Enhanced Skin Lesion Segmentation using MobileNet-Encoder

Objective: train a segmentation model using a MobileNet pre-trained on the ImageNet dataset as an encoder and design a decoder that predicts segmented masks.

https://colab.research.google.com/drive/1CWbaEJkdCz84hOXMuBxyDIOZTZESOpmY?usp=sharing

DataSet:

- The ISIC 2016 dataset is a collection of dermoscopic images (close-up skin lesion images) used for skin cancer detection tasks.
- It was created as part of the 2016 ISBI (International Symposium on Biomedical Imaging) challenge hosted by the International Skin Imaging Collaboration (ISIC).
- The dataset includes:
    - train: 900 training images.
    - train masks: Segmented masks for training images.
    - test: 379 test images.
    - test masks: Segmented masks for test images.

Pre - Processing:

- Data augmentation is a cornerstone technique in our pre-processing pipeline, aimed at artificially expanding the dataset to counter overfitting and improve model robustness. Given the variability in skin lesion appearances and imaging conditions, the following augmentation strategies were employed:
- Resizing: All images and masks were resized to 128x128 pixels, ensuring a consistent input size for the neural network, which is crucial for batch processing and model architecture compatibility.
- Flips: Horizontal and vertical flips were applied with a probability of 0.5. This step simulates the possible orientations of lesions on the skin, making our model more adaptable to different lesion positions.
- Random Brightness and Contrast Adjustment: Adjusting the brightness and contrast randomly within a predefined range helps the model become resilient to variations in lighting conditions that commonly occur in clinical settings.

- Normalization: The images were normalized using mean values of [0.485, 0.456, 0.406] and standard deviations of [0.229, 0.224, 0.225]. These values are standard for models pre-trained on the ImageNet dataset, aligning the data distribution with that expected by MobileNet's pre-trained layers, thus facilitating more effective transfer learning.
- Through strategic pre-processing steps, including data augmentation and normalization, we have successfully enhanced the ISIC dataset's utility for the task of skin lesion segmentation. These steps not only standardized the dataset for efficient model training but also introduced necessary variations that prepare the model for real-world application.

Experiment 6.1: Feature Extraction:

- We have used a pre-trained MobileNet encoder trained on ImageNetV1.
- We have designed a custom decoder atop this encoder for the segmentation task. And trained the decoder while keeping the encoder frozen.

Decoder Architecture:

- Decoder incrementally upsamples the feature maps reduced by the encoder, reintegrates spatial information through skip connections, and ultimately generates a high-resolution segmentation map.

Upsampling and Feature Integration

Initial Upsampling:

- The decoder begins by addressing the most compressed feature map provided by the encoder (from stage4, with 1280 channels). It uses a transposed convolution (ConvTranspose2d) to double the spatial dimensions (upsampling), reducing the channels to match the next level of feature integration.

Feature Integration via Skip Connections:

- Skip Connection from Stage 3: The upsampled features are then concatenated with the features directly from Stage 3 of the encoder. This reintroduces spatial information that was lost during the downsampling process in the encoder. The combined features have both high-level semantic information and spatial details.
- Further Upsampling: This process is repeated for Stages 2 and 1. Each time, the features are upsampled and then concatenated with features from the earlier stage of the encoder, reintroducing even more spatial details lost during initial downsampling.

Sequential Processing Through Blocks

- The decoder processes the features through a series of DilatedConvBlocks, each consisting of a convolutional layer followed by batch normalization and ReLU activation. These blocks refine the features at each resolution, improving the quality of the segmentation map.

Final Steps to Segmentation Map

Last Upsampling:

- After the sequential processing through dilated convolution blocks, the decoder performs a final upsampling to ensure that the output matches the target resolution of the input images (128x128 pixels in your case).

- Final Convolution: A convolutional layer with a kernel size of 1x1 projects the features onto the desired number of classes (it's set for binary segmentation, hence num_classes=1). This layer produces the final segmentation map.

- Optional Upsampling to Target Size: The decoder includes an additional upsampling step (Upsample layer) to precisely adjust the output size to the expected target resolution. This step ensures compatibility across different input sizes and maintains consistency with the ground truth masks for evaluation.

Loss Function:

- Our model employs a hybrid loss function, DiceBCE Loss, which amalgamates the strengths of Dice Loss and BCE Loss, optimizing the model for both class balance and segmentation precision.
- This combination, alongside the utilization of IoU and the Dice Coefficient as performance metrics, ensures a comprehensive and nuanced approach to model training and evaluation.
- Through this sophisticated loss function and evaluation metrics, our segmentation model achieves remarkable accuracy, adeptly navigating the challenges posed by class imbalance and the intricate demands of pixel-wise segmentation tasks.

