Image Segmentation Using MobileNetV2

Sachin Maurya
Roll Number: CS24MS002
Email: cs24ms002@iitdh.ac.in

***Abstract*—This project presents a deep learning-based approach for binary image segmentation using MobileNetV2 as the encoder and a custom decoder. Two models were developed and evaluated: Model 1 with a frozen encoder, and Model 2 with a fine-tuned encoder. A dataset of 900 training images and 379 test images, along with their corresponding segmentation masks, was used for training and evaluation. Key performance metrics include Intersection over Union (IoU) and Dice Score. Results show that Model 2 significantly outperforms Model 1 across all metrics, achieving superior accuracy and generalization capabilities.**

I. Introduction

Image segmentation is a key task in computer vision, where each pixel of an image is classified as either part of the foreground (region of interest) or the background. Applications of segmentation span diverse fields, such as medical imaging, autonomous driving, and object detection.

In this work, we leverage MobileNetV2, a lightweight, pre-trained encoder, for feature extraction. A custom decoder is designed to upsample the encoder's output and generate pixel-wise segmentation masks. The models are evaluated using Intersection over Union (IoU) and Dice Score metrics. This study compares two configurations:

- **Model 1:** Frozen encoder with trainable decoder.
- **Model 2:** Fine-tuned encoder with trainable decoder.

The objective is to compare the performance of these models, highlighting the effects of encoder fine-tuning and different loss functions on segmentation performance.

II. Methodology

A. Dataset (ISIC Dataset)

The dataset for the ISIC (International Skin Imaging Collaboration) dataset, which typically consists of skin lesion images and their corresponding binary segmentation masks:

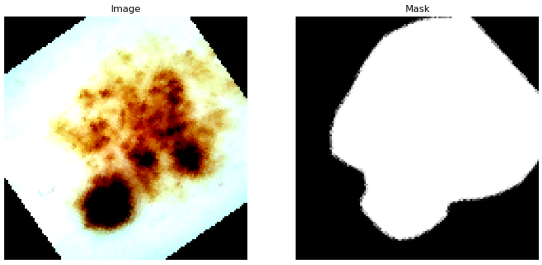


Fig. 1. Visualization of input image and corresponding mask.

- **Training Set:** 900 images resized to 128×128.
- **Testing Set:** 379 images resized to 128×128.
- **Train masks:** Segmented masks for training images.
- **test masks:** Segmented masks for test images.

B. Data Preprocessing

To enhance consistency and improve generalization:

- **Image Preprocessing:** Images are resized to 128×128, normalized using mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$, and augmented with random horizontal flipping and rotations ($\pm 60°$).
- **Mask Preprocessing:** Masks are resized, and augmentations are applied to match the image transformations.

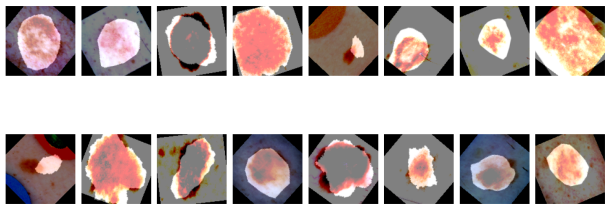


Fig. 2. Visualization of input image and corresponding mask.

C. Model Architecture

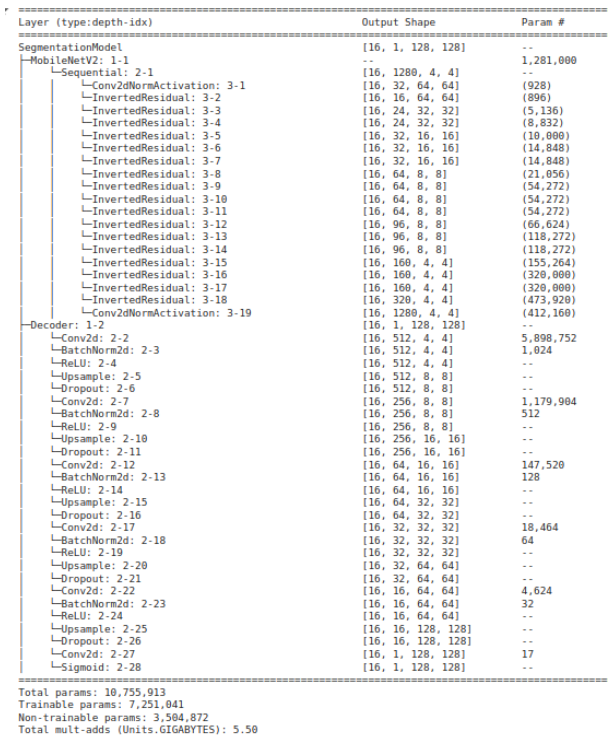
- **The parameters of the Segmentation Model is:**
- Total params: 10,755,913
- Trainable params: 7,251,041 (decoder weights)
- Non-trainable params: 3,504,872 (encoder weights)

1) Encoder: MobileNetV2, pre-trained on ImageNet, outputs feature maps of size $4 \times 4 \times 1280$.

