Individual Report: Classification of Skin Lesion Melanoma Detection

A. Introduction

Skin cancer, particularly melanoma, can be life-threatening if not detected early. However, early diagnosis significantly improves the chances of successful treatment. To ensure accurate diagnosis, physicians typically use a combination of clinical photographs, dermoscopic images, and patient history.

Dermoscopy has emerged as a more reliable imaging technique for skin cancer detection. It enhances visualization by producing magnified, illuminated images of the skin while eliminating surface reflections. Despite its advantages, diagnosing melanoma through dermoscopic image inspection remains challenging; it is time-consuming, prone to inconsistencies, and subject to human error.

B. Dataset

The dataset for ISIC 2019 contains 25,331 images available for the classification of dermoscopic images among nine different diagnostic categories:

Lesion Category	Class Name
Melanoma	MEL
Melanocytic nevus	NV
Basal cell carcinoma	BCC
Actinic keratosis	AK
Benign keratosis	BKL
Dermatofibroma	DF
Vascular lesion	VASC
Squamous cell carcinoma	SCC
None of the above	UNK

C. Dataset Preparation

The data loading process starts by reading the metadata CSV file, selecting the relevant disease labels, and converting them into PyTorch tensors. These labels are then saved in both .npy and .pt formats, making them easy to load and use during model training.

D. Preprocessing

1. Cropping Image

Let the input image be represented as a 3D tensor:

$$I \in H \times W \times C$$

where:

- H = image height
- W = image width
- $C \in \{1,3\}$ = number of channels (grayscale or RGB)

The function *'def crop_image_from_gray(img, tol=7)'* removes unnecessary grey or black borders from the image, areas that contain low-intensity pixels and don't contribute meaningful information. Using a brightness threshold (tol = 7), a binary mask is created to identify the region that contains relevant content (like a skin lesion). The image is then cropped to focus only on this region, reducing noise and helping the model focus on the lesion itself.

2. CLAHE (Contrast Limited Adaptive Histogram Equalization)

The clahe_lab function enhances the image by focusing on the L channel (lightness) from the LAB colour space, which separates lightness from colour.

Here's how it works step by step:

Color Space Conversion: The input image is first converted from BGR (the default OpenCV format) to LAB color space.

- In LAB, the L channel represents brightness.
- The A and B channels represent color:
    - A shifts from green to red
    - B shifts from blue to yellow

Channel Separation: The LAB image is then split into three separate channels:

- L (lightness)
- A (green–red)
- B (blue–yellow)

CLAHE on L channel: CLAHE (Contrast Limited Adaptive Histogram Equalisation) is applied only to the L channel to enhance contrast in the brightness of the image, making details more visible without over-amplifying noise.

Merge Channels: After enhancing the L channel, it's combined back with the untouched A and B channels.

Final Conversion: The image is converted back from LAB to RGB so it can be displayed normally.

3. Ben Graham Preprocessing

Ben Graham's method enhances image contrast by dividing the original image by its blurred version:

Enhanced Image=Original \ Blurred Version

This technique is especially effective in medical imaging, where it is critical to highlight fine details such as lesions. By performing this division, the method reduces the influence of uneven lighting and dark backgrounds, making the central object, such as a skin lesion, appear sharper and more prominent.

The equation shown visually represents this process, where:

- Original refers to the raw input image containing both details and broader lighting patterns.
- Blurred Version is created by applying a Gaussian or other low-pass blur to remove fine detail and retain only broad intensity trends.

The Enhanced Image results from dividing the original by the blurred version. This highlights local contrast and edges. In regions where the original image has small-scale texture or edge information, the blurred version remains smooth. Dividing the two increases the relative intensity, emphasizing those features. In flatter areas where both images are similar, the division yields a value near 1, so those regions remain less emphasized.

