and supports downstream tasks like classification and skin tone analysis.
- **Classification**: A **EfficientNet-B4** based classifier is trained to predict malignancy from the segmented lesion areas, enhancing focus on pathological features.

Additionally, our pipeline includes a **skin tone classification** component. After lesion segmentation, the model isolates the lesion and applies a custom-designed skin mask to the remaining image. By analyzing the brightness of these background skin regions, the model estimates the skin tone category.

To mitigate overfitting, we employed a gradual fine-tuning strategy: initially training only the final layers of the model, progressively unfreezing earlier layers over several epochs, and eventually fine-tuning the entire network. We used **Focal Loss** as the loss function and optimized its parameters to handle class imbalance effectively.

For **data splitting**, we designed a strategy tailored to the dataset structure:

- For the 2020 dataset, patient IDs were available, enabling us to split data by patient to prevent data leakage.
- For others, we performed stratified splitting by target class and skin tone, balancing the number of samples where possible.
- To address the strong class imbalance, we down sampled benign cases and applied aggressive data augmentation. We also up sampled underrepresented skin tone groups to ensure fairer exposure during training.

Finally, instead of using a fixed classification threshold, we dynamically adjusted the decision threshold at inference time to optimize performance metrics such as accuracy, recall, precision, and F1 score.

For training and validation of our models performances, we used the provided ISIC datasets (2016-2020) and 127 images from the Fitzpatrick17k dataset.

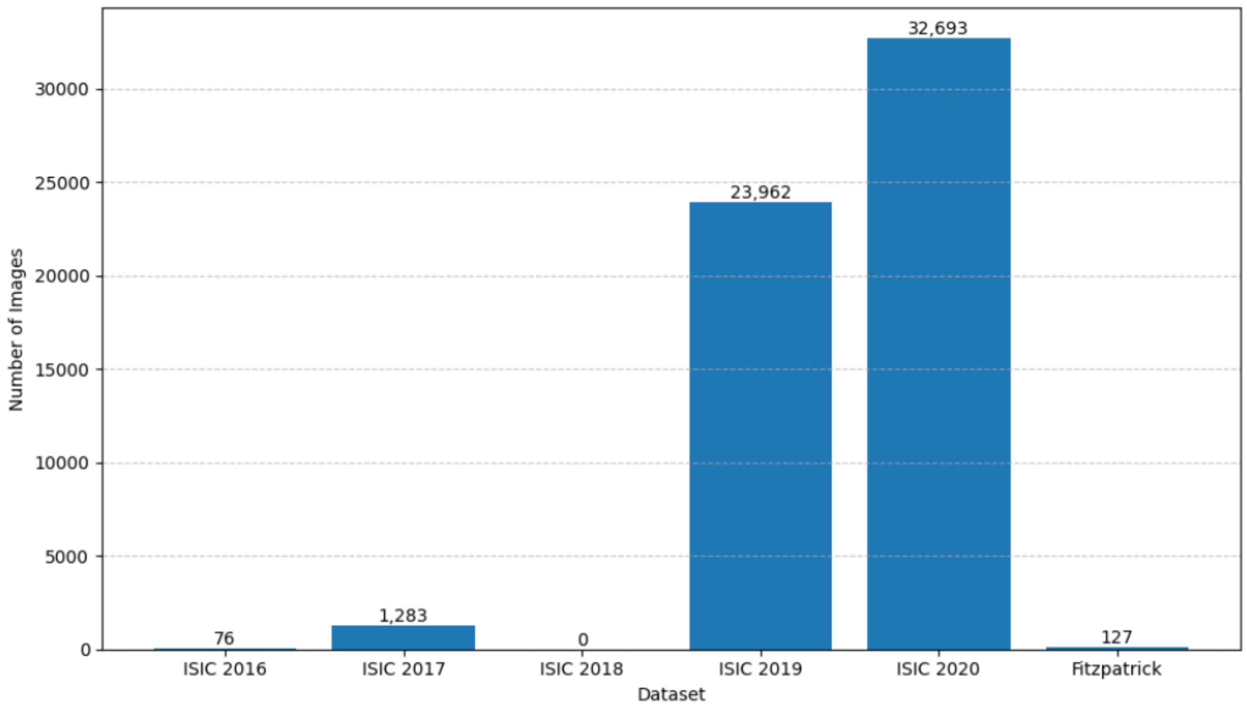
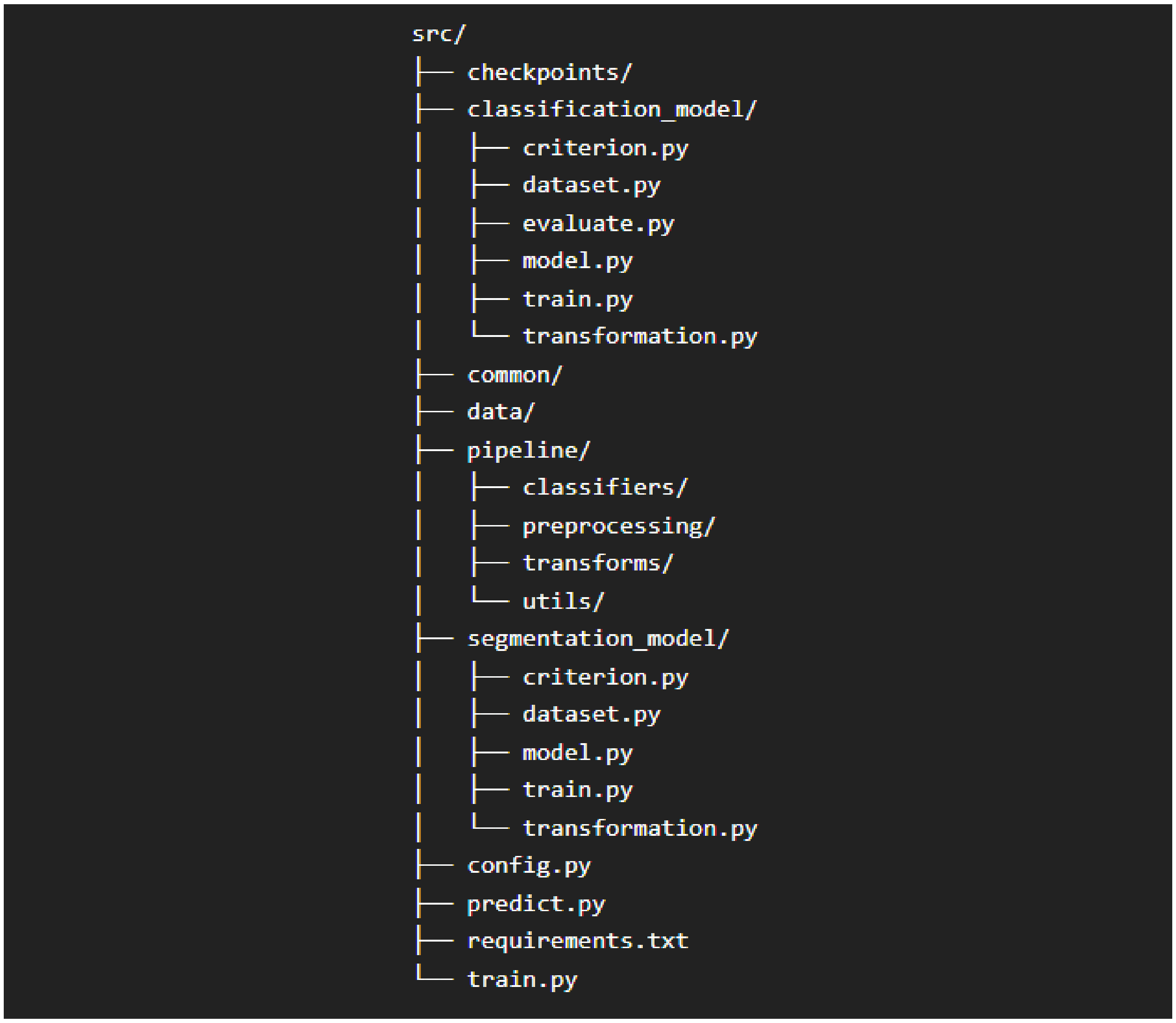