- **Model 1:** Frozen encoder where weights are not updated during training.
- **Model 2:** Fine-tuned encoder where weights are updated during training.

2) Decoder: The decoder reconstructs the feature maps to the input resolution (128×128) and consists of:

- Five convolutional layers to progressively reduce channel sizes ($1280 \rightarrow 512 \rightarrow 256 \rightarrow 64 \rightarrow 32 \rightarrow 16$).
- Bilinear upsampling layers to double spatial dimensions.
- Batch normalization, ReLU activation, and dropout layers for regularization.
- A final 1×1 convolution with sigmoid activation for binary mask generation.



```
==========================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
SegmentationModel                                       [16, 1, 128, 128]         --
├─MobileNetV2: 1-1                                      --                        1,281,000
│    └─Sequential: 2-1                                  [16, 1280, 4, 4]          --
│    │    └─Conv2dNormActivation: 3-1                   [16, 32, 64, 64]          (928)
│    │    └─InvertedResidual: 3-2                       [16, 16, 64, 64]          (896)
│    │    └─InvertedResidual: 3-3                       [16, 24, 32, 32]          (5,136)
│    │    └─InvertedResidual: 3-4                       [16, 24, 32, 32]          (8,832)
│    │    └─InvertedResidual: 3-5                       [16, 32, 16, 16]          (10,000)
│    │    └─InvertedResidual: 3-6                       [16, 32, 16, 16]          (14,848)
│    │    └─InvertedResidual: 3-7                       [16, 32, 16, 16]          (14,848)
│    │    └─InvertedResidual: 3-8                       [16, 64, 8, 8]            (21,056)
│    │    └─InvertedResidual: 3-9                       [16, 64, 8, 8]            (54,272)
│    │    └─InvertedResidual: 3-10                      [16, 64, 8, 8]            (54,272)
│    │    └─InvertedResidual: 3-11                      [16, 64, 8, 8]            (54,272)
│    │    └─InvertedResidual: 3-12                      [16, 96, 8, 8]            (66,624)
│    │    └─InvertedResidual: 3-13                      [16, 96, 8, 8]            (118,272)
│    │    └─InvertedResidual: 3-14                      [16, 96, 8, 8]            (118,272)
│    │    └─InvertedResidual: 3-15                      [16, 160, 4, 4]           (155,264)
│    │    └─InvertedResidual: 3-16                      [16, 160, 4, 4]           (320,000)
│    │    └─InvertedResidual: 3-17                      [16, 160, 4, 4]           (320,000)
│    │    └─InvertedResidual: 3-18                      [16, 320, 4, 4]           (473,920)
│    │    └─Conv2dNormActivation: 3-19                  [16, 1280, 4, 4]          (412,160)
├─Decoder: 1-2                                          [16, 1, 128, 128]         --
│    └─Conv2d: 2-2                                      [16, 512, 4, 4]           5,898,752
│    └─BatchNorm2d: 2-3                                 [16, 512, 4, 4]           1,024
│    └─ReLU: 2-4                                        [16, 512, 4, 4]           --
│    └─Upsample: 2-5                                    [16, 512, 8, 8]           --
│    └─Dropout: 2-6                                     [16, 512, 8, 8]           --
│    └─Conv2d: 2-7                                      [16, 256, 8, 8]           1,179,904
│    └─BatchNorm2d: 2-8                                 [16, 256, 8, 8]           512
│    └─ReLU: 2-9                                        [16, 256, 8, 8]           --
│    └─Upsample: 2-10                                   [16, 256, 16, 16]         --
│    └─Dropout: 2-11                                    [16, 256, 16, 16]         --
│    └─Conv2d: 2-12                                     [16, 64, 16, 16]          147,520
│    └─BatchNorm2d: 2-13                                [16, 64, 16, 16]          128
│    └─ReLU: 2-14                                       [16, 64, 16, 16]          --
│    └─Upsample: 2-15                                   [16, 64, 32, 32]          --
│    └─Dropout: 2-16                                    [16, 64, 32, 32]          --
│    └─Conv2d: 2-17                                     [16, 32, 32, 32]          18,464
│    └─BatchNorm2d: 2-18                                [16, 32, 32, 32]          64
│    └─ReLU: 2-19                                       [16, 32, 32, 32]          --
│    └─Upsample: 2-20                                   [16, 32, 64, 64]          --
│    └─Dropout: 2-21                                    [16, 32, 64, 64]          --
│    └─Conv2d: 2-22                                     [16, 16, 64, 64]          4,624
│    └─BatchNorm2d: 2-23                                [16, 16, 64, 64]          32
│    └─ReLU: 2-24                                       [16, 16, 64, 64]          --
│    └─Upsample: 2-25                                   [16, 16, 128, 128]        --
│    └─Dropout: 2-26                                    [16, 16, 128, 128]        --
│    └─Conv2d: 2-27                                     [16, 1, 128, 128]         17
│    └─Sigmoid: 2-28                                    [16, 1, 128, 128]         --
==========================================================================================
Total params: 10,755,913
Trainable params: 7,251,041
Non-trainable params: 3,504,872
Total mult-adds (Units.GIGABYTES): 5.50
```

