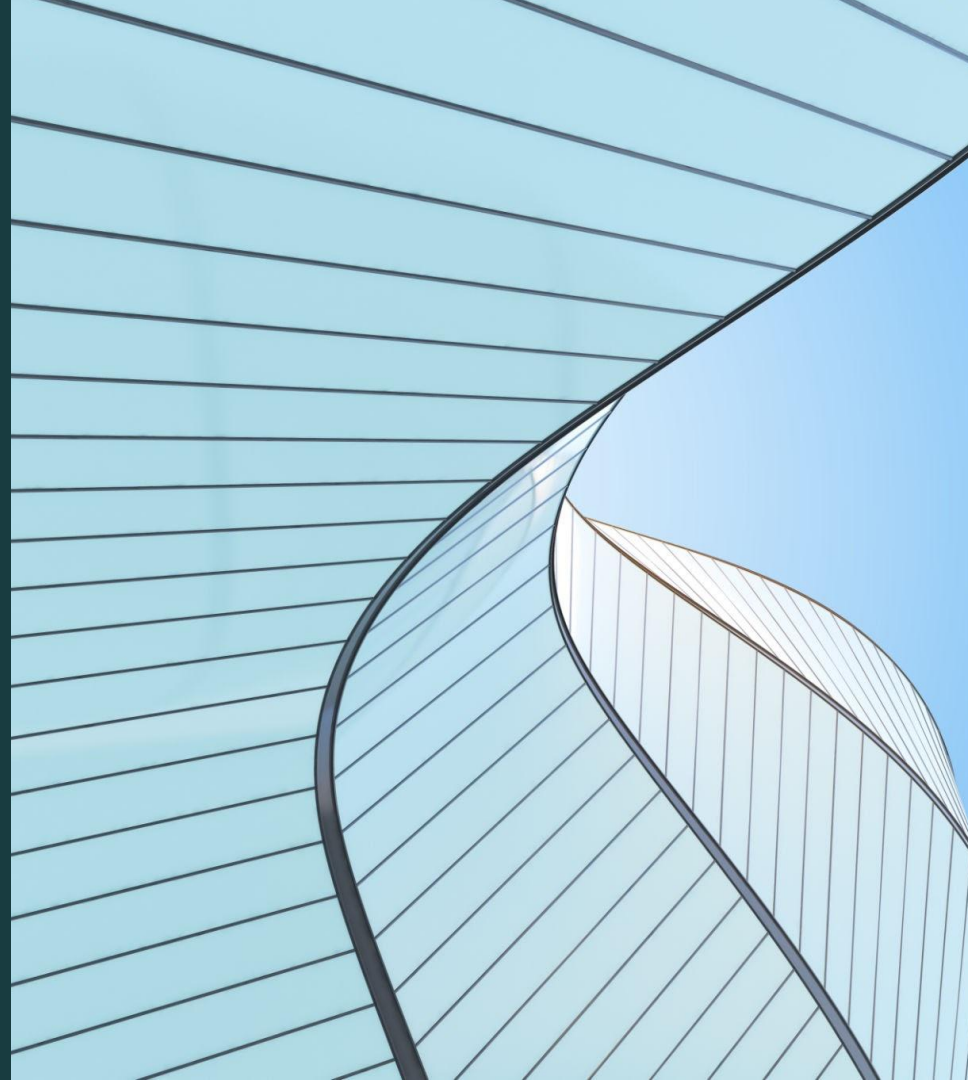


Skin Cancer Detection Milestone 2

By : Samuel Borges, Elif Dila Bayrak, Neva Ildem, Sophia
Zubieta



Progress Matrix Milestone 2

Task	Completion %	Elif	Neva	Samuel	Sophia
Dataset & Preprocessing Pipeline Execution	100%	0%	50%	0%	50%
CNN Model Training	100%	50%	50%	0%	0%
Mobile App Frontend UI	100%	0%	0%	100%	0%
REST API Backend & Preprocessing Endpoint	100%	0%	0%	0%	100%
Model Evaluation & Confusion Matrix	100%	50%	50%	0%	0%
App to Server Image Upload	100%	0%	0%	50%	50%

Milestone 2 Objectives & Progress

Milestone 2 Objectives

- Prepare and preprocess the ISIC 2019 skin lesion dataset for binary classification.
- Develop, train, and evaluate CNN and transfer learning models.
- Implement the mobile application's image capture and upload interface.
- Establish the Flask REST API and image preprocessing endpoint.
- Connect the mobile frontend to the backend server.

Milestone 2 Accomplishments

- Completed dataset cleaning, preprocessing, and lesion-level splitting.
- Trained and evaluated CNN, MobileNetV2, and EfficientNetV2-B2.
- Implemented image capture, upload, and quality verification.
- Developed Flask API endpoints and successfully connected the mobile application to the server.

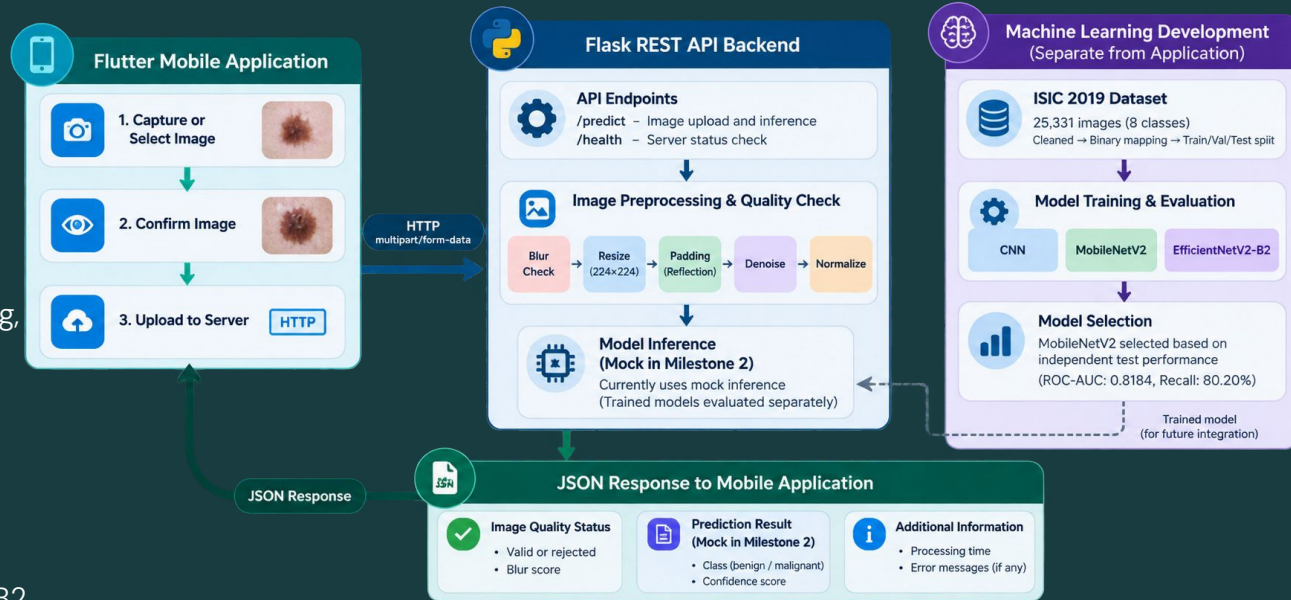
Technical Challenges & Solutions

- **Dataset Quality & Imbalance:** Resolved duplicate images, inconsistent labels, and potential data leakage through cleaning and lesion-level splitting.
- **Image Preprocessing:** Addressed resizing distortion and blur issues using reflection/blur padding and image quality checks.
- **Model Performance:** Investigated overfitting and class imbalance through fine-tuning, augmentation, weighted loss, and threshold optimization.
- **Frontend-Backend Communication:** Tested USB/ADB and Wi-Fi connections to establish reliable image uploads.
- **Incomplete Model Integration:** Used mock inference to enable API testing while model development continued.

System Architecture & Workflow

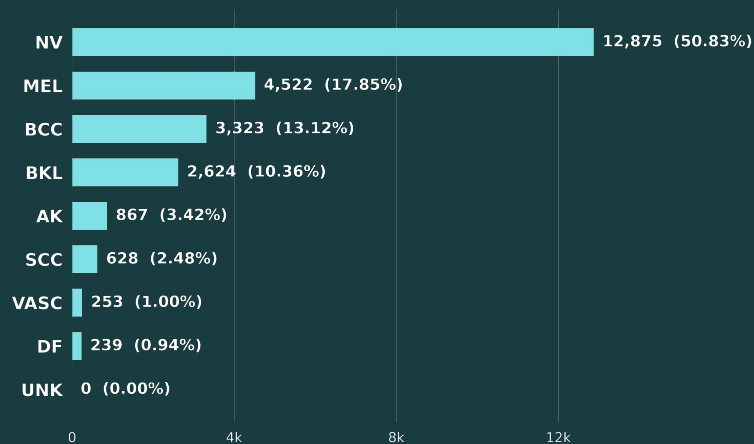
Milestone 2 Implementation

- **Frontend:** Flutter application for image capture, confirmation, and upload.
- **Backend:** Flask REST API with /health and /predict endpoints.
- **Preprocessing:** Image quality checks, padding, denoising, resizing, and normalization.
- **Communication:** HTTP image upload and JSON response displaying blur score and quality status.
- **Machine Learning:** CNN, MobileNetV2, and EfficientNetV2-B2 trained and evaluated separately; backend currently uses mock inference.



Dataset Preparation – ISIC 2019 Overview

- **Dataset:** ISIC 2019 Challenge – Training Dataset
- **Total images:** 25,331 labeled dermoscopic skin lesion images
- **Original classification:** Eight skin lesion diagnosis categories
- **Objective:** Prepare a reliable dataset for binary classification (benign vs. malignant)
- **Preparation approach:** Verify images, identify duplicates, map diagnosis labels, and create training, validation, and test sets
- **Tools:** Python, Pandas, Pillow, Google Colab, and Google Drive



Ground Truth Shape: (25331, 10)

```
Metadata Shape: (25331, 5)
```

[illegible]

Dataset Preparation – Image Verification & Duplicate Detection

Image Verification

- Verified all 25,331 official ISIC 2019 training images using Python's Pillow library.
- Checked that image files existed and could be opened successfully.
- Identified 3 additional files with duplicate-style filenames, which were not part of the official image list.

Exact Duplicate Detection

- Used SHA-256 hashing to identify images with identical file contents.
- Found 50 pairs of exact duplicate images.
- Identified 2 duplicate pairs with conflicting diagnosis labels.
- Excluded conflicting records and removed redundant duplicate images.

Final Result

- **Original dataset:** 25,331 images
- **Cleaned dataset:** 25,279 images
- **Total reduction:** 52 images
- Saved the cleaned dataset as ISIC2019_cleaned.csv.