Figure 1. Number of images per dataset

2. Project Structure

The project is organized into modular components to ensure clear separation between models, data handling, preprocessing, and utilities:

- **checkpoints/**
  Stores trained model weights, including the segmentation and classification model checkpoints.
- **classification_model/**
  Contains code for the classification model, including dataset definitions, model architecture, transformations, loss functions (criteria), evaluation and training logic.
- **common/**
  Shared utilities, constants, data visualization tools, and metadata loading functions.
- **data/**
  Folder for the provided dermoscopic images and corresponding ISIC (2016-2020) and Fitzpatrick metadata files.
- **pipeline/**
  Preprocessing and auxiliary components:
  - **classifiers/**: Skin tone classification logic based on image brightness.
  - **preprocessing/**: Contains modules for preparing image data for model input. Includes classification_preprocessor.py (handles skin tone classification, lesion cropping, resizing/padding for the classifier) and segmentation_preprocessor.py (handles resizing/padding for images and masks for the segmentation model).
  - **transforms/**: Hair removal and lesion cropping modules.
  - **utils/**: Contains helper functions for tasks supporting the main pipeline, such as data sampling, balancing and parallel processing
- **segmentation_model/**
  Contains code for the segmentation model, including dataset definitions, model architecture, transformations, loss functions (criteria), and training logic.
- **scripts at root level:**
  - **config.py**: File containing all the configurations
  - **train.py**: Entry point for training the segmentation and classification models.
  - **predict.py**: Script for running inference on new data.
  - **requirements.txt**: List of required Python packages.
  - **duplicate_images.txt**: List of known duplicate images excluded during metadata cleaning.



```
src/
├── checkpoints/
├── classification_model/
│   ├── criterion.py
│   ├── dataset.py
│   ├── evaluate.py
│   ├── model.py
│   ├── train.py
│   └── transformation.py
├── common/
├── data/
├── pipeline/
│   ├── classifiers/
│   ├── preprocessing/
│   ├── transforms/
│   └── utils/
├── segmentation_model/
│   ├── criterion.py
│   ├── dataset.py
│   ├── model.py
│   ├── train.py
│   └── transformation.py
├── config.py
├── predict.py
├── requirements.txt
└── train.py
```

Figure 2. Project structure

3. Pipeline Overview

Our melanoma classification system follows a multi-stage pipeline that combines lesion segmentation, skin tone estimation and malignancy classification. The full training and inference workflow is outlined below.

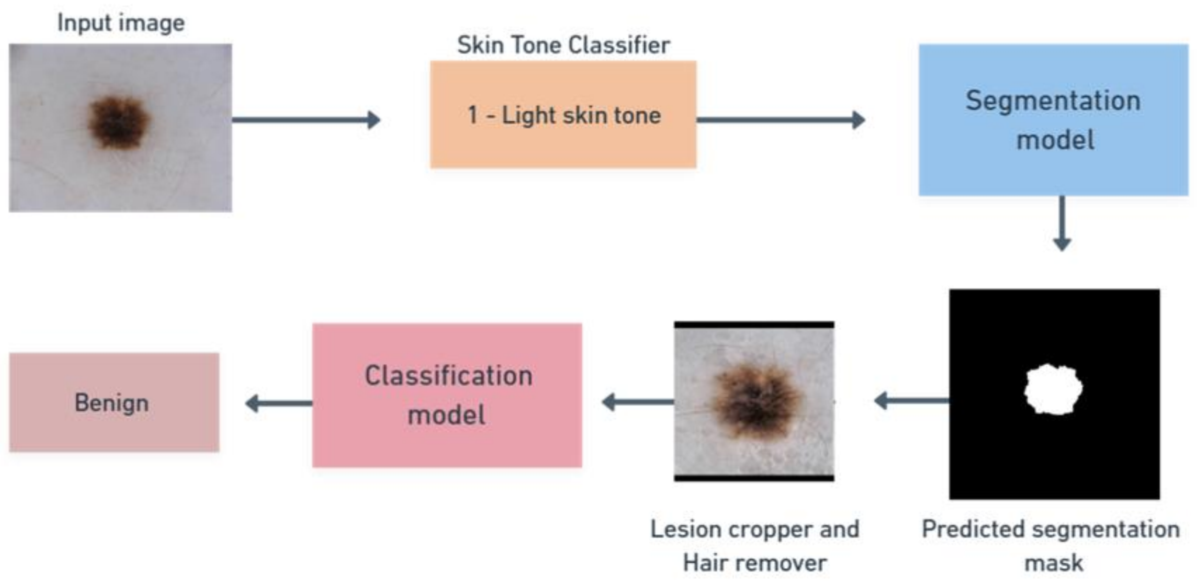


Figure 3. Lesion classification pipeline

3.1 Metadata Cleaning

Before starting the training processes, we preprocess the metadata by removing duplicate entries to prevent label noise, inconsistencies and to prevent training the model on duplicate images.

3.2 Segmentation Stage

1. **Data Preparation:**
   We begin by loading the provided segmentation masks and matching them with the corresponding original dermoscopic images based on their filenames. During this step, we normalize the image names by removing any suffixes such as _downsampled, ensuring consistent matching between images and masks. After filtering and sorting, we retain only those image-mask pairs where both components are available.

2. **Preprocessing pipeline**:
   Each image and its corresponding segmentation mask are resized and padded to a uniform square shape while preserving their aspect ratio. This ensures consistent input dimensions for the model. The processed images and masks are saved to disk, and metadata is updated accordingly.
3. **Augmentation:**
   With the goal of improving the robustness and generalization of the segmentation model, we apply a series of strong, custom-designed augmentations during training. Each transformation is applied simultaneously to both the input image and its associated mask to preserve spatial alignment. The augmentations include random horizontal and vertical flips, random rotations, affine transformations, brightness and contrast adjustments (color jitter), Gaussian blur, synthetic hair artifact generation to simulate occlusions, and local contrast enhancement using CLAHE. After augmentation, each image is resized and padded to a fixed square size while maintaining the original aspect ratio.
4. **Training:**
   We train a U-Net model with a **ResNet34** backbone pretrained on **ImageNet**. Initially, most encoder layers are frozen, allowing only the deeper layers and the decoder to adapt. As training progresses, additional encoder layers are gradually unfrozen to fine-tune the entire network.
   The model is optimized using the **AdamW** optimizer with a **ReduceLROnPlateau** learning rate scheduler that adjusts the learning rate dynamically based on validation loss trends. To address the class imbalance between lesion and background pixels, we use the **Tversky Loss** function. Early stopping is implemented to terminate training if validation performance does not improve over a sustained period. The best-performing model checkpoint, based on validation loss, is saved for use in subsequent classification tasks.