Fig. 3. summary of the model.

D. Loss Functions

1) Model 1: Combined Dice Loss and Binary Cross-Entropy (BCE) Loss: The combined loss function is defined as:

$$\text{Loss} = 0.1 \cdot \text{Dice Loss} + \text{BCE Loss} \tag{1}$$

Dice Loss emphasizes the overlap between predicted and ground truth masks:

$$\text{Dice Loss} = 1 - \frac{2 \cdot |\text{Intersection}|}{|\text{Predicted}| + |\text{Ground Truth}|} \tag{2}$$

Terms:

- $|\text{Intersection}|$: Number of overlapping pixels between the predicted and ground truth masks.
- $|\text{Predicted}|$: Total number of pixels in the predicted mask.
- $|\text{Ground Truth}|$: Total number of pixels in the ground truth mask.

BCE Loss penalizes pixel-wise prediction errors:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}[G_i \cdot \log(P_i) + (1 - G_i) \cdot \log(1 - P_i)] \tag{3}$$

Terms:

- N: Total number of pixels in the image.
- G_i: Ground truth value for pixel i (0 for background, 1 for foreground).
- P_i: Predicted probability for pixel i.

2) Model 2: Binary Cross-Entropy (BCE) Loss: The BCE Loss is defined as:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}[G_i \cdot \log(P_i) + (1 - G_i) \cdot \log(1 - P_i)] \tag{4}$$

E. Evaluation Metrics

1) Intersection over Union (IoU):

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{5}$$

Terms:

- Area of Overlap: Number of pixels where both predicted and ground truth masks are 1.
- Area of Union: Total number of pixels where either the predicted or ground truth mask is 1.

2) Dice Score:

$$\text{Dice} = \frac{2 \cdot |\text{Intersection}|}{|\text{Predicted}| + |\text{Ground Truth}|} \tag{6}$$

—

III. Results and Analysis

A. Model 1: Frozen Encoder with Combined Loss

Model 1 uses a frozen MobileNetV2 encoder and a combined loss function consisting of Dice Loss and Binary Cross-Entropy (BCE) Loss. The following results were observed:

- **IoU:** 0.6354
- **Dice Score:** 0.7675
- **Train Loss:** 0.2911
- **Validation Loss:** 0.2899

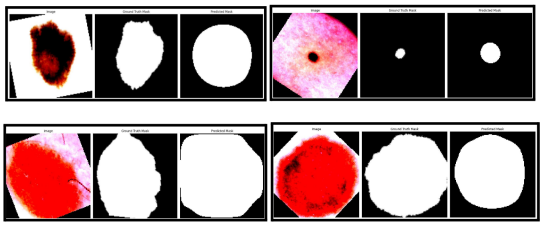


Fig. 4. Visualization of input image, ground truth mask, and predicted mask for Model 1.

Fig. 5. The training and validation loss curves for Model 1.

B. Model 2: Fine-Tuned Encoder with BCE Loss

Model 2 uses a fine-tuned MobileNetV2 encoder and Binary Cross-Entropy (BCE) Loss. The following results were observed:

- **IoU:** 0.6819
- **Dice Score:** 0.7972
- **Train Loss:** 0.2117
- **Validation Loss:** 0.2169

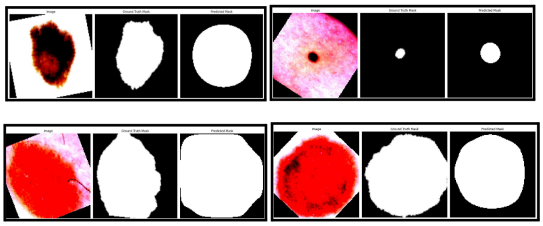


Fig. 6. Visualization of input image, ground truth mask, and predicted mask for Model 2.

Fig. 7. the training and validation loss curves for Model 2.

IV. Comparative Analysis

A. Overview of Models

Two experiments were conducted:

- **Model 1:** Encoder Frozen (only decoder weights updated during training).
- **Model 2:** Fine-tuned Model (both encoder and decoder weights updated during training).