First 20 duplicate image records:

	duplicate_group	image	lesion_id	diagnosis
0	1	ISIC_0012127_downsampled	NaN	NV
1	1	ISIC_0012970_downsampled	MSK4_0011472	NV

Conflicting image details:

	duplicate_group	image	lesion_id	diagnosis
92	47	ISIC_0067502	BCN_0003560	MEL
93	47	ISIC_0067980	BCN_0000237	NV

Preliminary Dataset Cleaning Summary

Original image records: 25331
Conflicting records excluded: 4
Redundant image records: 48
Images after cleaning: 25279

Dataset Preparation – Binary Label Mapping

Binary Classification

- The original ISIC 2019 dataset contains 8 diagnosis categories.
- Our model requires 2 output classes: Benign (0) and Malignant (1).
- Created a preliminary diagnosis-to-binary-label mapping.

Diagnosis Mapping

- **Benign (0):** NV, BKL, DF, VASC
- **Malignant (1):** MEL, BCC, SCC
- **Temporarily excluded:** AK (Actinic Keratosis)

Preliminary Binary Classification Analysis

```
-----  
Cleaned images: 25279  
Temporarily excluded images: 859  
Binary classification images: 24420
```

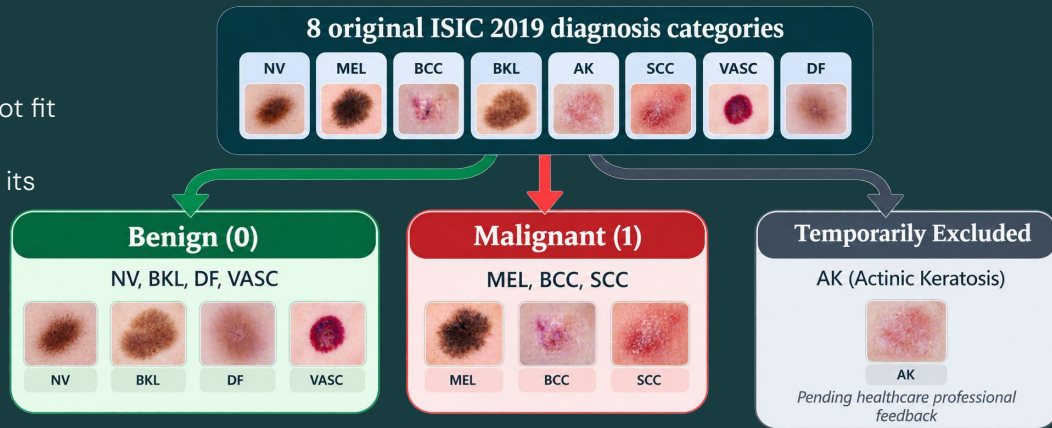
```
Binary label distribution:  
binary_label  
0      15964  
1      8456
```

Why Exclude AK?

- AK is generally considered a precancerous lesion and does not fit cleanly into our binary classification.
- Healthcare professional feedback is pending before finalizing its classification.

Results

- 24,420 images retained for preliminary binary classification.
- 15,964 benign (65.4%)
- 8,456 malignant (34.6%)



Dataset Preparation – Lesion IDs & Data Leakage Prevention

Why Lesion IDs Matter

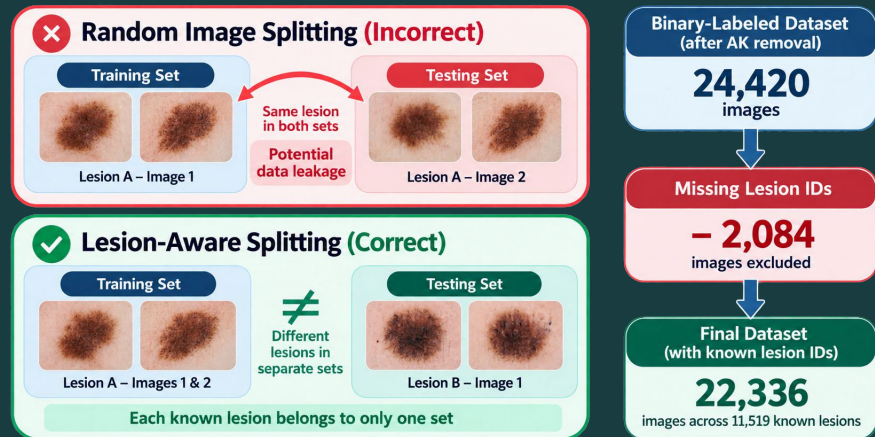
- Some skin lesions have multiple images taken from different angles or at different times.
- Images of the same lesion should remain in the same dataset split.
- Otherwise, the model could encounter the same lesion during training and testing, causing data leakage and overly optimistic evaluation results.

Missing Lesion ID Investigation

- Identified 2,084 images without lesion IDs in the preliminary binary dataset.
- Investigated a sample of 300 images using perceptual hashing (pHash).
- Found 18 potentially similar image pairs (pHash distance ≤ 10) and visually inspected the results.
- No convincing true duplicate pairs were confirmed from this sample.

Final Decision

- Excluded all 2,084 images without lesion IDs from the Milestone 2 splits.
- Retained 22,336 images representing 11,519 known lesions.
- Used lesion-aware splitting to keep images with the same lesion ID together.



Dataset Preparation – Stratified Train/Validation/Test Split

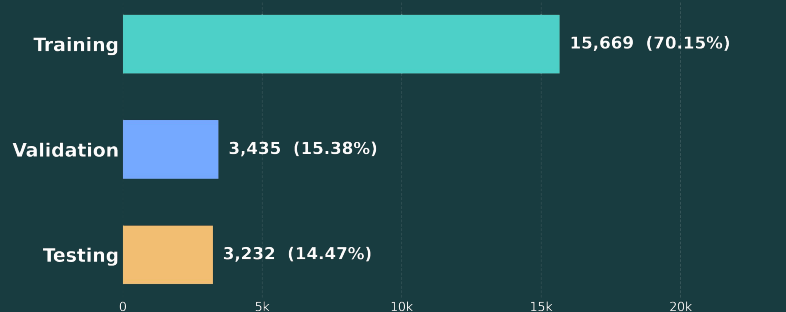
Final Dataset Split | 22,336 Images

Splitting Method

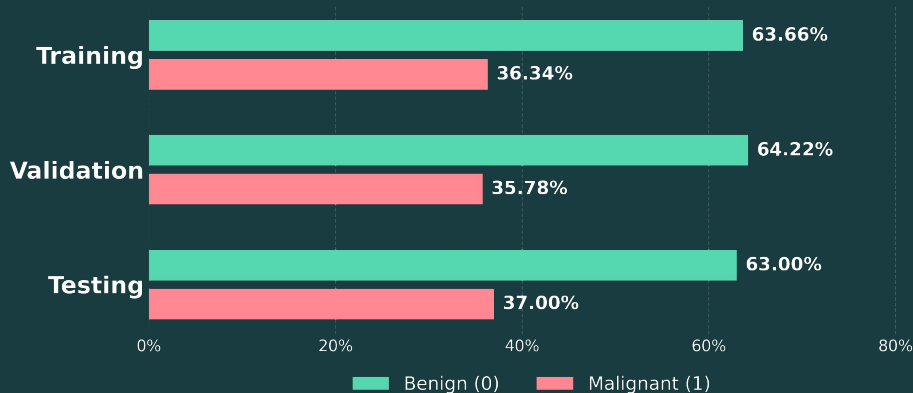
- Used StratifiedGroupKFold from Scikit-learn.
- **Stratification:** Maintains similar benign/malignant proportions across datasets.
- **Grouping:** Keeps all images with the same lesion ID in one split.
- Tested multiple random seeds to select a split close to our target 70% / 15% / 15% distribution.

Verification

- Confirmed no known lesion ID overlap between training, validation, and testing.
- Saved all three splits as separate CSV files.
- Reserved the test set for final model evaluation.