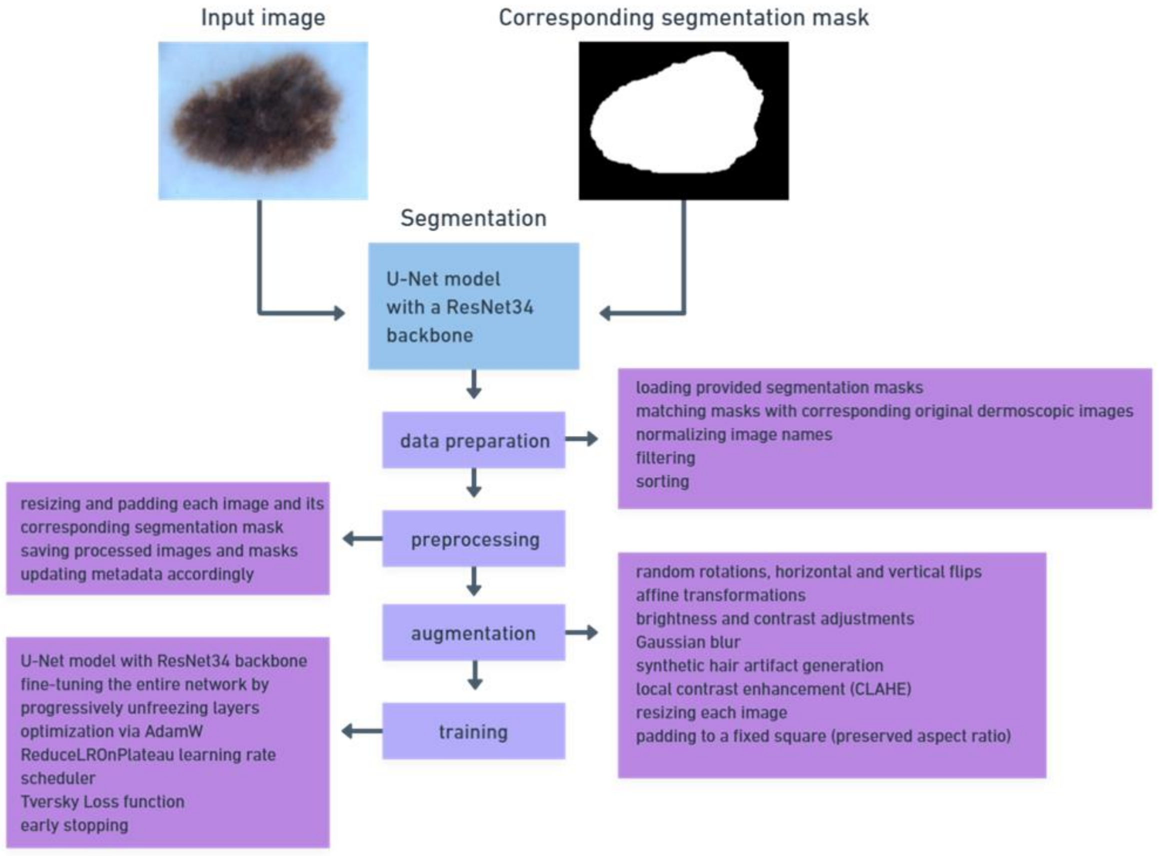

Input image
Corresponding segmentation mask
Segmentation
U-Net model with a ResNet34 backbone
data preparation
loading provided segmentation masks
matching masks with corresponding original dermoscopic images
normalizing image names
filtering
sorting
preprocessing
resizing and padding each image and its corresponding segmentation mask
saving processed images and masks
updating metadata accordingly
augmentation
random rotations, horizontal and vertical flips
affine transformations
brightness and contrast adjustments
Gaussian blur
synthetic hair artifact generation
local contrast enhancement (CLAHE)
resizing each image
padding to a fixed square (preserved aspect ratio)
training
U-Net model with ResNet34 backbone
fine-tuning the entire network by progressively unfreezing layers
optimization via AdamW
ReduceLROnPlateau learning rate scheduler
Tversky Loss function
early stopping

Figure 4. Segmentation model pipeline

3.3 Classification Stage

1. **Preprocessing Pipeline:**

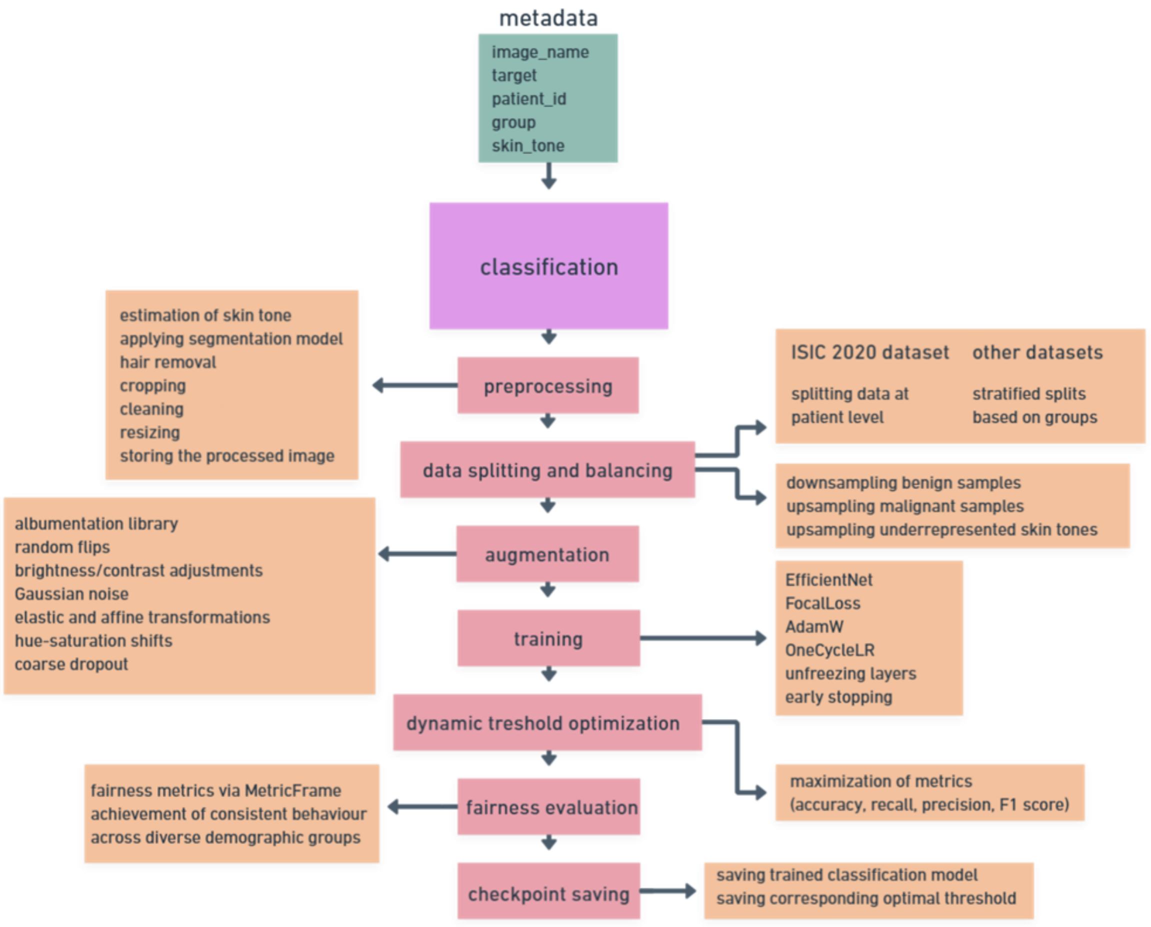
   For each image:

   a. Estimate skin tone by analyzing the brightness of the background skin area after masking the lesion.
   b. Apply the trained segmentation model to detect and crop the lesion area from the original image.
   c. Perform hair removal on the cropped lesion image to enhance feature clarity.
   d. Resize the cropped and cleaned lesion image to a fixed square size suitable for input to the classifier.
   e. Store the final preprocessed image to disk for faster subsequent data loading and to minimize redundant computation during training.