TABLE I
Comparison of Metrics for Model 1 and Model 2

Metric	Model 1	Model 2
IoU	0.6354	0.6819
Dice Score	0.7675	0.7972
Train Loss	0.2911	0.2117
Validation Loss	0.2899	0.2169

B. Bar Plots

Figure 8 first row shows :

- The first row plot left side displays the metrics (IoU, Dice Score, Train Loss, and Validation Loss) for Model 1.
- The first row plot right side the same metrics for Model 2.
- **Comparison :** Figure 8 Second row shows compares the two models side by side:
  - The Second row plot left side displays the metrics (IoU and Dice Score.)
  - The Second row plot right side displays the metrics (Train Loss and Validation Loss.)

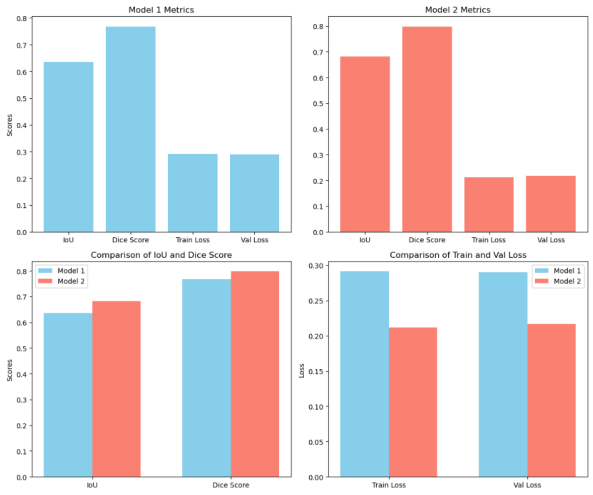


Fig. 8. Bar plots comparing metrics for Model 1 and Model 2.

C. Radar Chart

A radar chart, shown in Figure 9, visualizes the overall performance of both models across all metrics:

- Each model's performance is represented by a polygon, where each vertex corresponds to a different metric.
- Model 1's polygon is filled with light blue, while Model 2's polygon is filled with light salmon.

This chart provides a quick visual comparison of how well each model performs across the different metrics.

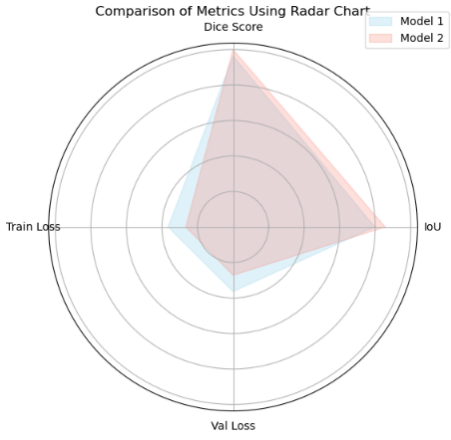


Fig. 9. Radar chart visualizing the performance of Model 1 and Model 2 across all metrics.

D. Analysis of Metrics

- **IoU (Intersection over Union):** Model 2 (0.6819) outperforms Model 1 (0.6354). A higher IoU indicates better overlap between predicted and ground truth masks, demonstrating Model 2's superior segmentation accuracy.
- **Dice Score:** Model 2 (0.7972) achieves a higher Dice Score compared to Model 1 (0.7675). This indicates better spatial overlap for Model 2.
- **Train Loss:** Model 1 (0.2911) has a higher training loss compared to Model 2 (0.2169). A lower training loss suggests that Model 2 has better training convergence.
- **Validation Loss:** Model 2 (0.2169) exhibits a lower validation loss than Model 1 (0.2899), indicating better generalization on unseen data.

V. Conclusion

This study demonstrates that fine-tuning the MobileNetV2 encoder with BCE Loss (Model 2) achieves superior segmentation performance compared to using a frozen encoder with a combined Dice and BCE Loss (Model 1). Model 2 consistently delivers better results in terms of segmentation accuracy, training convergence, and generalization ability.

Future work will focus on:

- Exploring advanced loss functions, to enhance segmentation performance further.
- Incorporating multi-scale feature aggregation techniques to capture finer details in segmentation tasks.

References

[1] A. Kanadath, J. A. Arul Jothi, and S. Urolagin, "Histopathology Image Segmentation Using MobileNetV2 based U-net Model," *2021 International Conference on Intelligent Technologies (CONIT)*, Karnataka, India, June 25–27, 2021.

[2] G. Du, X. Cao, J. Liang, X. Chen, and Y. Zhan, "Medical Image Segmentation based on U-Net: A Review," *Journal of Imaging Science & Technology*, vol. 64, no. 2, pp. 1–12, 2020.