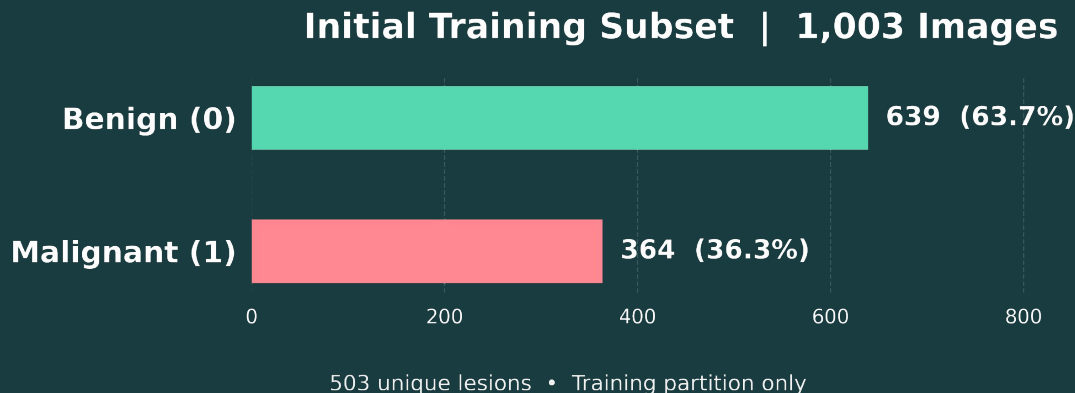
Class Distribution Across Final Splits



Dataset Preparation – Final Dataset & Preprocessing Handoff

Initial Training Subset

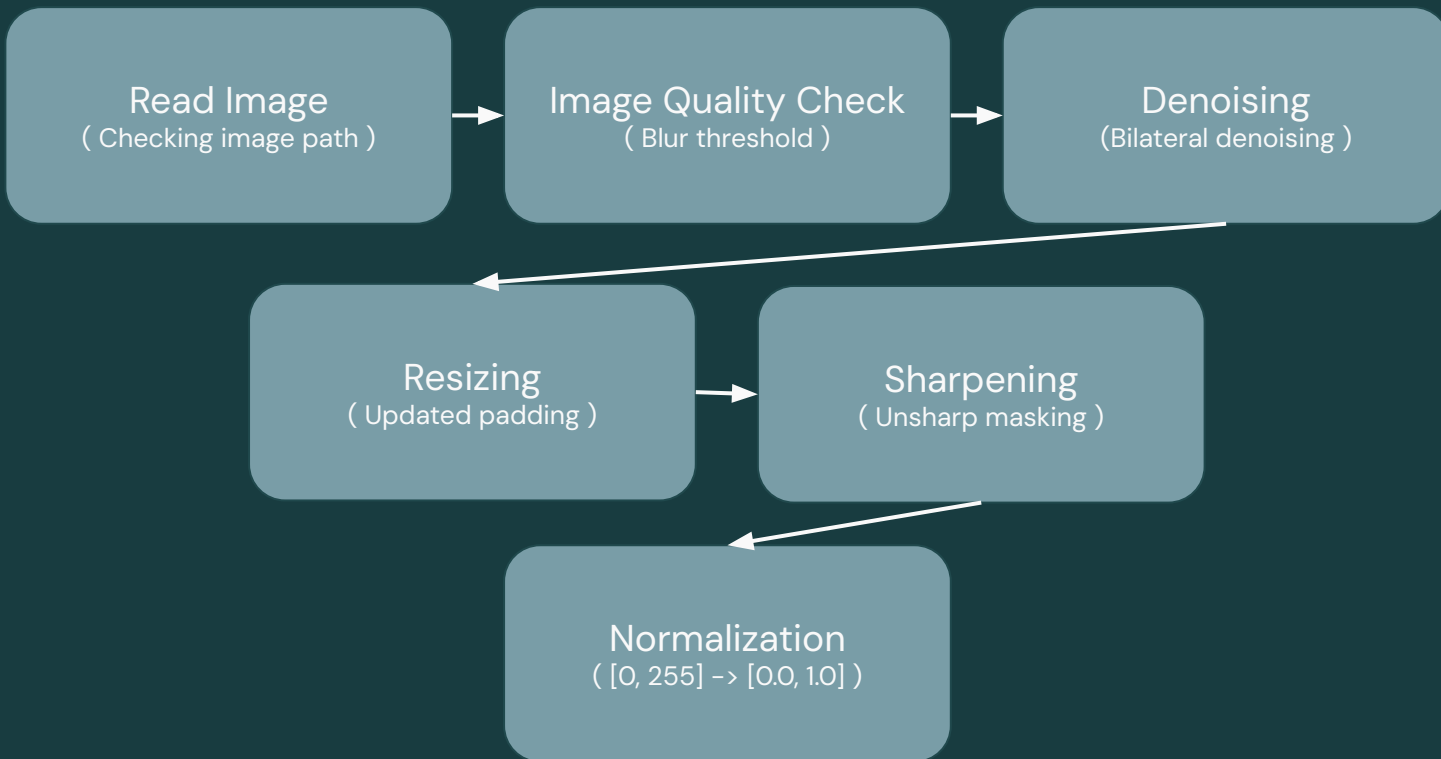
- Prepared a smaller dataset of approximately 1,000 images for initial model training.
- Selected 1,003 images from the training partition only.
- Contains 503 unique lesions:
 - 639 benign images (63.7%)
 - 364 malignant images (36.3%)
- Kept all selected images of the same lesion together.
- The full training dataset remains available for larger experiments.



Handoff

- All CSV files and original images are stored in Google Drive.
- Dataset partitions will remain unchanged to prevent leakage between training and evaluation.

Image Preprocessing Steps

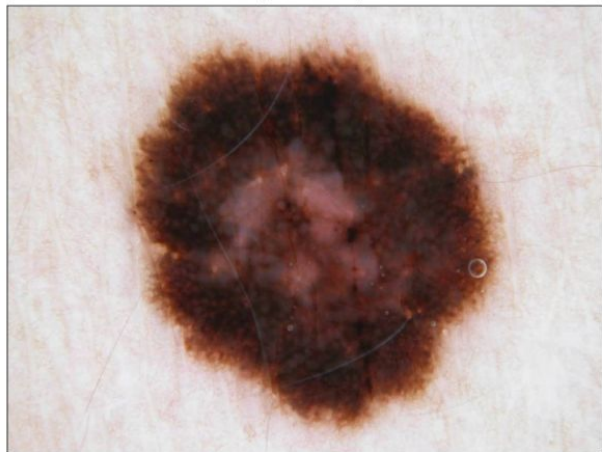


Preprocessing Aspect Ratio

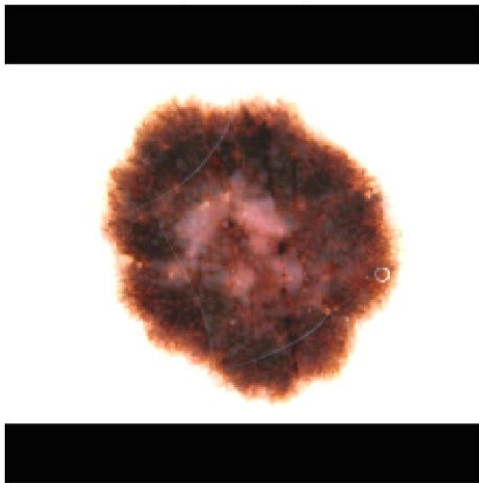
- An aspect of our preprocessing we wanted to improve upon was the padding included in our image to maintain aspect ratio when resizing images.
- Decided to use the image itself reflected and blurred to construct a more seamless background that will hopefully not negatively affect the training of the models
- In a future milestone we will return to further correct the border padding

Side-by-Side Normalization & Padding Comparison

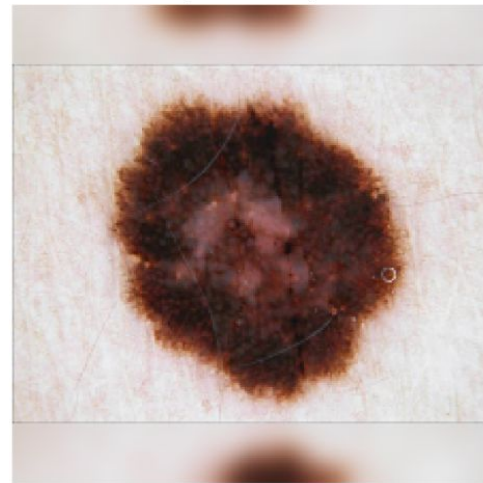
Original Image



Original Preprocessing Output
(Milestone 1)



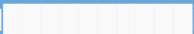
New Preprocessing Output
(Milestone 2)



Implementation Preprocessing

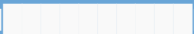
- Completing preprocessing pipeline for subset of ISIC 2019 dataset.
- Included
 - Training dataset – 1003 images
 - Validation dataset – 3435 images
- Output preprocessed sets into shared Google Drive folder to train models

Processing 1003 images from ISIC2019_M2_train_1000.csv...

100%  1003/1003 [02:29<00:00, 6.71it/s]

Done! Saved preprocessed files to /content/drive/MyDrive/Senior Design M2 (1)/ISIC2019/preprocessed_train_1000

Processing 3435 images from ISIC2019_M2_validation.csv...

100%  3435/3435 [08:24<00:00, 6.81it/s]

Done! Saved preprocessed files to /content/drive/MyDrive/Senior Design M2 (1)/ISIC2019/preprocessed_validation

Laplacian Variance Blur Threshold

- Once preprocessing of training and validation sets concluded ran verification stats
- Threshold was too high, almost all the images in the ISIC 2019 dataset were being rejected due to blur
- Reconfigured and optimized blur threshold score
 - Originally 100.0
 - Changed to 15.0
- Supports real-world skin photos

Original Blur Threshold (100.0)

```
def check_image_quality(image_bgr, blur_threshold=100.0):
```

```
✓ Found dataset folder at: /content/drive/MyDrive/Senior Design M2  
(1)/ISIC2019
```

```
=====
VERIFICATION STATS: TRAIN 1000 SUBSET
=====
```

```
Expected Input Images : 1003  
Images Passed QC : 126 (12.56%)  
Images Rejected (Blur): 877  
Saved Images on Disk : 126
```

```
✓ SUCCESS: Saved image count perfectly matches QC log passed count!
```

```
=====
VERIFICATION STATS: VALIDATION SPLIT
=====
```

```
Expected Input Images : 3435  
Images Passed QC : 509 (14.82%)  
Images Rejected (Blur): 2926  
Saved Images on Disk : 509
```

```
✓ SUCCESS: Saved image count perfectly matches QC log passed count!
```

Blur Threshold

- Once verification stats for training and validation sets were established
- Pytorch dataloader test verification was established
 - To ensure preprocessed images are at desired states

Updated blur threshold (15.0)

```
def check_image_quality(image_bgr, blur_threshold=15.0):  
# Updated default threshold to 15.0
```

✓ Found dataset folder at: /content/drive/MyDrive/Senior Design M2 (1)/ISIC2019

=====

VERIFICATION STATS: TRAIN 1000 SUBSET

=====

- Expected Input Images : 1003
- Images Passed QC : 893 (89.03%)
- Images Rejected (Blur): 110
- Saved Images on Disk : 893
- Class Breakdown : Benign (0): 581 | Malignant (1): 312

✓ SUCCESS: Saved image count perfectly matches QC log passed count!

=====

VERIFICATION STATS: VALIDATION SPLIT

=====

- Expected Input Images : 3435
- Images Passed QC : 2997 (87.25%)
- Images Rejected (Blur): 438
- Saved Images on Disk : 2997
- Class Breakdown : Benign (0): 1986 | Malignant (1): 1011

✓ SUCCESS: Saved image count perfectly matches QC log passed count!

=====

PYTORCH DATALOADER TEST VERIFICATION

=====

✓ DataLoader operational!

- Batch Image Tensor Shape : torch.Size([16, 3, 224, 224]) -->
(Batch, Channels, Height, Width)

- Batch Label Tensor Shape : torch.Size([16])

- Pixel Range Verification : Min=0.00, Max=1.00

=====

CNN Model Training

- **Dataset: ISIC 2019 skin lesion images**
- **Training images: 893**
- **Validation images: 2,997**
- **Model: Basic Convolutional Neural Network (CNN)**
- **Trained for 10 epochs using PyTorch**
- **Goal: Classify skin lesions as benign or malignant**

```
Device: cuda
CNN model created successfully!
SkinCancerCNN(
  (features): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=100352, out_features=128, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

Figure 1. CNN Model Architecture: The Basic CNN was successfully created using PyTorch with GPU acceleration (CUDA). The model uses convolutional, ReLU, and max-pooling layers to extract image features, followed by fully connected layers to classify skin lesions as benign or malignant.

CNN Implementation Details

- **Framework:** PyTorch with CUDA GPU support.
- **Architecture:** Three convolutional layers (32, 64, 128 filters).
- **Activation & Pooling:** ReLU and MaxPool2D.
- **Regularization:** Dropout (0.5) to help prevent overfitting.
- **Fully Connected Layer:** Approximately 12.8 million parameters.
- **Optimizer:** Adam with a learning rate of 0.0001.
- **Loss Function:** Weighted Cross-Entropy to address class imbalance.