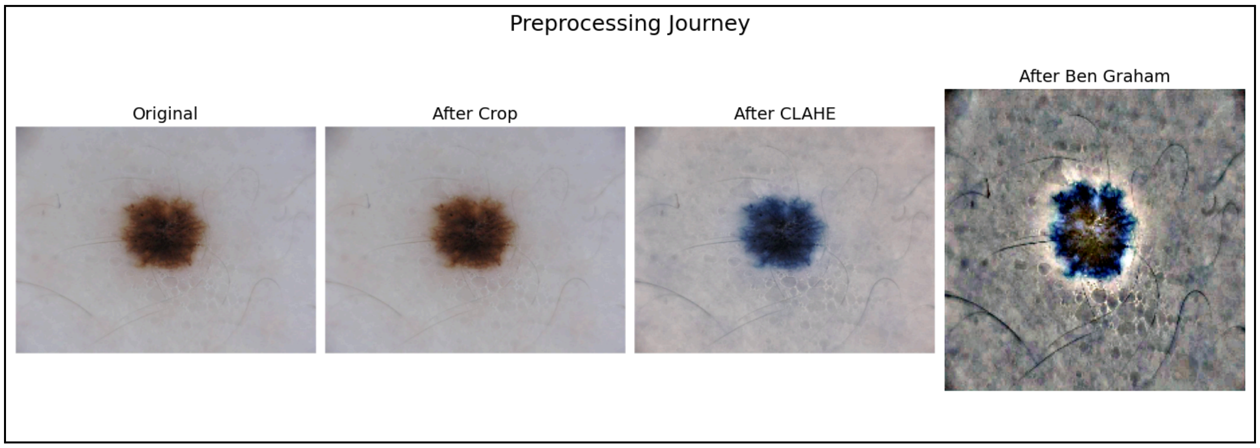


Fig: Preprocessing pipeline showcasing cropping, CLAHE, and Ben Graham enhancement

E. Skin Lesion Classification Using ResNet-18 and Custom Head

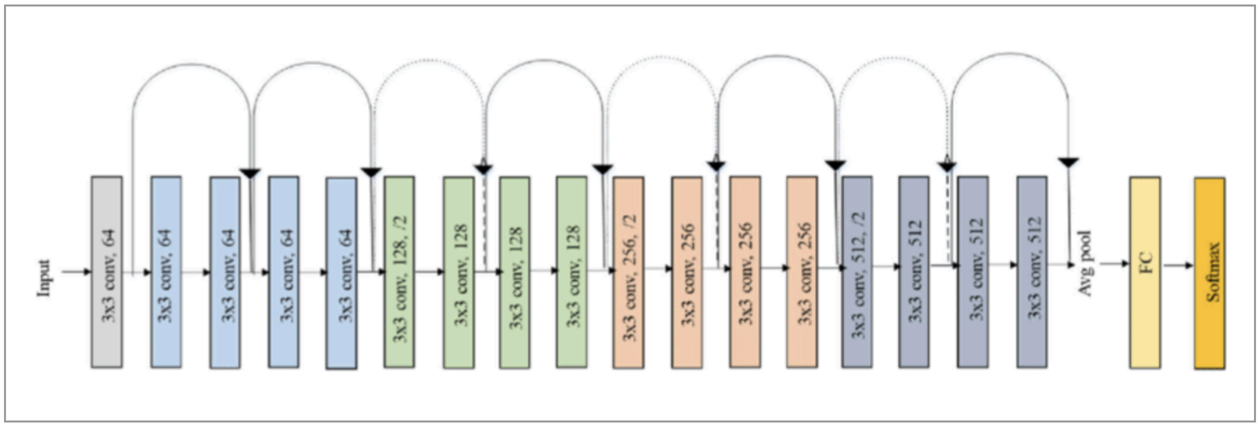


Fig: General reference of the Resnet18 model

Model Overview

ResNet-18, a convolutional neural network pretrained on the ImageNet dataset, is used as a fixed feature extractor. The input image, $p=224\times224\times3$, passes through the convolutional layers up to the global average pooling stage. This transforms the input into a 512-dimensional feature vector:

These layers include a 7×7 convolution, batch normalization, ReLU activation, max pooling, and four residual blocks. This setup enables the extraction of hierarchical visual features such as edges, textures, and high-level patterns.

The original fully connected (FC) layer of ResNet-18, which is designed for 1000-class ImageNet classification, is removed by replacing it with nn.Identity(). This modification allows retention of pretrained feature extraction while discarding the task-specific classification head.