By caching the preprocessed images, we significantly reduced training time and improved overall training efficiency.

2. **Data Splitting and Balancing:**
    a. For the ISIC 2020 dataset, we split data at the patient level to avoid data leakage and to ensure that the images from the same patient are not split across training and validation sets.
    b. For other datasets, we perform stratified splits based on (target, skin_tone) groups, sampling more heavily from minority classes when necessary.
    c. To mitigate severe class imbalance, we down sampled benign samples and applied aggressive data augmentations. We also up sample images from underrepresented skin tones to promote fairness across demographic groups.
3. **Augmentation:**
    During training, we apply strong image augmentations using the Albumentations library. These include random flips, brightness/contrast adjustments, Gaussian noise, elastic transformations, hue-saturation shifts, affine transformations, and coarse dropout. Validation images are only resized and normalized.
4. **Training:**
    The classification model is based on an **EfficientNet**-like backbone, initialized with pretrained weights. The training procedure includes:
    a. Using **FocalLoss** to address class imbalance during optimization.
    b. Training with the **AdamW** optimizer and the **OneCycleLR** scheduler to efficiently adapt learning rates.
    c. Progressive unfreezing of model layers: starting with only the classifier head, then gradually unfreezing deeper layers over several epochs to fine-tune the entire model carefully.
    d. Early stopping based on validation loss trends to prevent overfitting.
5. **Dynamic Threshold Optimization:**
    Instead of applying a fixed probability threshold for classification, we dynamically compute the optimal threshold after each evaluation cycle to maximize combined metrics: accuracy, recall, precision, and F1 score.
6. **Fairness Evaluation:**
    Throughout validation, we monitor performance separately across different skin tone groups, using fairness metrics (via the MetricFrame framework). This helps ensure that the model achieves consistent behavior across diverse demographic groups.

7. **Checkpoint Saving:**
   Upon completion, the trained classification model and the corresponding optimal threshold are saved for inference and evaluation on the private test set.



metadata
image_name
target
patient_id
group
skin_tone
classification
preprocessing
estimation of skin tone
applying segmentation model
hair removal
cropping
cleaning
resizing
storing the processed image
data splitting and balancing
ISIC 2020 dataset
splitting data at patient level
other datasets
stratified splits based on groups
downsampling benign samples
upsampling malignant samples
upsampling underrepresented skin tones
augmentation
albumentation library
random flips
brightness/contrast adjustments
Gaussian noise
elastic and affine transformations
hue-saturation shifts
coarse dropout
training
EfficientNet
FocalLoss
AdamW
OneCycleLR
unfreezing layers
early stopping
dynamic treshold optimization
maximization of metrics (accuracy, recall, precision, F1 score)
fairness evaluation
fairness metrics via MetricFrame
achievement of consistent behaviour across diverse demographic groups
checkpoint saving
saving trained classification model
saving corresponding optimal threshold

Figure 5. Classification model pipeline

3.4 Dataset Statistics

Below we summarize the dataset sizes before and after preprocessing, balancing, and splitting as explained in section 3.3.

Set	Total samples	Malignant cases (%)	Benign cases (%)
Train (before)	35 864	22.10	77.90
Train (after)	46 178	37.86	62.14
Validation	8 974	22.52	77.48

Table 1. Dataset composition before and after balancing

The "Train (before)" row reflects the initial composition of the combined training data derived from the ISIC and Fitzpatrick datasets after duplicate removal and the initial train/validation splits, showing a significant class imbalance with only 22.10% malignant cases.

As shown in Table 1, our balancing strategy increased the effective training set size to 46,178 samples and significantly raised the proportion of malignant cases to 37.86%. This re-balancing combined with heavy augmentation aims to help the model learn more robust features for the minority class.

The validation set composition was unaffected and maintained a malignant case percentage (22.52%) like the original data distribution.

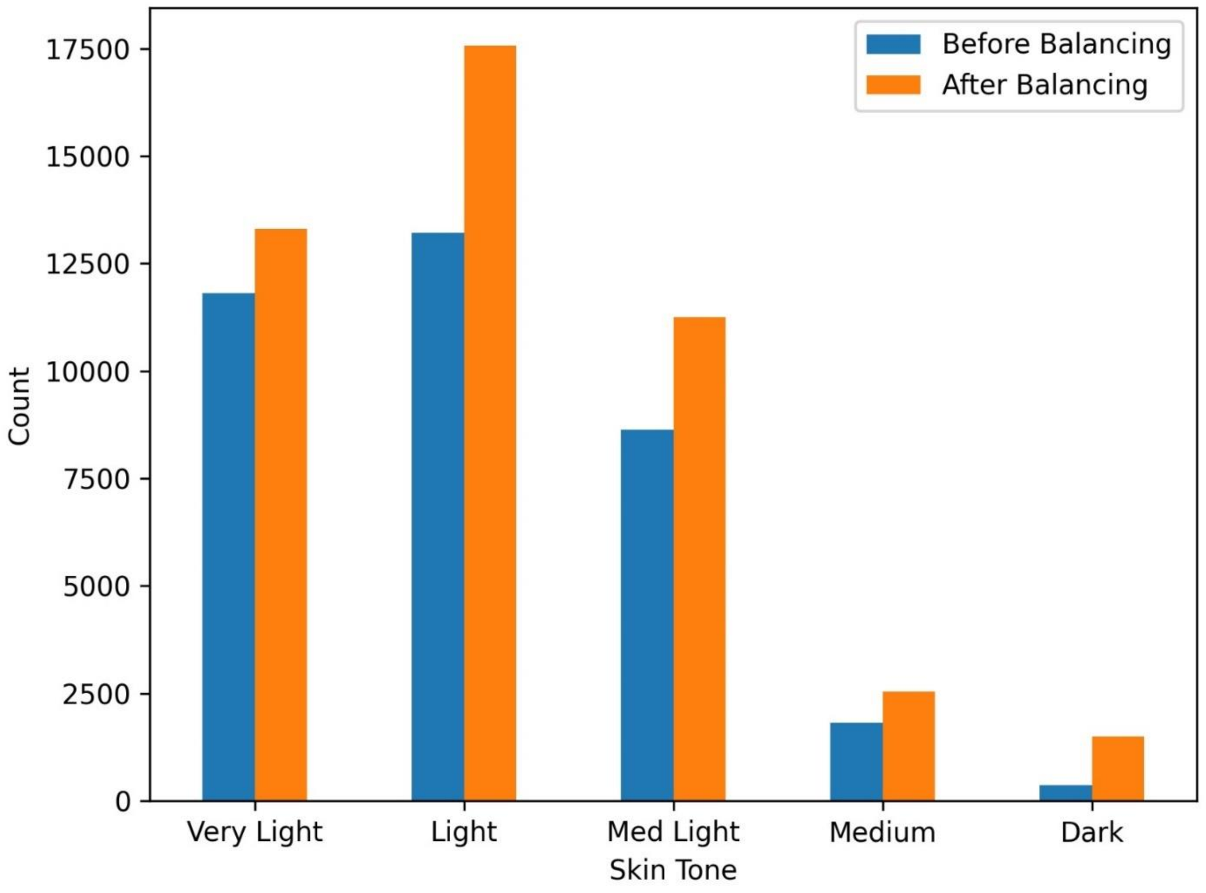


Figure 6. Skin tone distribution on the Training set before and after balancing

Since most malignant lesions were classified as lighter skin tones, our up sampling of malignant cases led to an increase in the number of training samples across all skin tone categories. However, consistent with our fairness objectives, the up sampling techniques resulted in a notably larger relative increase for the 'Medium' and 'Dark' skin tones compared to the 'Very Light', 'Light', and 'Med Light' categories. While still extremely imbalanced, this provided the model with significantly more exposure to the underrepresented darker skin tones during training.