```
Class weights: tensor([0.7685, 1.4311])  
Loss function: CrossEntropyLoss()  
Optimizer: Adam  
Learning rate: 0.0001
```

Figure 2. CNN Training Configuration:
The model uses the Adam optimizer with a learning rate of 0.0001 and weighted cross-entropy loss to account for class imbalance during training.

$\text{Linear}(100352, 128) = 100352 \times 128 + 128 = 12,845,184$

CNN Training Performance

- Trained the CNN for 10 epochs using PyTorch.
- Training accuracy increased from 54.42% to 74.36%.
- Final validation accuracy reached 69.60%.
- Training loss decreased from 0.6878 to 0.5036.
- Validation accuracy fluctuated, indicating room for improvement.

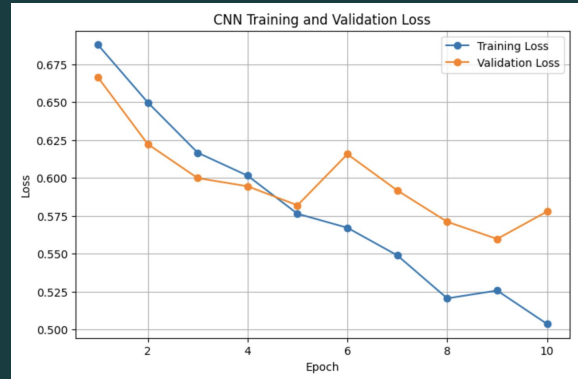


Figure 3. Training and Validation Loss: Training loss decreased over 10 epochs, while validation loss fluctuated.

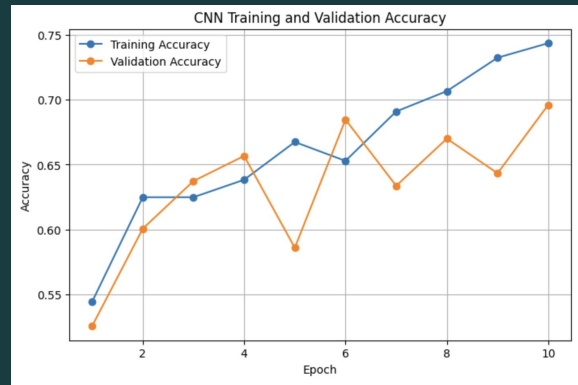


Figure 4. Training and Validation Accuracy: Training accuracy increased, while validation accuracy varied between epochs.

Classification Threshold Comparison

- Compared three classification thresholds: 0.30, 0.40, and 0.50.
- Lower thresholds helped detect more malignant lesions.
- Selected 0.30 to prioritize malignant detection.
- Achieved 85.56% malignant recall on validation data.
- Our goal was to reduce the number of missed malignant lesions.

Threshold	Accuracy	Malignant Precision	Malignant Recall	Malignant F1	False Positives	False Negatives
0.3	0.5949	0.4475	0.8556	0.5876	1068	146
0.4	0.6517	0.4893	0.7428	0.5899	784	260
0.5	0.6960	0.5429	0.6261	0.5815	533	378

Figure 5. Classification Threshold Comparison: Three thresholds were evaluated to balance overall accuracy and malignant detection. A threshold of 0.30 was selected because it achieved the highest malignant recall (85.56%), reducing missed malignant cases.

Final CNN Test Results

- Evaluated the CNN on 2,823 independent test images.
- Used the selected classification threshold of 0.30.
- Achieved 60.64% overall test accuracy.
- Malignant recall reached 84.59%.
- Correctly identified 829 malignant lesions.
- Missed 151 malignant lesions, showing room for improvement.

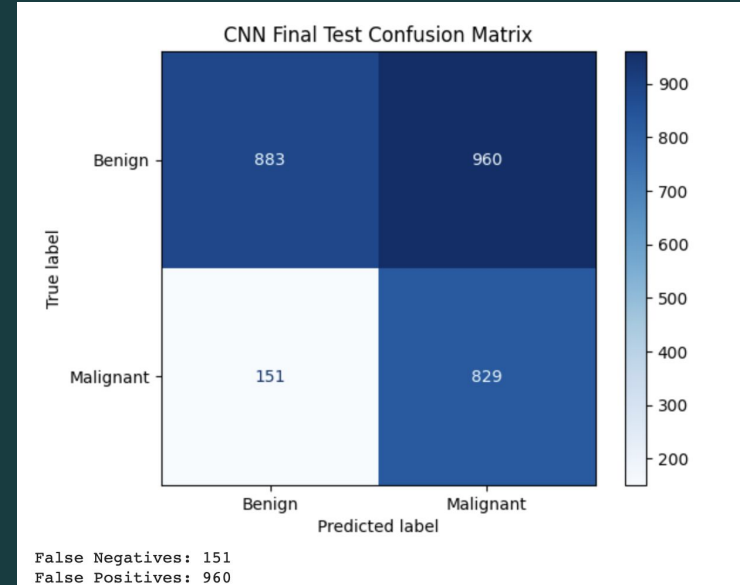


Figure 6. Final Test Confusion Matrix: The CNN correctly classified 829 malignant and 883 benign lesions. However, it missed 151 malignant lesions and incorrectly classified 960 benign lesions as malignant.

Training Environment & Dataset Verification

Development Environment

- Python and PyTorch
- ALPanther GPU computing environment
- CUDA for GPU acceleration
- torchvision (MobileNetV2) and timm (EfficientNetV2-B2) model libraries

```
Python: 3.12.3
PyTorch: 2.14.1+cu126
Torchvision: 0.29.1+cu126
Pandas: 3.0.6
Scikit-learn: 1.9.1
```

```
CUDA available: True
GPU: NVIDIA L40S-12Q
GPU count: 1
CUDA version: 12.6
```

Dataset

- 893 preprocessed training images
- 2,997 validation images
- Binary classification: benign (0), malignant (1)
- RGB image inputs

```
Training
-----
CSV records: 1003
Processed images: 893
Matched images: 893
Benign: 581
Malignant: 312
Unmatched image files: 0
```

```
Validation
-----
CSV records: 3435
Processed images: 2997
Matched images: 2997
Benign: 1986
Malignant: 1011
Unmatched image files: 0
```

Transfer Learning – Data Loading & GPU Training

Data Loading

- PyTorch Dataset and DataLoader for image and label handling
- Mini-batch processing for efficient training
- Separate training and validation data pipelines

```
Training batches: 56  
Validation batches: 188  
Batch image shape: torch.Size([16, 3, 224, 224])  
Batch label shape: torch.Size([16])  
Device available: cuda
```

GPU Training

- CUDA-enabled NVIDIA L40S-12Q GPU
- Model and image batches transferred to the selected device
- Forward propagation, loss computation, and backpropagation
- Validation performed without gradient updates

```
Best model saved!  
Epoch 1/20 | Train Loss: 0.1043 | Train Acc: 0.9048 | Val Loss: 0.2876 | Val Acc: 0.7601 | Recall: 0.6330 |  
F1: 0.6403 | AUC: 0.8265 | Time: 6.0s  
Epoch 2/20 | Train Loss: 0.0893 | Train Acc: 0.9250 | Val Loss: 0.2882 | Val Acc: 0.7668 | Recall: 0.6588 |  
F1: 0.6558 | AUC: 0.8321 | Time: 5.3s  
Epoch 3/20 | Train Loss: 0.0749 | Train Acc: 0.9384 | Val Loss: 0.3317 | Val Acc: 0.7688 | Recall: 0.5569 |  
F1: 0.6190 | AUC: 0.8288 | Time: 5.3s
```

MobileNetV2 – Experiment 1: Full Fine-Tuning

Model Initialization

- Loaded MobileNetV2 with ImageNet-pretrained weights
- Replaced the original classifier with a single-output linear layer (1,280 → 1)
- Transferred the model to the CUDA device

```
Alpha: 0.6506
Benign weight: 0.3494
Malignant weight: 0.6506
Loss function: AlphaWeightedBCELoss
```

Training Configuration

- 20 epochs, full-model fine-tuning
- Adam optimizer and alpha-weighted binary cross-entropy
- 893 training images and 2,997 validation images

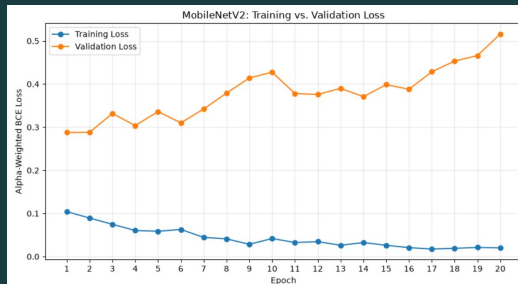
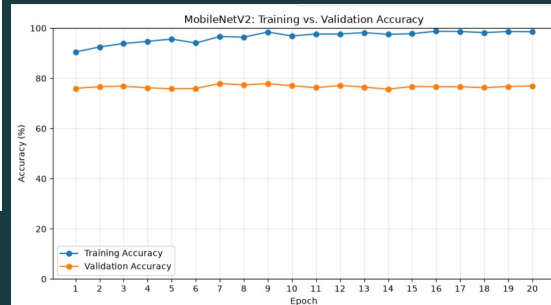
Training Analysis

- Monitored training and validation loss across epochs
- Tracked training and validation accuracy
- Evaluated malignant recall, F1-score, and ROC-AUC
- Saved the best-performing model checkpoint during training

Initial Training Configuration

- Full-model fine-tuning (pretrained layers not frozen)
- Adam optimizer with learning rate 0.0001
- Alpha-weighted binary cross-entropy loss
- Class weights: benign 0.3494, malignant 0.6506

```
Downloading: "https://download.pytorch.org/models/mobilenet_v2-7ebf99e0.pth" to
ch/hub/checkpoints/mobilenet_v2-7ebf99e0.pth
100% |██████████| 13.6M/13.6M [00:00<00:00, 57.7MB/s]
Device: cuda
Model: MobileNetV2
Final classification layer: Linear(in_features=1280, out_features=1, bias=True)
```



MobileNetV2: Experiment 2 – Frozen Backbone

Objective

- Reduce overfitting by freezing the pretrained feature extractor and training only the new binary classification head.

Implementation

- Loaded a fresh ImageNet-pretrained MobileNetV2.
- Froze the convolutional feature-extraction layers.
- Kept the final classification layer trainable.
- Used the same alpha-weighted BCE loss as Experiment 1.
- Optimized only the unfrozen parameters.

```
Experiment 2 model created!  
Total parameters: 2225153  
Trainable parameters: 1281  
Frozen parameters: 2223872  
Device: cuda
```

```
Experiment 2 loss: AlphaWeightedBCELoss  
Alpha: 0.6506  
Optimizer: Adam  
Learning rate: 0.0001  
Trainable parameters: 1281
```

Training Results

- Completed 20 epochs with the pretrained backbone frozen.
- Training accuracy increased from 54.42% to 77.83%.
- Validation loss decreased from 0.3090 to 0.2497.
- Best checkpoint saved based on minimum validation loss.

Key Observation

- Freezing the backbone produced more stable validation-loss behavior than Experiment 1, although the final validation accuracy was lower.