Custom Classification Head

1. Linear Transformation (512 → 256): reduces the dimensionality of extracted features

$$n(1) = W(1) * a(0) + b(1), \text{ with } W(1) \in R(256\times512),\ b(1) \in R(256)$$

2. Activation Function:

$$a(1)=f(1)(n(1))=\mathrm{ReLU}(n(1))$$

3. Dropout (during training only):

$$a(1)=\mathrm{Dropout}(a(1),\ p=0.5)$$

4. Linear Transformation (256 → 9):

$$n(2)=W(2) * a(1)+b(2) \text{ with } W(2) \in R9\times256,\ b(2) \in R9$$

5. Sigmoid Activation (for multi-label probability):

$$a(2)=f(2)(n(2))=\sigma(n(2))$$

where the sigmoid function is applied element-wise: $\sigma(x)=1/1+e-x$

Final Output

This architecture supports multi-label classification, where an image may belong to multiple classes simultaneously.

The final output is computed as:

$$\text{Output} = p.W^T + b$$

Where:

- p is the input tensor with shape [B, 256]
- W is the weight matrix with shape [9, 256]
- b is the bias vector with shape [9]
- The resulting output shape is [B, 9], representing the probability of each class per image

This configuration facilitates efficient adaptation of a pretrained model to domain-specific tasks such as skin lesion analysis.

The final network output $a(2)\in[0,1]9$ represents the predicted probability of each of the 9 skin lesion classes. The full forward pass can be expressed as:

$$a(2)=\sigma(W(2)\cdot \text{Dropout}(\text{ReLU}(W(1)a(0)+b(1)))+b(2))$$

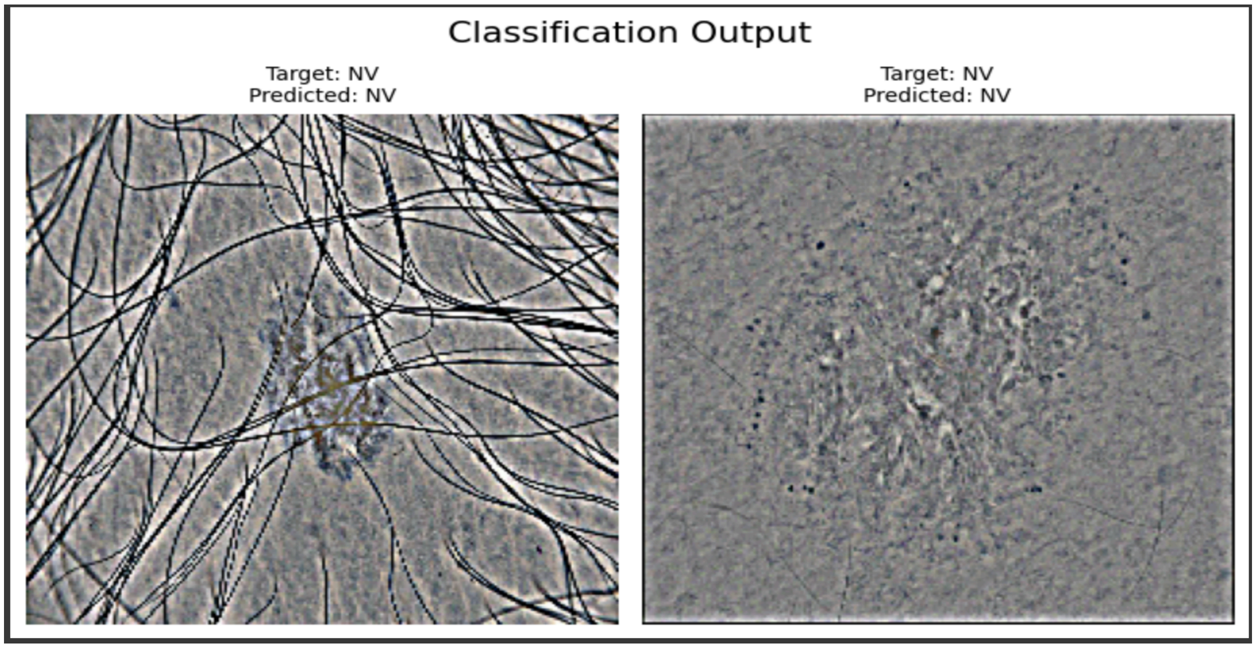


Fig: Classification Output (Target Vs Predicted) image belonging to the Melanocytic nevus(NV) lesion category