4. Model Components and Design Decisions

4.1 Segmentation Model

Given that U-Net was proven to be highly effective in medical image segmentation tasks due to its ability to combine low-level and high-level features through skip connections, we used it for lesion segmentation with a **ResNet34** encoder pretrained on ImageNet.

To further improve segmentation focus, we integrated **Squeeze-and-Excitation (SCSE) attention blocks** into the decoder layers, enabling the model to better emphasize informative regions.

Additionally, **dropout regularization** was applied in the decoder to reduce overfitting, especially given the relatively small lesion regions compared to the full image size.

We chose **ResNet34** as the encoder backbone to balance between feature extraction capacity and training efficiency, ensuring good performance without excessive computational cost.

The **Tversky Loss function** was used to address the class imbalance between lesion and background pixels, giving more weight to false negatives, which are crucial for detecting melanoma.

Progressive unfreezing during training allowed us to first adapt only the higher layers of the encoder before fine-tuning the entire network. Combined with heavy data augmentation strategies, this training regime improved generalization on unseen dermoscopic images.

4.2 Classification Model

For classification, we employed an **EfficientNet-B4** backbone, chosen for its strong representational capacity, which is important for distinguishing subtle differences between benign and malignant lesions.

To further regularize the model and improve robustness, we applied:

- **Dropout** at the classifier level, and
- **Stochastic depth (DropPath)** within the **EfficientNet** backbone during training.

To handle the severe class imbalance in the dataset, we used **Focal Loss**, which makes the learning process focus on harder, misclassified examples.

Automatic Mixed Precision (AMP) was used throughout training to reduce memory usage and accelerate computations.

A progressive unfreezing schedule was adopted, starting with the classifier head and gradually involving deeper backbone layers to allow fine-tuning without catastrophic forgetting.

Finally, **dynamic threshold optimization** was performed after each validation phase to maximize key evaluation metrics such as F1 score, precision, and recall.

4.3 Lesion Cropper

To standardize lesion-centric inputs for classification, we developed a custom cropping module that leverages a pretrained segmentation model to detect and isolate lesion regions within dermoscopic images. The cropper operates by predicting a lesion mask, followed by morphological cleaning and component filtering to discard small artifacts.

To preserve important context while avoiding excessive background, we applied a dynamic cropping strategy: if the detected lesion covered an excessively large portion of the image or if no valid mask was found, the cropper defaulted to using the entire image. Otherwise, a bounding box with configurable margin was extracted around the lesion region, and the result was padded to a square aspect ratio and resized to a uniform target resolution.

Prior to inference, images were enhanced using CLAHE (Contrast Limited Adaptive Histogram Equalization) and gamma correction to improve lesion visibility under varied lighting conditions. Additional morphological dilation ensured that surrounding context was preserved without overcropping.

This modular approach ensures consistency in lesion presentation across the dataset while mitigating risks of overfitting to background artifacts or missing subtle lesion boundaries.

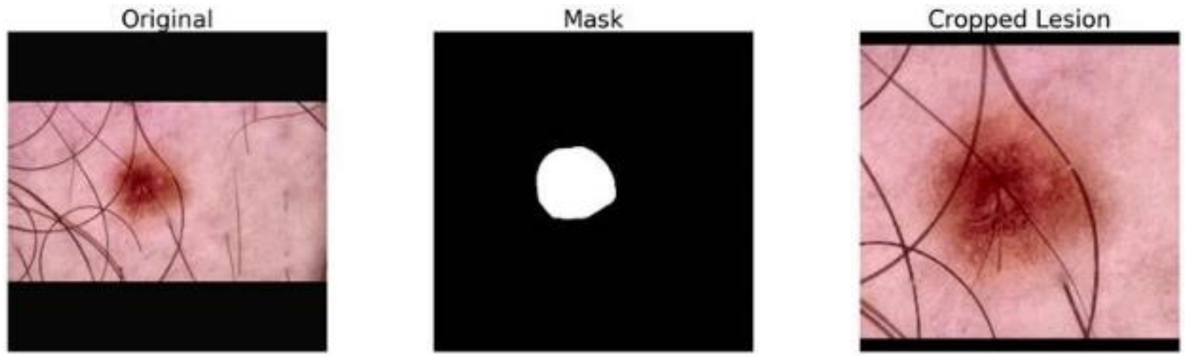


Figure 7. Cropped lesion (successful lesion recognition)

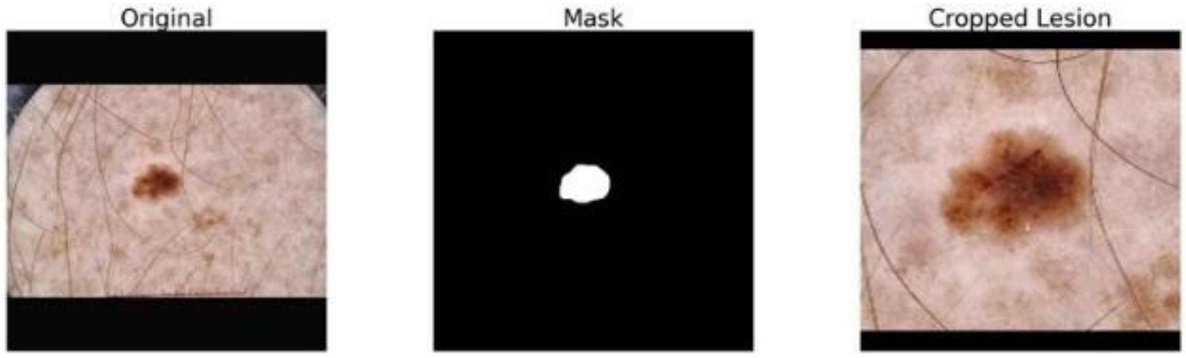


Figure 8. Cropped lesion (successful lesion recognition)

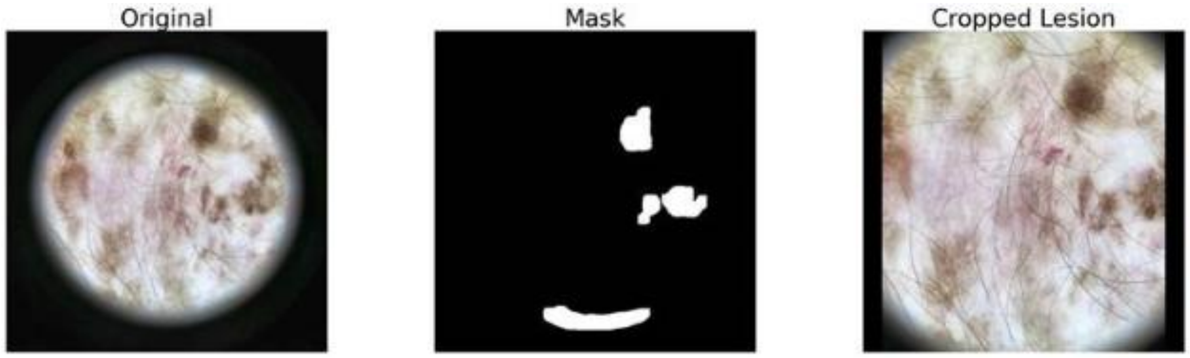


Figure 9. Cropped lesion (partial lesion recognition)