```
Epoch 17/20 | Train Loss: 0.2371 | Train Acc: 0.7816 | Val Loss: 0.2526 | Val Acc: 0.7314 | Recall: 0.6756 |  
F1: 0.6292 | AUC: 0.8002 | Time: 3.8s  
Best Experiment 2 model saved!  
Epoch 18/20 | Train Loss: 0.2329 | Train Acc: 0.7861 | Val Loss: 0.2518 | Val Acc: 0.7364 | Recall: 0.6637 |  
F1: 0.6295 | AUC: 0.8006 | Time: 4.0s  
Best Experiment 2 model saved!  
Epoch 19/20 | Train Loss: 0.2341 | Train Acc: 0.7861 | Val Loss: 0.2508 | Val Acc: 0.7371 | Recall: 0.6637 |  
F1: 0.6300 | AUC: 0.8010 | Time: 4.0s  
Best Experiment 2 model saved!  
Epoch 20/20 | Train Loss: 0.2303 | Train Acc: 0.7783 | Val Loss: 0.2497 | Val Acc: 0.7364 | Recall: 0.6746 |  
F1: 0.6332 | AUC: 0.8015 | Time: 3.8s  
  
Experiment 2 training complete!  
Best checkpoint: /home1/zildem2023/MobileNetV2_Project/checkpoints/mobilenetv2_exp2_frozen_best.pth  
Training history: /home1/zildem2023/MobileNetV2_Project/results/mobilenetv2_exp2_frozen_20epochs_history.csv
```

MobileNetV2: Experiment 3 – Partial Fine-Tuning

Objective

- Improve feature adaptation while limiting overfitting by unfreezing only the last three feature-extraction blocks.

Configuration

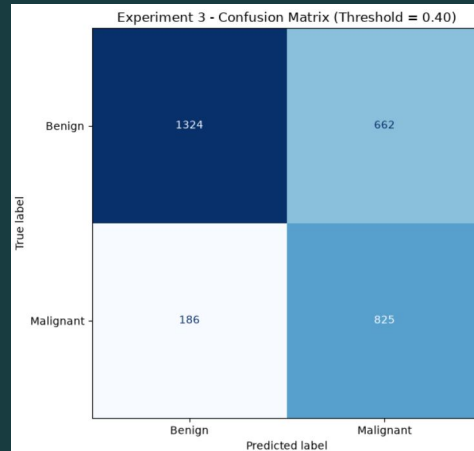
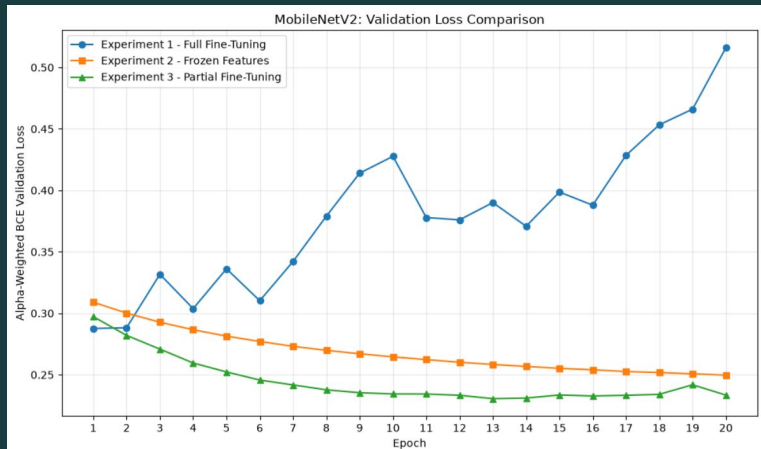
- 1,207,361 trainable parameters; 1,017,792 frozen
- Adam optimizer with separate learning rates: feature blocks 10^{-5} , classifier 10^{-4}
- Alpha-weighted BCE loss ($\alpha = 0.6506$); 20 training epochs

Validation Results

- Best checkpoint: Epoch 13
- Selected threshold: 0.40
- Accuracy: **71.71%** | Malignant recall: **81.60%**
- F1-score: **0.6605** | ROC-AUC: **0.8214**

Experiment 3 model created!
Total parameters: 2225153
Trainable parameters: 1207361
Frozen parameters: 1017792
Device: cuda

Experiment 3 loss: AlphaWeightedBCELoss
Alpha: 0.6506
Optimizer: Adam
Feature learning rate: 1e-05
Classifier learning rate: 0.0001



MobileNetV2 – Experiment 4: Hyperparameter Grid Search

Objective

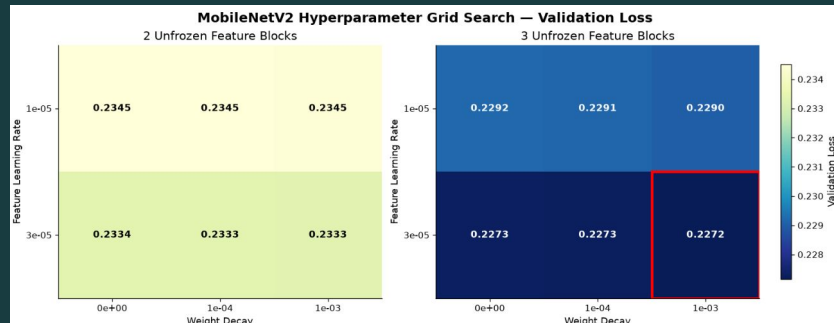
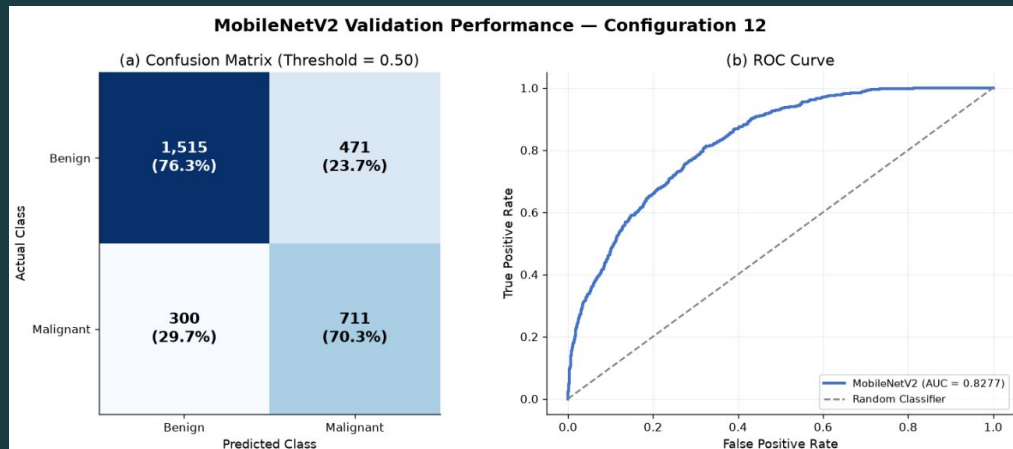
- Optimize partial fine-tuning by systematically testing different learning rates, weight decay values, and numbers of unfrozen feature blocks.

Experimental Configuration

- 12 configurations: 2 feature learning rates $\times$ 3 weight decay values $\times$ 2 unfreezing strategies
- Adam optimizer with separate feature and classifier learning rates
- ReduceLROnPlateau scheduler and early stopping
- Maximum 30 epochs per configuration

Selected Model Configuration 12

- Feature learning rate: 3×10^{-5}
- Weight decay: 0.001
- Last 3 feature blocks unfrozen
- Best checkpoint: Epoch 6
- Validation ROC-AUC: 0.8277



MobileNetV2 – Experiment 5: SE Attention

Objective

- Evaluate whether channel-wise attention improves MobileNetV2's ability to identify relevant skin lesion features.

Implementation

- Added a Squeeze-and-Excitation attention module to the pretrained MobileNetV2 architecture.
- Partially fine-tuned the backbone with the last 3 feature blocks unfrozen.
- Used Adam with separate learning rates for backbone, attention module, and classifier.
- Applied alpha-weighted BCE loss and early stopping.

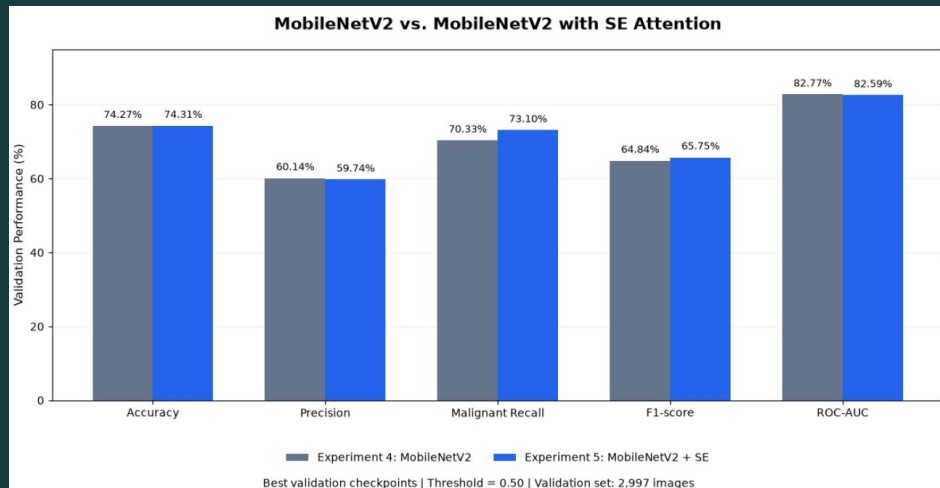
Selected Model Configuration 12

- Best checkpoint: **Epoch 8**
- Validation accuracy: **74.31%**
- Malignant recall: **73.10%**
- F1-score: **0.6575**

Baseline: Experiment 4, Configuration 12
Feature learning rate: 3e-5
Classifier learning rate: 1e-4
Weight decay: 1e-3
Unfrozen feature blocks: 3
Loss: AlphaWeightedBCELoss
Maximum epochs: 30
Early stopping patience: 5

SE attention module created successfully!
Input shape: torch.Size([2, 128, 7, 7])
Output shape: torch.Size([2, 128, 7, 7])
SE parameters: 2184

Device: cuda
GPU: NVIDIA L40S-12Q



MobileNetV2 – Experiment 6: ASPP

Objective

- Evaluate whether multi-scale feature extraction improves MobileNetV2's classification performance.

Implementation

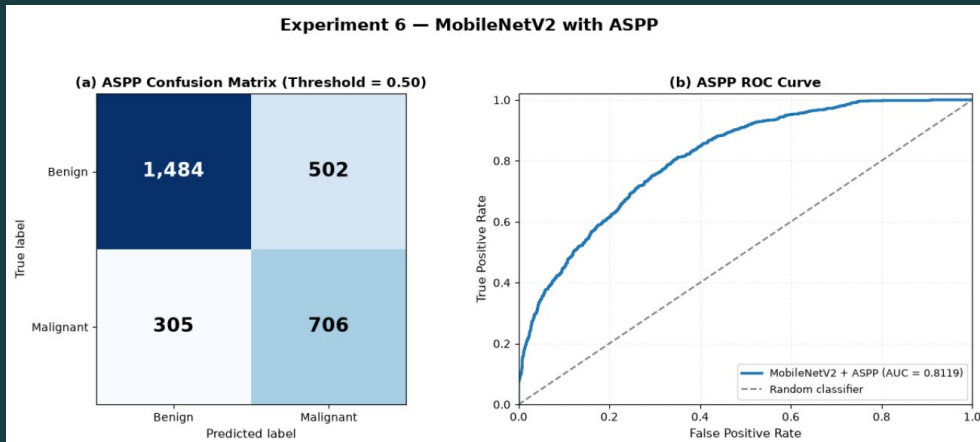
- Added ASPP with dilation rates of 1, 2, and 3, plus a global-context branch
- Unfroze the final 3 MobileNetV2 feature blocks.
- Adam optimizer: backbone LR 3×10^{-5} ; ASPP/classifier LR 3×10^{-4} .
- Alpha-weighted BCE loss; weight decay 0.001; early stopping.

Selected Model Configuration 12

- Best checkpoint: **Epoch 8**
- Validation accuracy: **74.31%**
- Malignant recall: **73.10%**
- F1-score: **0.6575**

Experiment 6 model created successfully!
Total parameters: 5566337
Trainable parameters: 4548545
Frozen parameters: 1017792
Output shape: torch.Size([2, 1])
Device: cuda

Best F1 threshold:
Threshold: 0.40
Accuracy: 0.7074
Precision: 0.5459
Recall: 0.7883
F1-score: 0.6451



MobileNetV2 – Experiment 7: ASPP + SE Attention

Objective

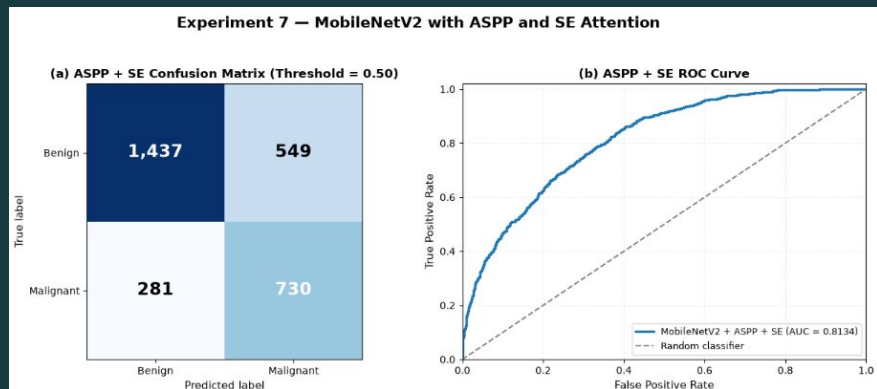
- Evaluate whether combining multi-scale feature extraction with channel-wise attention improves skin lesion classification.

Implementation

Validation Results – Best Checkpoint

- Accuracy: **72.31%**
- Malignant Recall: **72.21%**
- F1-Score: **0.6376**
- ROC-AUC: **0.8134**
- Classification Threshold: **0.50**

- ImageNet-pretrained MobileNetV2 with ASPP and SE modules
- Final 3 backbone feature blocks unfrozen
- Adam optimizer: backbone LR 3×10^{-5} ; ASPP, SE, and classifier LR 1×10^{-4} .
- Alpha-weighted BCE loss, weight decay 0.001, and ReduceLROnPlateau scheduler
- Maximum 30 epochs with early stopping



Experiment 7 best checkpoint restored!

Saved best epoch: 2

Restored model validation metrics:

Validation loss: 0.2372
Accuracy: 0.7231
Precision: 0.5708
Recall: 0.7221
F1-score: 0.6376
ROC-AUC: 0.8134

Checkpoint verification:

loss: difference = 0.00000000
accuracy: difference = 0.00000000
precision: difference = 0.00000000
recall: difference = 0.00000000
f1: difference = 0.00000000
roc_auc: difference = 0.00000000

MobileNetV2 – Architecture Comparison

Objective

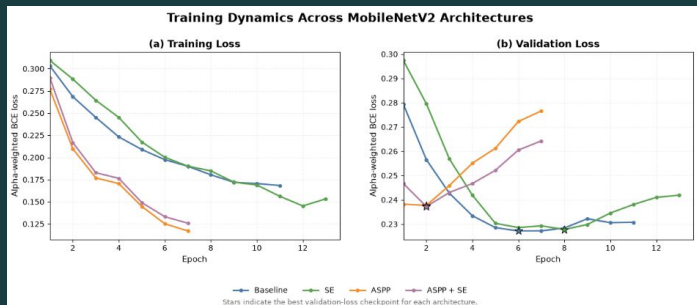
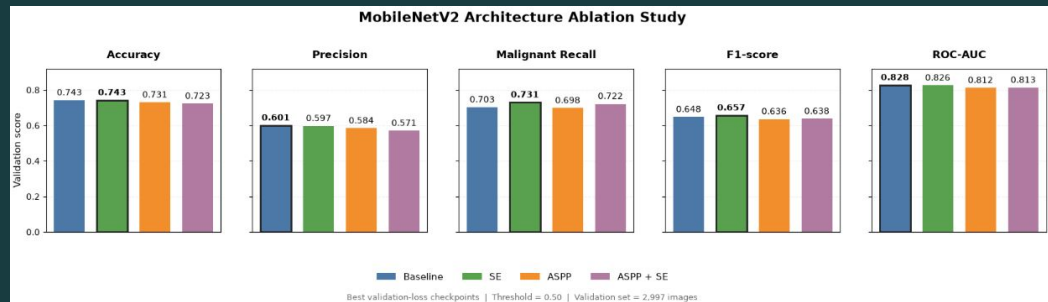
- Compare the selected baseline MobileNetV2 against three modified architectures.
- Determine whether attention and multi-scale feature extraction improve validation performance.

Architectures Evaluated

- **Baseline (Experiment 4):** Optimized partial fine-tuning using grid search.
- **SE Attention (Experiment 5):** Emphasizes important feature channels.
- **ASPP (Experiment 6):** Extracts features at multiple spatial scales.
- **ASPP + SE (Experiment 7):** Combines both techniques.

Milestone 2 architecture comparison saved!

Architecture	Validation Loss	F1	ROC-AUC	Best F1 Threshold	Best Threshold F1
MobileNetV2 Baseline	0.2272	0.6484	0.8277	0.40	0.6629
MobileNetV2 + SE	0.2277	0.6575	0.8259	0.50	0.6575
MobileNetV2 + ASPP	0.2377	0.6363	0.8119	0.40	0.6451
MobileNetV2 + ASPP + SE	0.2372	0.6376	0.8134	0.35	0.6478



MobileNetV2 – Final Model Selection & Threshold Optimization

Selected Model

- Experiment 4, Configuration 12 — Grid Search
- Partial fine-tuning: last 3 backbone blocks unfrozen
- Feature learning rate: 3×10^{-5}
- Weight decay: 0.001
- Best checkpoint: Epoch 6
- Highest validation ROC-AUC among the compared architectures: **0.8277**

Milestone 2 architecture comparison saved!

Architecture	Validation Loss	F1	ROC-AUC	Best F1 Threshold	Best Threshold F1
MobileNetV2 Baseline	0.2272	0.6484	0.8277	0.40	0.6629
MobileNetV2 + SE	0.2277	0.6575	0.8259	0.50	0.6575
MobileNetV2 + ASPP	0.2377	0.6363	0.8119	0.40	0.6451
MobileNetV2 + ASPP + SE	0.2372	0.6376	0.8134	0.35	0.6478

Threshold Optimization

- Evaluated different classification thresholds using validation predictions.
- Selected **0.40** to improve malignant detection.
- Predicted malignant probability $\geq 0.40 \rightarrow$ Malignant
- Predicted malignant probability below 0.40 $\rightarrow$ Benign

EfficientNetV2-B2 – Experiment 1: Two-Stage Fine-Tuning

Objective

- Establish an initial EfficientNetV2-B2 transfer learning baseline
- Gradually adapt pretrained features to benign/malignant classification.

Implementation

- **Stage 1: Frozen Backbone**
 - Train only the final classifier (1,409 trainable parameters).
 - AdamW optimizer, learning rate 1×10^{-3}
 - Weight decay: 1×10^{-4}
 - Best checkpoint: Epoch 15.
- **Stage 2: Partial Fine-Tuning**
 - Unfreeze the final feature block group, convolutional head, final BatchNorm, and classifier.
 - Trainable parameters: 6,589,075.
 - Backbone learning rate 1×10^{-5} ; classifier: 1×10^{-4} .
 - AdamW, weighted BCE loss, learning-rate scheduler, and early stopping.
 - Best checkpoint: Epoch 16.