F. Training and evaluation

The training was conducted over 6 epochs with a batch size of 64. The loss function used was Binary Cross-Entropy Loss (BCELoss) as the task was multi-label, i.e., an image could belong to more than one class. The Adam optimizer was used with its adaptive learning and an initial learning rate of 1e-4.

To prevent overfitting and optimize learning, we employed a ReduceLROnPlateau scheduler that reduces the learning rate when validation accuracy plateaus. During the training process, real-time metrics were tracked using a custom MetricTracker utility, which displayed training and validation losses to visually assess model performance.

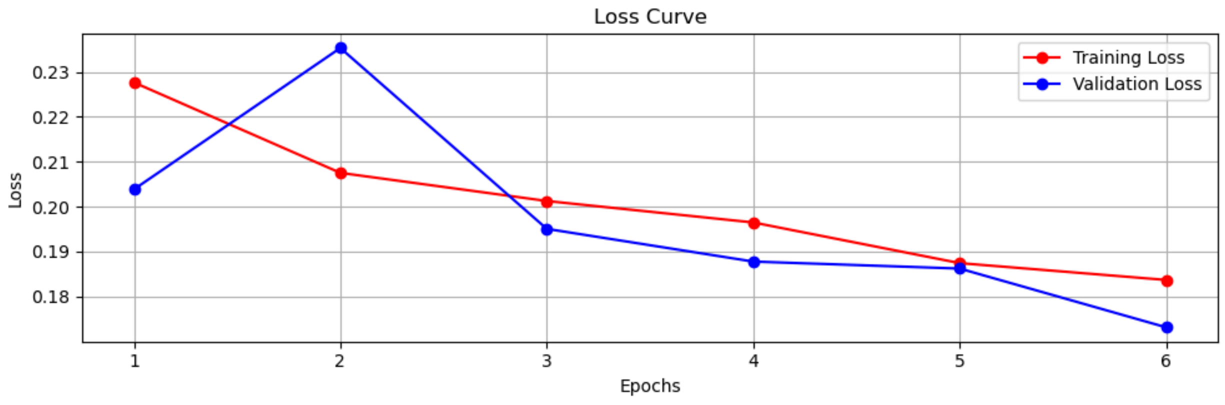
The model's evaluation phase was done on the validation split, and thresholded sigmoid outputs were utilized to establish predicted class membership. The prediction is positive for a class if its probability exceeds 0.5 (or a tuned threshold). The final evaluation was based on validation accuracy, which calculated the ratio of correctly predicted labels to a total of nine classes.

Validation Accuracy = (Correct Predictions / Total Predictions) × 100

The model achieved strong training loss convergence, and the classification outputs were visualized to verify that the model learned to distinguish lesion types accurately.

G. Evaluation Metrics

1.Loss Curve



Training Loss (Red Curve):
The training loss shows a consistent **downward trend**, dropping from approximately 0.23 to 0.182.This indicates that the model is steadily learning from the training data and improving its ability to minimize error during backpropagation.A smooth decline without spikes suggests that the optimizer is stable and not overfitting to noisy samples.

Validation Loss (Blue Curve):
The validation loss initially **increases** from ~0.20 to ~0.235 at epoch 2, before dropping consistently to ~0.177 by epoch 6.This early increase may indicate the model was adjusting to generalizing beyond the training data.The subsequent decline suggests that the model successfully **generalized better after the initial adjustment**, which is a good sign.

2. Precision/ Recall/AUC:

Accuracy: 0.5540
This means the model correctly predicted all the labels for an image (i.e., exact match across all 9 classes) 55.4% of the time. In multi-label settings, this is a strict metric, as it requires the model to get all labels correct for a sample.