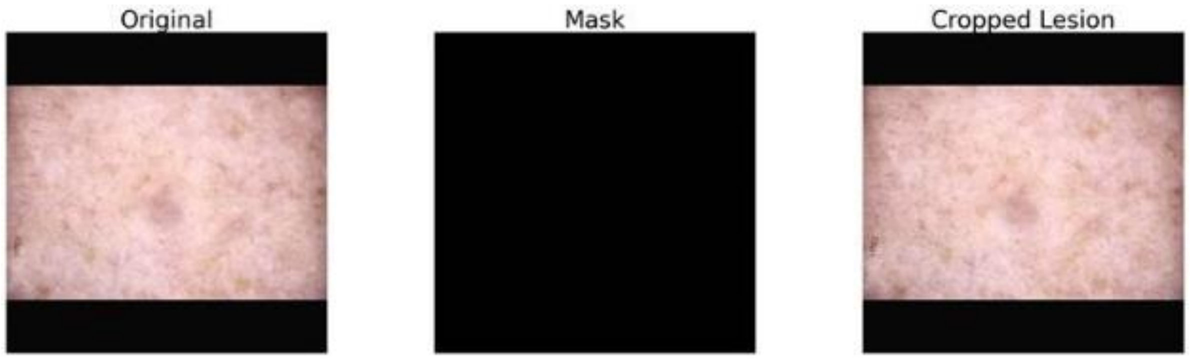


Figure 10. Cropped lesion (unsuccessful lesion recognition)

4.4 Skin Tone Classification

Rather than training a separate model, skin tone was estimated based on the average brightness of background skin areas after lesion segmentation. This lightweight, heuristic-based approach avoids introducing additional sources of bias and complexity while still providing a reliable signal for fairness evaluation.

Initially, we planned to categorize skin tone into just three broad groups: light, medium, and dark. However, during exploratory analysis, we discovered a strong imbalance – most of the dataset consisted of light-skinned samples, while medium and dark tones were significantly underrepresented. To better reflect this distribution and enable more granular fairness assessments, we decided to subdivide the "light" category into three finer-grained groups, resulting in five distinct classes overall: **very light**, **light**, **medium-light**, **medium**, and **dark**.

This adaptive binning based on median brightness allows us to better characterize skin tone variations present in the data and supports more detailed subgroup analysis during model evaluation.

4.5 Hair Removal Algorithm

Many dermoscopic images in the datasets contained hair artifacts which could obscure critical features of skin lesions and negatively impact classification performance.
To address this, we apply a dedicated hair removal algorithm as part of our preprocessing pipeline.

The method detects dark linear structures consistent with hair and removes them by inpainting the affected regions, effectively reconstructing the background underneath.
This step improves lesion visibility and ensures that the classifier focuses on meaningful skin patterns rather than irrelevant occlusions.

The original implementation can be found here:
https://www.kaggle.com/code/vatsalparsaniya/melanoma-hair-remove

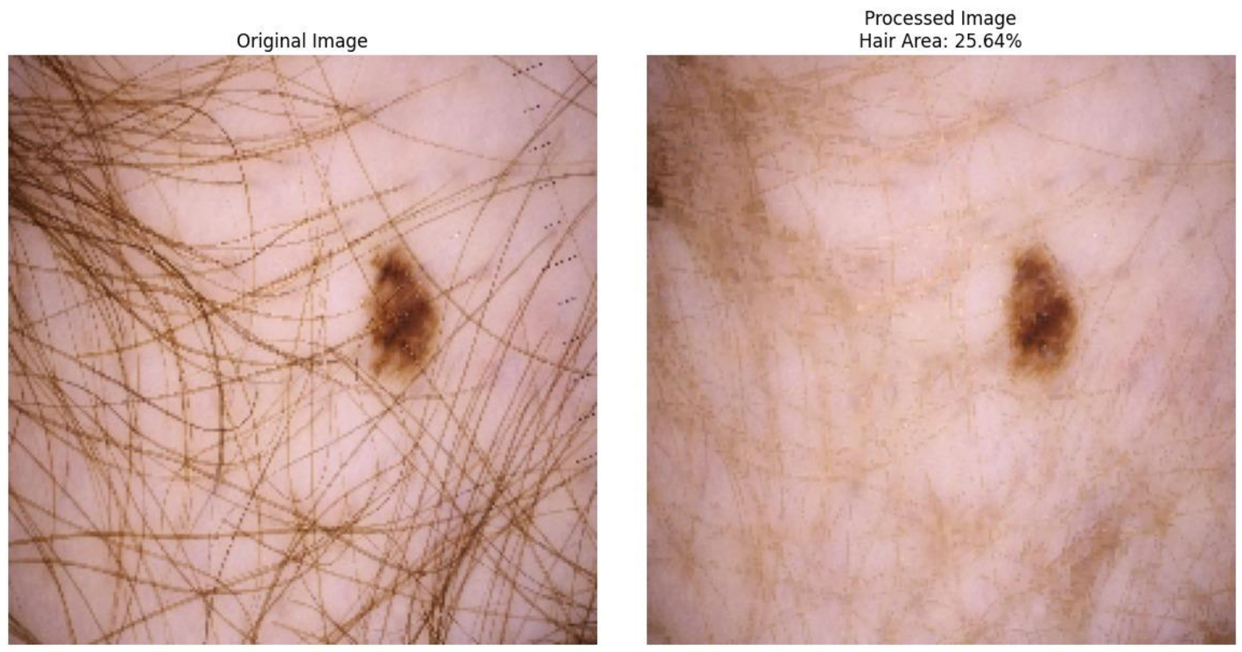
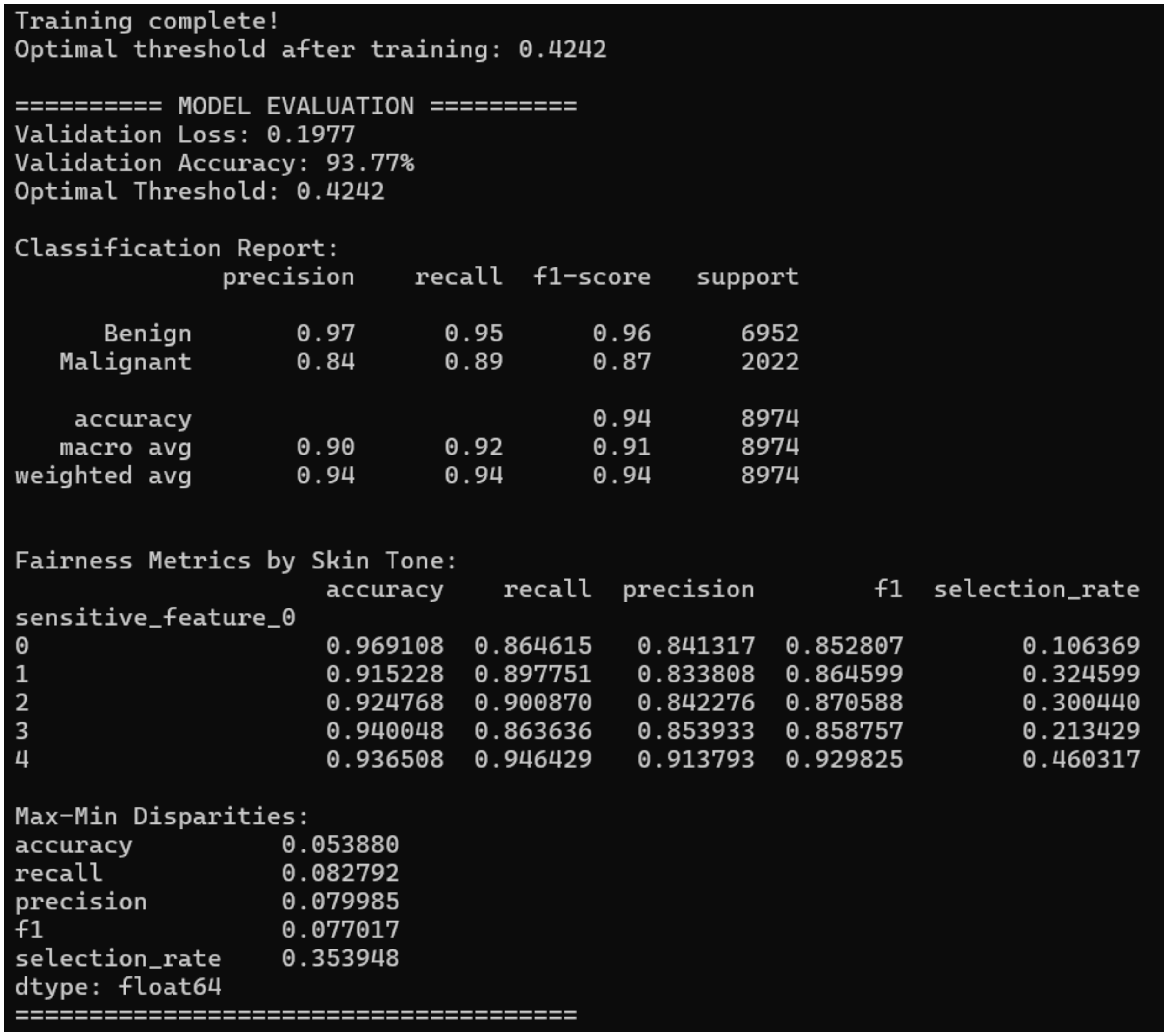