```
EfficientNetV2-B2 Stage 1 Configuration
=====
Architecture: tf_efficientnetv2_b2.in1k
Device: cuda

Model parameters:
Total: 8,688,495
Trainable: 1,409
Frozen: 8,687,086

Classifier:
Linear(in_features=1408, out_features=1, bias=True)

Training settings:
Strategy: Frozen backbone, train classifier only
Optimizer: AdamW
Classifier learning rate: 0.001
Weight decay: 0.0001
Loss: AlphaWeightedBCELoss
```

```
EfficientNetV2-B2 Stage 1 Configuration
=====
Architecture: tf_efficientnetv2_b2.in1k
Device: cuda

Model parameters:
Total: 8,688,495
Trainable: 1,409
Frozen: 8,687,086

Classifier:
Linear(in_features=1408, out_features=1, bias=True)

Training settings:
Strategy: Frozen backbone, train classifier only
Optimizer: AdamW
Classifier learning rate: 0.001
Weight decay: 0.0001
Loss: AlphaWeightedBCELoss
```

```
EfficientNetV2-B2 Threshold Optimization
=====
threshold  accuracy  precision  recall  f1
0.20      0.6747    0.5128    0.7122  0.5963
0.25      0.6854    0.5254    0.6944  0.5982
0.30      0.6884    0.5304    0.6637  0.5896
0.35      0.6927    0.5371    0.6439  0.5857
0.40      0.7017    0.5507    0.6281  0.5869
0.45      0.7034    0.5549    0.6103  0.5813
0.50      0.7094    0.5654    0.5984  0.5815
0.55      0.7087    0.5670    0.5776  0.5723
0.60      0.7137    0.5775    0.5638  0.5706
0.65      0.7201    0.5919    0.5480  0.5691
0.70      0.7231    0.6009    0.5331  0.5650
0.75      0.7254    0.6106    0.5134  0.5578
0.80      0.7297    0.6242    0.4995  0.5549

Best validation F1 threshold:
Threshold: 0.25
Accuracy: 0.6854
Precision: 0.5254
Malignant recall: 0.6944
F1-score: 0.5982

ROC-AUC: 0.7442
```

```
EfficientNetV2-B2 – Best Stage 2 Checkpoint
=====
Checkpoint epoch: 16
Model: tf_efficientnetv2_b2.in1k

Saved validation metrics:
loss      : 0.500569
accuracy  : 0.709376
precision : 0.565421
recall    : 0.598417
f1        : 0.581451
roc_auc   : 0.744207
```

EfficientNetV2-B2 – Experiment 2: Conservative Fine-Tuning

Objective

- Reduce overfitting by limiting the number of trainable layers.
- Improve validation performance through stronger regularization.

Implementation

- Initialized from Experiment 1's best Stage 1 checkpoint (Epoch 15).
- Unfroze only the final 3 sub-blocks, convolutional head, final BatchNorm, and classifier.
- Trainable parameters: 2,297,789 / 8,688,495 (26.45%).
- Optimizer: AdamW with ReduceLROnPlateau scheduler.
- Backbone learning rate 1×10^{-5} ; classifier: 1×10^{-4} .
- Weight decay: 0.001 (10× Experiment 1).
- Loss: Alpha-weighted binary cross-entropy.
- Early stopping: patience 5; maximum 25 epochs.