Precision: 0.2880
Precision measures how many of the labels predicted as "positive" by the model were actually correct. A precision of 28.8% indicates that many of the predicted positive labels were false positives — the model is overpredicting some classes.

Recall: 0.1993
Recall reflects how many of the actual positive labels the model successfully identified. A recall of 19.93% shows that the model missed a significant portion of true positive labels — it's under detecting certain lesion types.

H. Conclusion

Here, we could deploy a pipeline of deep learning for skin lesion image classification with an adapted ResNet18 model. We enhanced the quality of the input image with cropping, CLAHE, and Ben Graham contrast. The model was strengthened by augmentation techniques such as flipping, rotation, and colour jitter.

Our model successfully applied transfer learning to transform a general ResNet model to a domain-adapted task. While there was promising convergence in training loss, the validation behaviour indicates that more examination into class imbalance or data leakage possibility is warranted.

The project illustrates the capability of deep learning for dermatologic image classification and the need for preprocessing and careful model adaptation for specialized medical imaging tasks.

I. Future Scope

To further develop on this work, we recommend the following paths for future development:

1. Model Enhancements: Attempt to use deeper models such as ResNet50 or EfficientNet to improve feature extraction and accuracy of classification.

2. Treatment of Class Imbalance: Apply class reweighting, oversampling, or focal loss to handle class imbalance in the data.

3. Ensemble Learning: Combine numerous models to reduce variance and enhance predictive performance.

4. Explainability: Add Grad-CAM or other explainable AI systems to show what parts of the lesion the model focuses on in classification.

5. Mobile Deployment: Port the trained model into an optimized form (e.g., ONNX or TensorFlow Lite) for real-time mobile-based skin analysis apps.

6. Larger Dataset and Cross-Dataset Validation: Train on larger variations of the ISIC dataset and cross-validate on external datasets to cross-validate model generalizability.

These advances can progress the system closer to real-world clinical usefulness.

J. Citations

[1] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C. Halpern, Susana Puig, Josep Malvehy: “BCN20000: Dermoscopic Lesions in the Wild”, 2019; arXiv:1908.02288.`

[2] Saujanya Nath Tiwari; Naman Deep; Shreya; Sudhansu Kumar Mishra .Deep Learning based Malignant Melanoma Detection in Dermoscopy Images.https://ieeexplore.ieee.org/document/10094404

[3]Deep learning — Computer vision (CV) using Transfer Learning (ResNet-18) in Pytorch — Skin cancer classification. | by Kiel Dang | Medium

My Individual Contribution

In this project, I was responsible for the entire **data preprocessing pipeline**, the design of the **custom ResNet-18 model architecture**, and **training and evaluation monitoring**. Each step was implemented from scratch using PyTorch and OpenCV, with modifications tailored for the ISIC 2019 skin lesion classification dataset.

1. Image Preprocessing

I implemented three key preprocessing techniques:

- **Cropping Images** using a custom function `crop_image_from_gray()` to remove low-intensity borders and focus on lesion areas.
- **CLAHE in LAB Color Space** via the `clahe_lab()` function to enhance contrast in the L (lightness) channel, helping the model capture subtle lesion features.
- **Ben Graham's Method**, which involved dividing the original image by its Gaussian-blurred version to normalize lighting and emphasize edges.

Each of these steps helped improve the clarity and quality of the dermoscopic images before feeding them into the model.

2. Dataset Preparation

I managed the loading and formatting of the ISIC 2019 dataset. This included:

- Extracting disease labels from the metadata CSV
- Converting them into binary multi-label format
- Saving them as `.pt` and `.npy` tensors for efficient use in PyTorch

3. Model Architecture

I modified a pretrained ResNet-18 model:

- Removed the original classification head (`nn.Identity()`)
- Added a custom head with linear layers, ReLU, dropout, and sigmoid to support multi-label output for nine skin lesion classes
- Implemented the forward pass logic using matrix equations in line with standard neural network design principles

4. Training and Evaluation Setup

I set up:

- The training loop with loss tracking
- The validation loop to monitor overfitting
- Logging of training and validation loss per epoch
- Visualization of loss curves for analysis