Figure 11. Hair removal

5. Results

We evaluated our final melanoma classification model on the internal validation set, with a focus on both overall predictive performance and fairness across skin tone groups.



```
Training complete!
Optimal threshold after training: 0.4242

========== MODEL EVALUATION ==========
Validation Loss: 0.1977
Validation Accuracy: 93.77%
Optimal Threshold: 0.4242

Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.95      0.96      6952
   Malignant       0.84      0.89      0.87      2022

    accuracy                           0.94      8974
   macro avg       0.90      0.92      0.91      8974
weighted avg       0.94      0.94      0.94      8974


Fairness Metrics by Skin Tone:
                    accuracy    recall  precision        f1  selection_rate
sensitive_feature_0
0                   0.969108  0.864615   0.841317  0.852807        0.106369
1                   0.915228  0.897751   0.833808  0.864599        0.324599
2                   0.924768  0.900870   0.842276  0.870588        0.300440
3                   0.940048  0.863636   0.853933  0.858757        0.213429
4                   0.936508  0.946429   0.913793  0.929825        0.460317

Max-Min Disparities:
accuracy          0.053880
recall            0.082792
precision         0.079985
f1                0.077017
selection_rate    0.353948
dtype: float64
======================================
```

Figure 12. Model evaluation after training

5.1 Overall Performance

Our model achieved a validation accuracy of approximately 94**%** and a weighted average F1 score of around **94%**.

It demonstrated very high precision when identifying benign lesions and strong recall on malignant cases, reflecting good sensitivity, which is particularly important for a melanoma detection system.

However, despite the strong overall performance, there were challenges in maintaining a perfect balance between precision and recall across all categories. Precision for malignant cases was noticeably lower than for benign cases, indicating that some false positives occurred.

5.2 Fairness Analysis

The model's fairness was evaluated across five defined skin tone categories:

- 0 - Very Light Skin Tone
- 1 - Light Skin Tone
- 2 - Medium Light Skin Tone
- 3 - Medium Skin Tone
- 4 - Dark Skin Tone

Although overall model performance remained strong across groups, earlier evaluations showed **notably lower precision and higher selection rates** for **category 4 (Dark Skin Tone) -** highlighting a potential fairness concern.

To address this, we incorporated additional samples from the **Fitzpatrick 17k dataset**, specifically targeting underrepresented darker skin tones. This augmentation led to **marked improvements in performance for category 4**, bringing its precision, recall, and f1-score closer in line with other groups. The updated evaluation shows:

- **Precision and recall** for dark skin tones improved, reducing the gap with other skin tone categories.
- **Selection rate disparity**, while still the highest among metrics (~35%), was somewhat reduced after dataset balancing. This is largely due to inherent label imbalance. Category 4 has a much higher proportion of malignant cases (44%) compared to other skin tones

(11% category 0, 29% category 1, 27% category 2, 20% category 3), which skews the model’s selection behavior.

- **Maximum disparities** across groups now stand at:
  - **Accuracy:** 5.4%
  - **Recall:** 8.3%
  - **Precision:** 8.0%
  - **F1 Score:** 7.7%
  - **Selection Rate:** 35.4%

These results suggest that **adding targeted data for underrepresented groups** can be an effective strategy to mitigate fairness gaps – especially when large imbalances exist in the original dataset. The high selection rate disparity largely mirrors the true malignancy distribution across skin tones, though threshold calibration may still improve fairness perceptions.

5.3 Summary

The model demonstrated strong overall performance in melanoma detection, with balanced sensitivity and specificity. However, fairness analysis revealed persistent **disparities** – particularly in **precision** and **selection rates** between very light and dark skin tones. These gaps are mostly due to underlying differences in malignancy prevalence across groups. Future work should focus on curating datasets with more balanced malignancy rates across skin tones, alongside augmentation and fairness-aware training, to improve equity without sacrificing diagnostic accuracy.

6. Conclusion

In this project, we developed a two-stage pipeline for melanoma detection, combining lesion segmentation, skin tone estimation, and malignancy classification. Our solution achieved strong overall accuracy and demonstrated good sensitivity for malignant cases, while also striving for fairness across diverse skin tone groups.

Through extensive data preprocessing, augmentation, and progressive fine-tuning, we built a robust system capable of generalizing to varied dermoscopic conditions. Fairness evaluations, however, revealed disparities, particularly in precision and selection rates for darker skin tones, that largely mirror malignancy by skin tone imbalances present in the dataset.

Future work should focus on curating datasets with more consistent malignancy distributions across skin tones, in addition to exploring fairness-aware training and targeted augmentation, to reduce disparity without sacrificing diagnostic performance.

7. Instructions for Reproducing the Results

In this section we will describe how to prepare the environment, organize the datasets, and run the training and prediction pipelines to reproduce the results presented.

7.1 Environment Setup

This section details the hardware and software environment used for developing, training, and evaluating the melanoma detection model.

Hardware:

- CPU: AMD Ryzen Threadripper 3990X 64-Core Processor
- GPU: 2 x NVIDIA GeForce RTX 3090 (24GB GDDR6X VRAM each), we ran one script on each GPU for faster iteration
- System Memory (RAM): 256 GB

Software:

- Operating System: Ubuntu 22.04.4 LTS
- NVIDIA Driver version 550.120
- CUDA version 12.4
- Python v3.11.7

Setup Instructions:

- Linux:
    - Create a Python virtual environment: **python3 -m venv venv**
    - Activate the environment: **source venv/bin/activate**
    - Install dependencies: **pip install -r requirements.txt**
- Windows:
    - Create a Python virtual environment: **python -m venv venv**
    - Activate the environment: **venv\Scripts\activate**
    - Install dependencies: **pip install -r requirements.txt**

7.2 Preparing the Data

First, download and organize the datasets as follows:

7.2.1 Downloading the Dermoscopic Images

Download the ISIC training images and store them in the corresponding folders:

- data/2016_train <- 2016 Images
- data/2017_train <- 2017 Images
- data/2019_train <- 2019 Images
- data/2020_train <- 2020 Images

Note: we don't need to download the 2018 dataset because it is completely removed in the deduplication process (it consists entirely of duplicates that can be found in the other ISIC datasets). The 127 Fitzpatrick17k images will automatically download when running the program.