```
Experiment 2 – Conservative Fine-Tuning
=====
Architecture: tf_efficientnetv2_b2.in1k
Initialization: Best Stage 1 checkpoint
Trainable final sub-blocks: 3

Parameters:
Total: 8,688,495
Trainable: 2,297,789
Frozen: 6,390,706

Stage 1 source epoch: 15

Experiment 2 model initialized successfully!
```

```
Experiment 2 Training Configuration
=====
Optimizer: AdamW
Scheduler: ReduceLROnPlateau

Learning rates:
Backbone: 1e-05
Classifier: 0.0001

Weight decay: 0.001
Loss: AlphaWeightedBCELoss

Frozen early BatchNorm layers: evaluation mode
Trainable final sub-blocks: 3

Experiment 2 training function ready!
```

```
EfficientNetV2-B2 – Experiment 2 Best Checkpoint
=====
Best epoch: 13
Architecture: tf_efficientnetv2_b2.in1k

Saved validation metrics:
loss      : 0.482876
accuracy  : 0.705706
precision : 0.554707
recall    : 0.646884
f1        : 0.597260
roc_auc   : 0.755254

Fine-tuning strategy: final_3_subblocks
Weight decay: 0.001
```

```
EfficientNetV2-B2 – Experiment 2 Threshold Optimization
=====
threshold accuracy precision recall f1
0.20 0.6643 0.5017 0.7507 0.6014
0.25 0.6753 0.5132 0.7280 0.6020
0.30 0.6834 0.5226 0.7082 0.6014
0.35 0.6907 0.5318 0.6944 0.6023
0.40 0.6977 0.5414 0.6785 0.6023
0.45 0.7027 0.5492 0.6627 0.6006
0.50 0.7057 0.5547 0.6469 0.5973
0.55 0.7077 0.5595 0.6381 0.5918
0.60 0.7154 0.5729 0.6142 0.5928
0.65 0.7191 0.5807 0.6014 0.5909
0.70 0.7187 0.5843 0.5757 0.5800
0.75 0.7237 0.5964 0.5598 0.5776
0.80 0.7277 0.6089 0.5391 0.5719

Best validation F1 threshold:
Threshold: 0.35
Accuracy: 0.6907
Precision: 0.5318
Malignant recall: 0.6944
F1-score: 0.6023

ROC-AUC: 0.7553
```

EfficientNetV2-B2 – Experiment Comparison

experiment	best_epoch	validation_loss	accuracy_at_0.50	recall_at_0.50	f1_at_0.50	roc_auc	best_f1_threshold	optimized_accuracy	optimized_precision	optimized_recall	optimized_f1
Exp1_Staged_Finetuning	16	0.5006	0.7094	0.5984	0.5815	0.7442	0.25	0.6854	0.5254	0.6944	0.5982
Exp2_Conservative_Finetuning	13	0.4829	0.7057	0.6469	0.5973	0.7553	0.35	0.6907	0.5318	0.6944	0.6023

EfficientNetV2-B2 – Experiment 3: Enhanced Data Augmentation

Objective

- Improve generalization by increasing training image diversity.
- Evaluate whether stronger augmentation improves malignant classification.

Implementation

- Initialized from the best Stage 1 checkpoint (Epoch 15).
- Unfroze the final 3 sub-blocks, convolutional head, final BatchNorm, and classifier.
- Trainable parameters: 2,297,789.
- AdamW optimizer with backbone learning rate 1×10^{-5} ; classifier: 1×10^{-4} .
- Weight decay: 0.001; alpha-weighted BCE loss.
- Maximum 25 epochs; early stopping patience 5.

EfficientNetV2-B2 – Experiment 3 Augmentation

=====

Training transform:

```
Compose(  
  Resize(size=(288, 288), interpolation=bicubic, max_size=None, antialias=True)  
  RandomHorizontalFlip(p=0.5)  
  RandomVerticalFlip(p=0.5)  
  RandomAffine(degrees=[-15.0, 15.0], translate=(0.05, 0.05), scale=(0.95, 1.05), interpolation=bilinear, fill=(0, 0, 0))  
  ColorJitter(brightness=(0.9, 1.1), contrast=(0.9, 1.1), saturation=(0.95, 1.05), hue=(-0.01, 0.01))  
  ToTensor()  
  Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225))  
)
```

Dataset verification:

Training images: 893
Validation images: 2997
Training batches: 56

Experiment 3 augmentation configured successfully!

EfficientNetV2-B2 – Experiment 3 Best Checkpoint

=====

Best epoch: 16

Architecture: tf_efficientnetv2_b2.in1k

Saved validation metrics:

loss : 0.475295
accuracy : 0.709376
precision : 0.561082
recall : 0.636004
f1 : 0.596198
roc_auc : 0.758775

Experiment configuration:

Fine-tuning: final_3_subblocks
Augmentation: enhanced
Weight decay: 0.001

EfficientNetV2-B2 – Experiment 3 Threshold Optimization

=====

threshold	accuracy	precision	recall	f1
0.20	0.6717	0.5092	0.7409	0.6035
0.25	0.6767	0.5149	0.7161	0.5991
0.30	0.6857	0.5258	0.6963	0.5991
0.35	0.6964	0.5394	0.6835	0.6030
0.40	0.7027	0.5489	0.6667	0.6021
0.45	0.7064	0.5553	0.6508	0.5993
0.50	0.7094	0.5611	0.6360	0.5962
0.55	0.7171	0.5747	0.6202	0.5966
0.60	0.7181	0.5789	0.6024	0.5904
0.65	0.7237	0.5901	0.5925	0.5913
0.70	0.7264	0.5984	0.5747	0.5863
0.75	0.7287	0.6076	0.5529	0.5790
0.80	0.7311	0.6190	0.5272	0.5694

EfficientNetV2-B2 – Experiment 4: Label Smoothing

Objective

- Reduce overconfident predictions through label smoothing.
- Improve malignant classification without changing the backbone fine-tuning strategy.

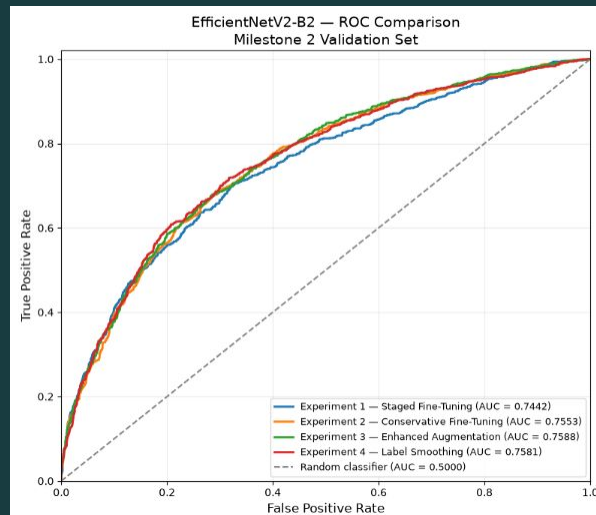
Implementation

- ImageNet-pretrained EfficientNetV2-B2, initialized from the best Stage 1 checkpoint.
- Final 3 sub-blocks, convolutional head, final BatchNorm, and classifier trainable.
- Enhanced augmentation retained from Experiment 3.
- AdamW optimizer with backbone learning rate 1×10^{-5} ; classifier: 1×10^{-4} .
- Weight decay: **0.001**.
- Alpha-weighted BCE loss with **label smoothing = 0.05**
- Maximum 30 epochs with early stopping and learning-rate scheduling

```
EfficientNetV2-B2 – Experiment 4 Loss Configuration
=====
Loss: LabelSmoothedAlphaWeightedBCE
Alpha: 0.650616
Label smoothing: 0.05