7.2.2 Downloading the Metadata

Download the ISIC and Fitzpatrick metadata files and store them directly inside the data/ directory.
Make sure to rename the metadata files accordingly:

- data/2016_Metadata.csv <- 2016 Metadata
- data/2017_Metadata.csv <- 2017 Metadata
- data/2018_Metadata.csv <- 2018 Metadata
- data/2019_Metadata.csv <- 2019 Metadata
- data/2020_Metadata.csv <- 2020 Metadata
- data/Fitzpatrick_Metadata.csv <- Fitzpatrick Metadata

7.2.3 Downloading the Segmentation Masks

Download the available segmentation masks from the ISIC datasets (where provided) and store them under:

- data/masks <- 2017 Masks
- data/masks <- 2018 Masks

This folder should contain masks from ISIC 2017 and 2018 datasets.

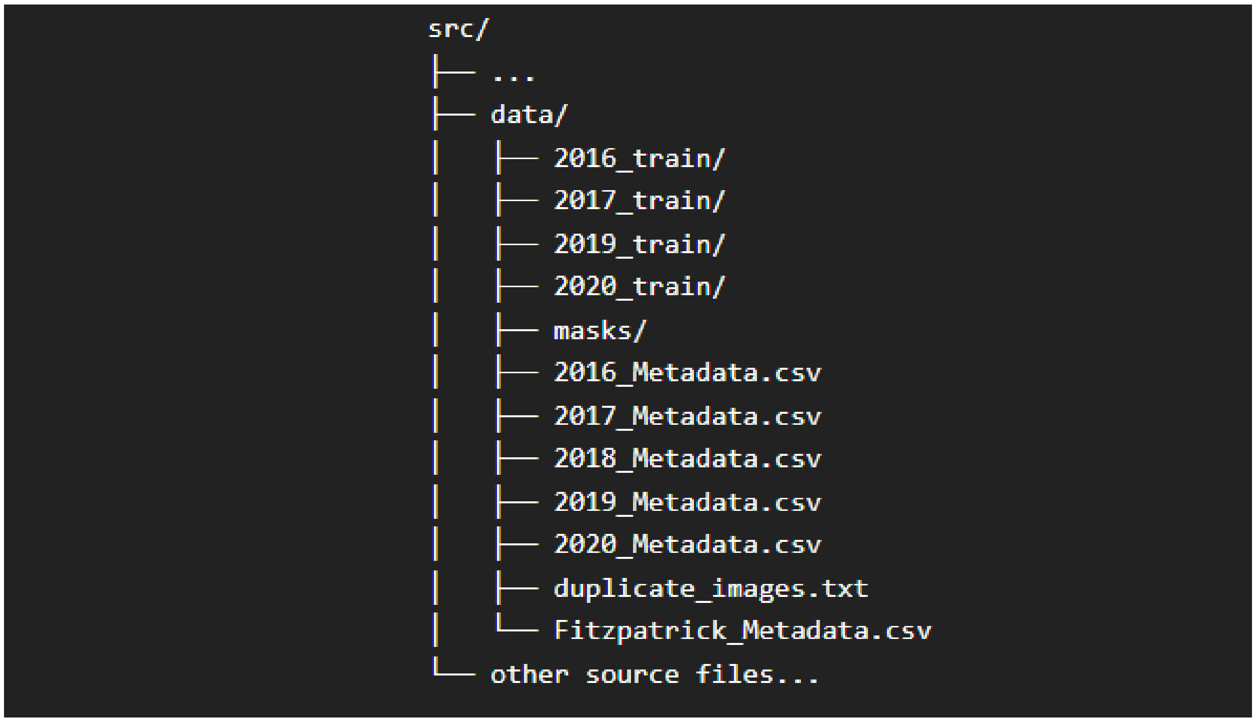
Note: There will be around 1800 images with duplicate names. When copying over, choose "skip these files" - they are just duplicates.

7.2.4 Downloading the Duplicate Images List

Download the text file listing duplicate images and place it inside the data folder. This file is used to remove duplicate entries during data preparation, make sure the names are correct:

- data/duplicate_images.txt <- Duplicate images

After completing these steps, your folder structure should look like this:



```
src/
├── ...
├── data/
│   ├── 2016_train/
│   ├── 2017_train/
│   ├── 2019_train/
│   ├── 2020_train/
│   ├── masks/
│   ├── 2016_Metadata.csv
│   ├── 2017_Metadata.csv
│   ├── 2018_Metadata.csv
│   ├── 2019_Metadata.csv
│   ├── 2020_Metadata.csv
│   ├── duplicate_images.txt
│   └── Fitzpatrick_Metadata.csv
└── other source files...
```

Figure 13. Data folder structure

If you wish to use different paths for the datasets, masks, or outputs, you can update the paths inside the configuration file: **config.py**

7.3 Training the Model

To start the full training pipeline (segmentation model + classification model), simply run:

```
python train.py
```

This will:

- Train the segmentation model on the dermoscopic images using the provided masks
- Train the melanoma classification model on the preprocessed and cropped lesion images

The resulting model checkpoints will be saved under the checkpoints/ directory.

We got our results using:

Parameter	Classification model	Segmentation model
Image size	512x512	512x512
Batch size	32	16
Number of epochs	30	18

Table 2. Training Parameters for Classification and Segmentation Models

These parameters can be tweaked in **config.py**

7.4 Making Predictions

To generate predictions on a new dataset using a trained model checkpoint, run:

```
python predict.py [INPUT_DATA_DIRECTORY] [OUTPUT_FILE]
```

where:

- [INPUT_DATA_DIRECTORY] is the path to the directory containing input images.
- [OUTPUT_FILE] is the desired path for storing the model's predictions (e.g., a CSV file).

Example:

```
python predict.py data/test/ predictions.csv
```

8. Sources

2016 Images - https://isic-challenge-data.s3.amazonaws.com/2016/ISBI2016_ISIC_Part3_Training_Data.zip

2017 Images - https://isic-challenge-data.s3.amazonaws.com/2017/ISIC-2017_Training_Data.zip

2019 Images - https://isic-challenge-data.s3.amazonaws.com/2019/ISIC_2019_Training_Input.zip

2020 Images - https://isic-challenge-data.s3.amazonaws.com/2020/ISIC_2020_Training_JPEG.zip

2017 Masks - https://isic-challenge-data.s3.amazonaws.com/2017/ISIC-2017_Training_Part1_GroundTruth.zip

2018 Masks - https://isic-challenge-data.s3.amazonaws.com/2018/ISIC2018_Task1_Training_GroundTruth.zip

2016 Metadata - https://isic-challenge-data.s3.amazonaws.com/2016/ISBI2016_ISIC_Part3_Training_GroundTruth.csv

2017 Metadata - https://isic-challenge-data.s3.amazonaws.com/2017/ISIC-2017_Training_Part3_GroundTruth.csv

2018 Metadata - https://isic-challenge-data.s3.amazonaws.com/2018/ISIC2018_Task3_Training_GroundTruth.zip

2019 Metadata - https://isic-challenge-data.s3.amazonaws.com/2019/ISIC_2019_Training_GroundTruth.csv

2020 Metadata - https://isic-challenge-data.s3.amazonaws.com/2020/ISIC_2020_Training_GroundTruth.csv

Duplicate images - https://github.com/mmu-dermatology-research/isic_duplicate_removal_strategy/blob/main/file_lists/06%20-%20all_train_duplicates_deleted_(all%20but%20newest).txt

Fitzpatrick Metadata - https://github.com/mattgroh/fitzpatrick17k/blob/main/fitzpatrick17k.csv