Original -> Smoothed targets:
0.0 -> 0.025
1.0 -> 0.975

Class weights:
Benign: 0.349384
Malignant: 0.650616
```



EfficientNetV2-B2 – Experiment Comparison & Final Selection

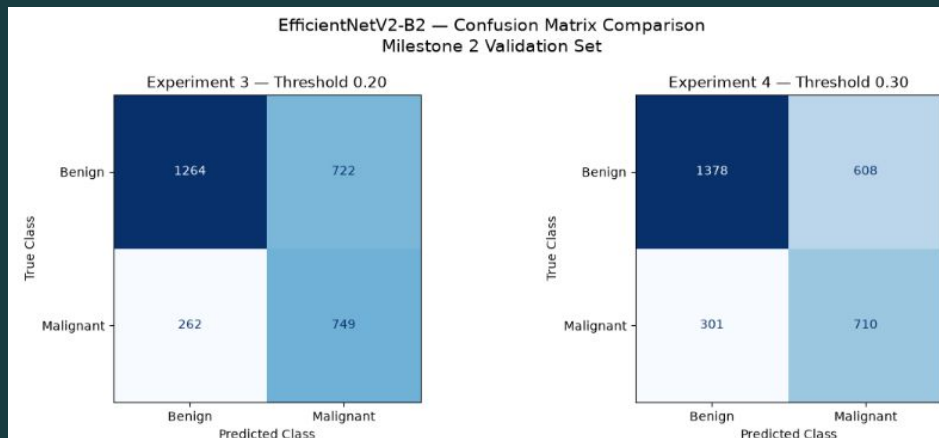
Objective

- Compare the performance of four transfer learning configurations.
- Select the model that provides the most useful balance between malignant recall, F1-score, and overall discrimination.

EfficientNetV2-B2 – Four-Experiment Comparison														
=====														
	experiment	best_epoch	validation_loss_unsmoothed	checkpoint_selection_loss	accuracy_at_0.50	precision_at_0.50	recall_at_0.50	f1_at_0.50	roc_auc	best_f1_threshold	optimized_accuracy	optimized_precision	optimized_recall	optimized_f1
0	Exp1_Staged_Finetuning	16	0.5006	0.5006	0.7094	0.5654	0.5984	0.5815	0.7442	0.25	0.6854	0.5254	0.6944	0.5982
1	Exp2_Conservative_Finetuning	13	0.4829	0.4829	0.7057	0.5547	0.6469	0.5973	0.7553	0.35	0.6907	0.5318	0.6944	0.6023
2	Exp3_Enhanced_Augmentation	16	0.4753	0.4753	0.7094	0.5611	0.6360	0.5962	0.7588	0.20	0.6717	0.5092	0.7409	0.6035
3	Exp4_Label_Smoothing	25	0.4487	0.4721	0.7171	0.5743	0.6231	0.5977	0.7581	0.30	0.6967	0.5387	0.7023	0.6097

Final Selection

- Achieved the highest optimized F1-score (**0.6097**) among all four experiments.
- Maintained a competitive ROC-AUC (**0.7581**), nearly matching Experiment 3 (**0.7588**).
- Selected for its improved precision–recall balance, combining conservative fine-tuning, enhanced augmentation, and label smoothing.



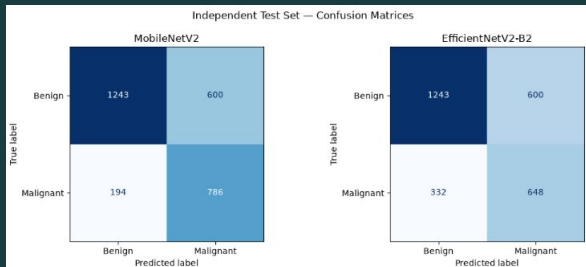
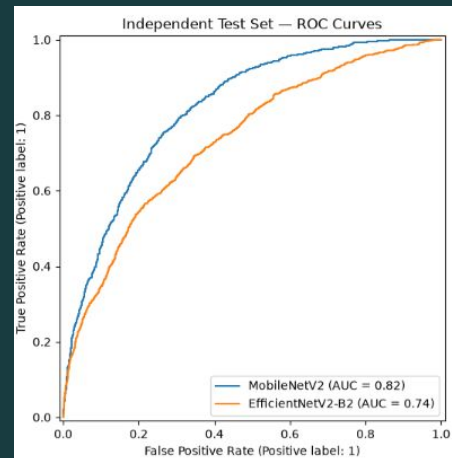
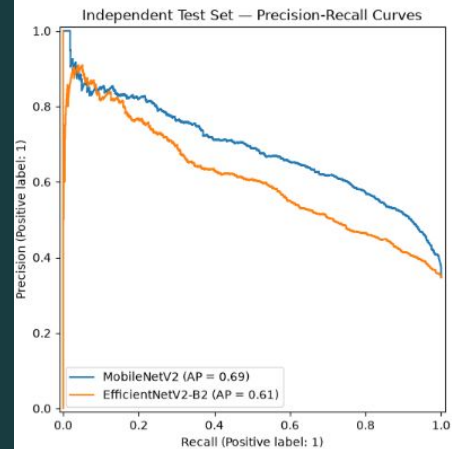
Independent Test Comparison & Final Model Selection

Objective

- Evaluate both selected models on the same independent test dataset.
- Select the stronger architecture for further optimization in Milestone 3.

Test Dataset

- 2,823 unseen images: 1,843 benign and 980 malignant.
- No test images used during training, validation, or threshold selection.
- MobileNetV2: Experiment 4, Configuration 12, threshold 0.40.
- EfficientNetV2-B2: Experiment 4, label smoothing, threshold 0.30.



Independent Test Performance

Model	Threshold	Accuracy	Precision	Malignant Recall	Specificity	F1 Score	ROC-AUC	Average Precision	TN	FP	FN	TP
MobileNetV2	0.4	0.7187	0.5671	0.8020	0.6744	0.6644	0.8184	0.6882	1243	600	194	786
EfficientNetV2-B2	0.3	0.6699	0.5192	0.6612	0.6744	0.5817	0.7353	0.6082	1243	600	332	648

Flask REST API Implementation

- Framework & Modular DesignTechnology Stack: Built using Python 3 and Flask, establishing a lightweight, high-performance RESTful backend (skin-cancer-detection-backend).Architecture: Implemented a clean separation of concerns across modules—decoupling route controllers, OpenCV image preprocessing pipelines, and model inference interfaces to ensure maintainability and testability.
- Core API EndpointsGET /healthPurpose: Provides a rapid diagnostic route for frontend clients and network monitoring.Response: Returns a 200 OK status with JSON confirming service availability and health status ("status": "healthy").POST /predictPurpose: The primary inference endpoint that ingests raw skin lesion photographs via multipart/form-data.Workflow: Automatically validates file presence, routes the image through the OpenCV preprocessing and quality gate, executes model inference (or mock interface), and returns structured JSON.
- Input Validation & Quality Control IntegrationEarly Rejection Gate: Intercepts uploaded files to verify image integrity before passing data to the neural network.Blur Detection: Calculates Laplacian variance across the image matrix. If the calculated score falls below the strict threshold (\$15.0\$), the API halts processing and immediately returns a 400 Bad Request with diagnostic details ("is_blurry": true, "blur_score").
- Standardized JSON Response Contracts
- Success Scenario (200 OK):

JSON

```
{
```

```
  "status": "success",
  "binary_label": 0,
  "label_name": "Benign",
  "confidence": 0.942,
  "quality_metrics": {
    "blur_score": 84.6,
    "passed_quality_check": true
  }
}
```

- Error Scenario (400 Bad Request):

JSON

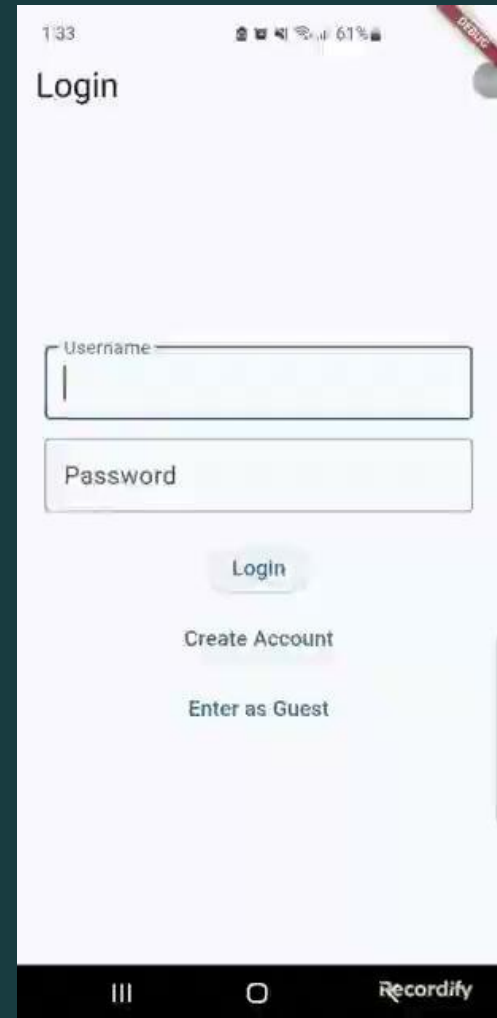
```
{
```

```
  "status": "error",
  "message": "Image quality check failed. Photo is too blurry.",
  "quality_metrics": {
    "blur_score": 8.42,
    "threshold": 15.0
  }
}
```

- End-to-End Testing & VerificationVerified endpoint reliability and payload parsing using native curl.exe commands in PowerShell across both success (clear lesion images) and failure (intentionally blurry test files) scenarios.

User Interface Recap

- **Flutter:** Framework used to develop the mobile application.
- **Android Studio:** Provides the Android development and deployment tools.
- **Dart:** Programming language that supports the Flutter application.
- **SQLite:** Local database used to store user account information.
- **User Interface:** Users can create accounts, log in, and navigate between the application's screens.



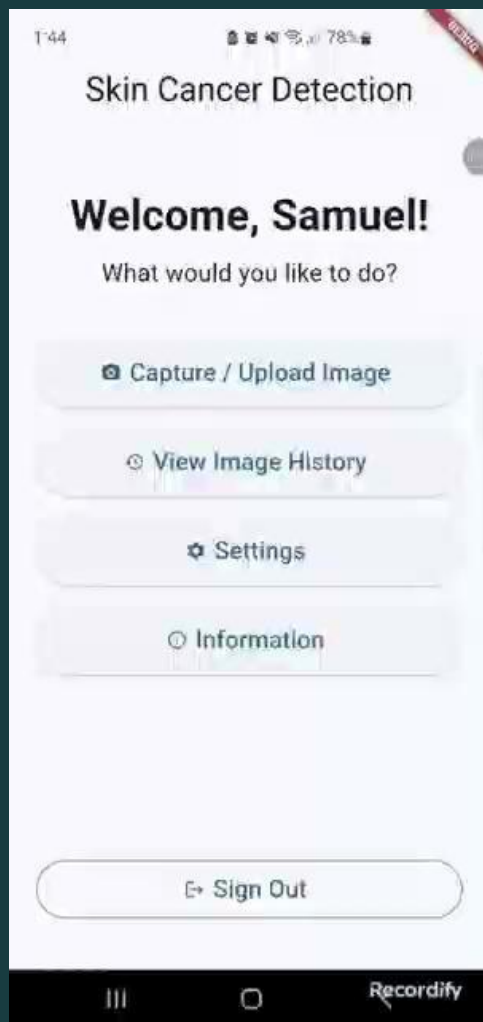
Flask REST API Integration

- **Flask REST API:** Backend server running locally on the computer.
- **Server Port:** Uses port 5000 to receive HTTP requests from the Flutter application.
- **Localhost (Loopback):** Allows the server to be accessed locally, including through ADB reverse over USB.
- **Wi-Fi LAN IP:** Allows the mobile device to communicate wirelessly with the Flask server using the computer's private IP address over the local network (eduroam).
- **Image Upload:** Flutter sends captured photographs to the /predict endpoint using HTTP POST requests.
- **Backend Response:** Flask processes uploaded images and returns JSON data containing image-quality information.

```
(.venv) PS C:\Users\User\Desktop\skin-cancer-detection-backend> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.xxx.xx.xxx:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 860-928-379
10.xxx.xx.xxx -- [28/Oct/2026 14:59:04] "POST /predict HTTP/1.1" 200 -
```

Image Upload Demo

- Image Capture: User can capture, preview, and retake photos using the device's camera.
- Backend Integration: HTTP communication between the Flutter application and Flask REST API.
- Image Quality Feedback: Displays only blur scores, preprocessing status, and upload results for now.



Progress Matrix Milestone 3

Task	Elif	Neva	Samuel	Sophia
Dataset Expansion & Preprocessing Optimization	-	50%	-	50%
MobileNetV2 Optimization & Performance Improvement	-	100%	-	-
CNN Optimization & Comparative Evaluation	100%	-	-	-
ML Model Integration into Flask REST API	-	-	-	100%
Mobile Results Interface & Prediction Display	-	-	100%	-
User Authentication, Database & Analysis History	-	-	100%	-
End-to-End Integration & Functional Testing	50%	-	-	50%
Application Reliability, Security & Performance Testing	50%	-	50%	-

Milestone 3: Planned Tasks & Responsibilities

1. **Dataset Expansion & Preprocessing Optimization — Neva & Sophia:** Expand and refine the dataset, resolve remaining quality issues, and improve preprocessing consistency.
2. **MobileNetV2 Optimization — Neva:** Improve the selected model through fine-tuning, augmentation, and class imbalance handling.
3. **CNN Optimization & Evaluation — Elif:** Refine the baseline CNN and compare its performance against previous models.
4. **ML Model Integration into Flask API — Sophia:** Replace mock inference with the trained model to enable real predictions through the backend.
5. **Mobile Results Interface — Samuel:** Develop a results screen displaying lesion classifications, confidence scores, and image-quality feedback.
6. **User Authentication & Analysis History — Samuel:** Improve authentication and database functionality to store and retrieve previous lesion images and analyses.
7. **End-to-End Integration & Testing — Elif & Sophia:** Test the complete workflow from image capture and upload to model inference and results display.
8. **Reliability, Security & Performance Testing — Elif & Samuel:** Evaluate connection stability, upload performance, and protection of user data.

Thank you!