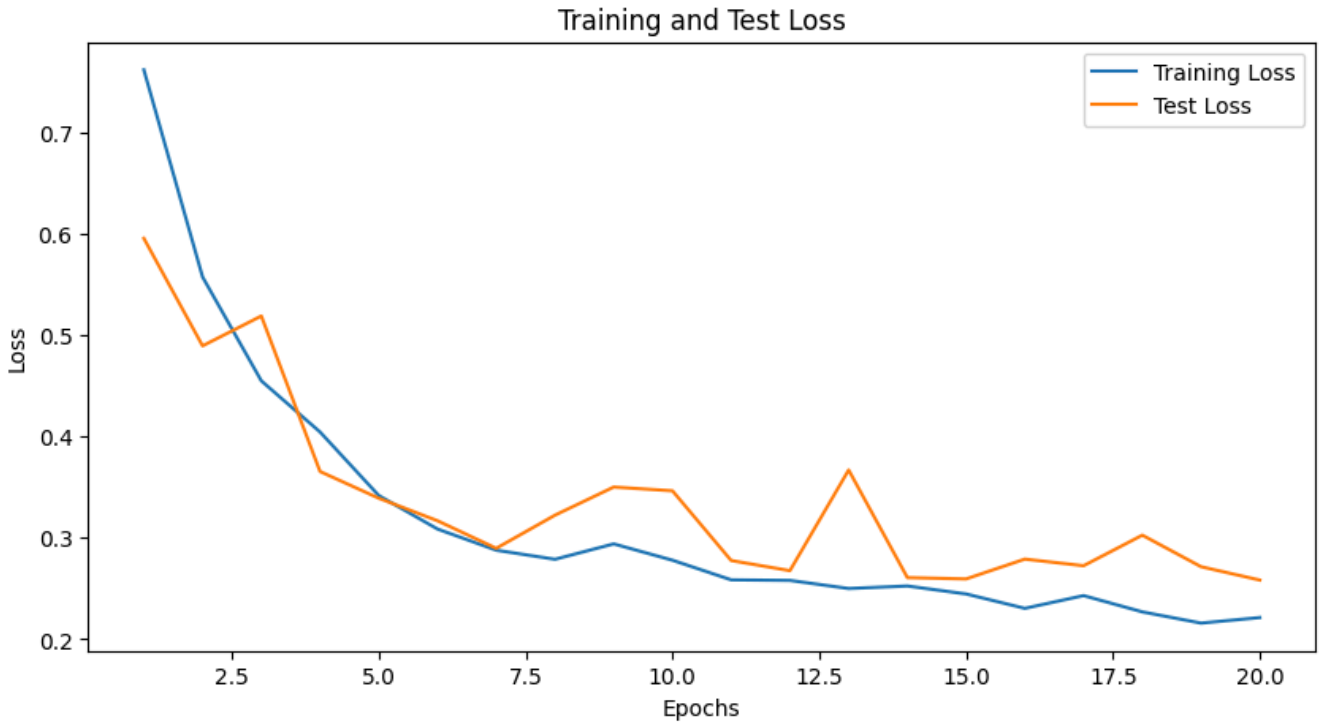
Results and Analysis:

Intersection over Union (IoU) and Dice score:

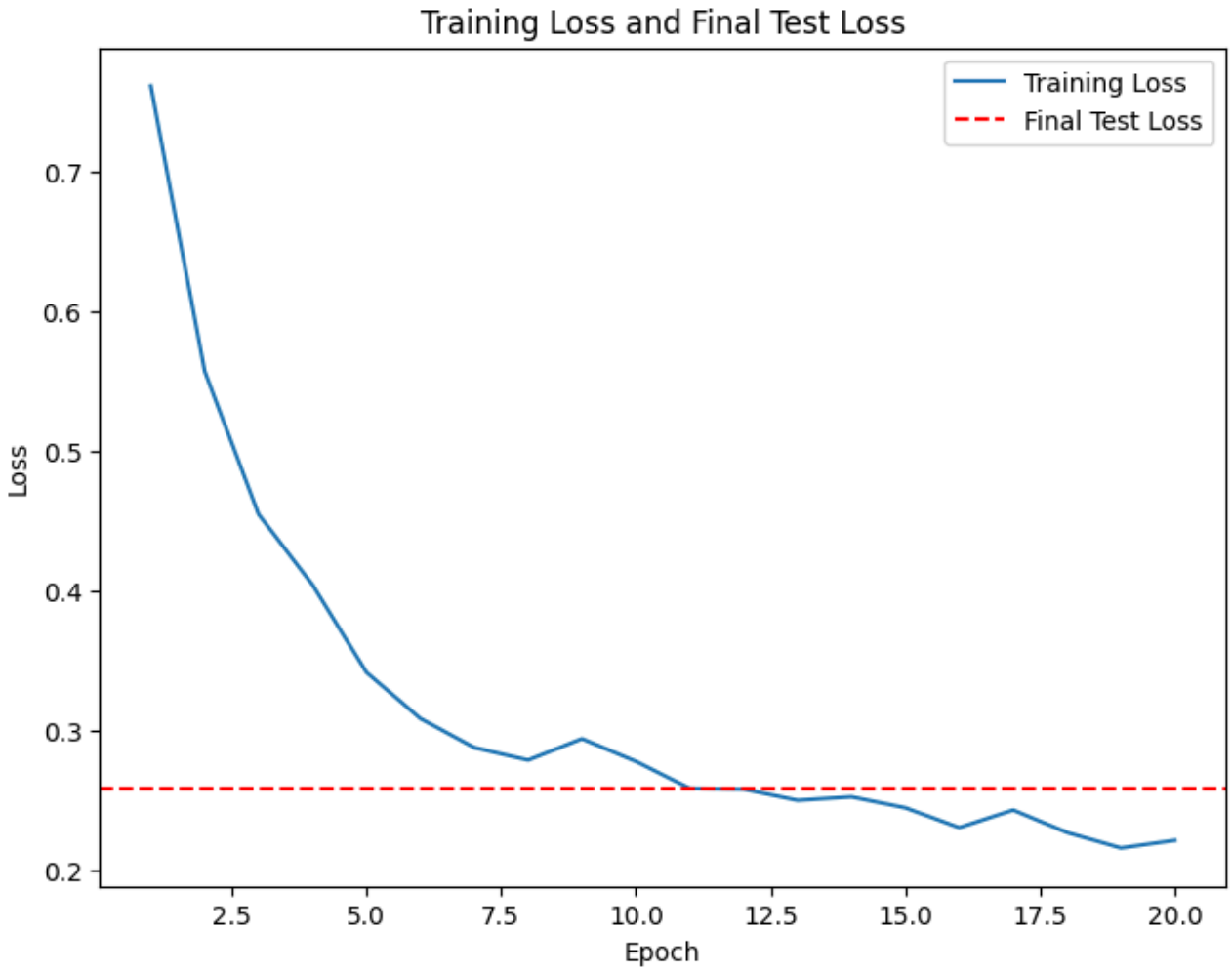
- **Test Loss: 0.2583,**
- **IoU: 0.8716**
- **Dice: 0.9023**

Training and Validation Loss Plots:

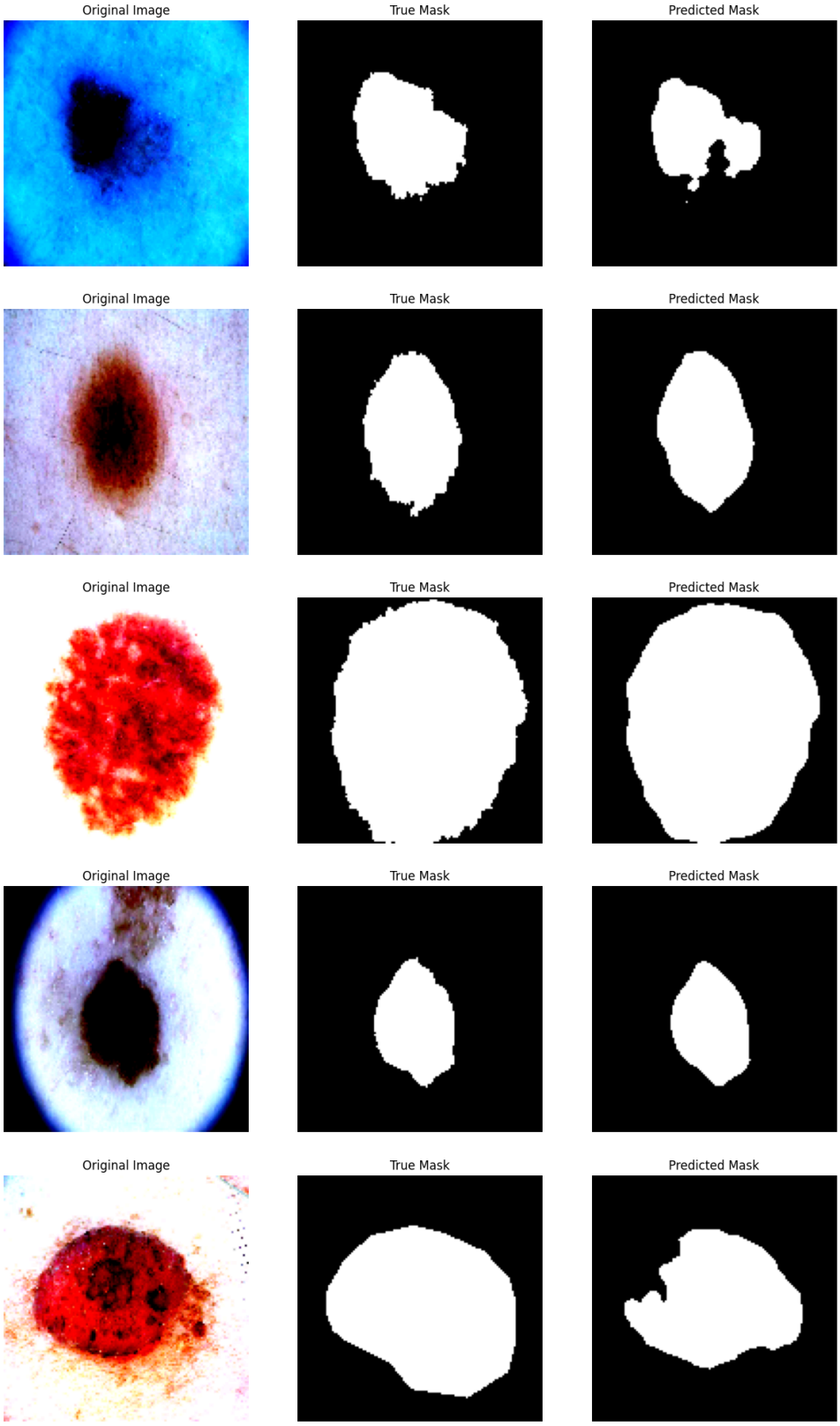
- With Training



- Post Training

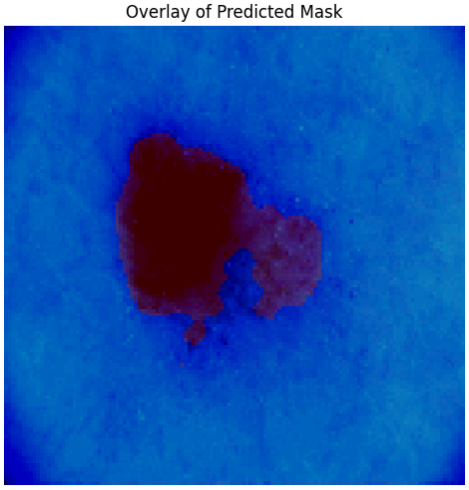


Visualizing samples alongside their generated masks and ground truth:

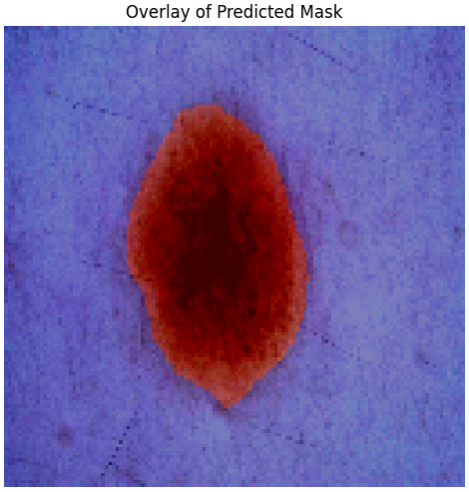


Visualizing Overlay of Predicted Masks:

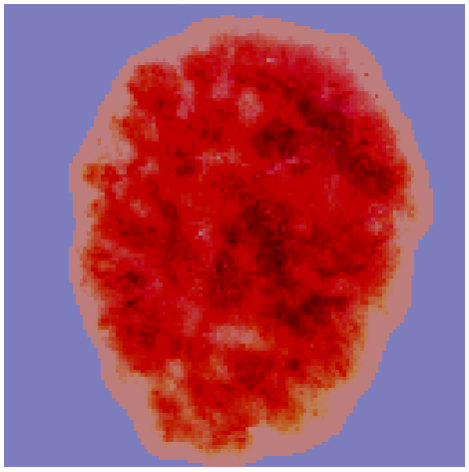
Overlay of Predicted Mask



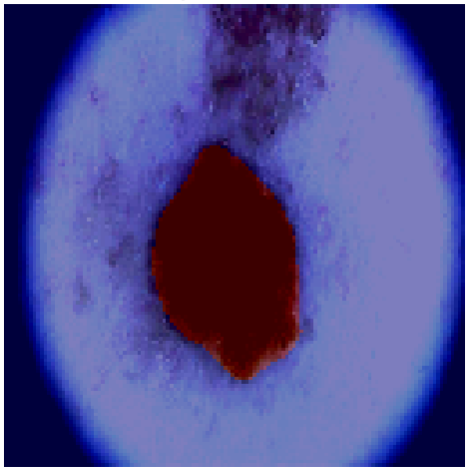
Overlay of Predicted Mask



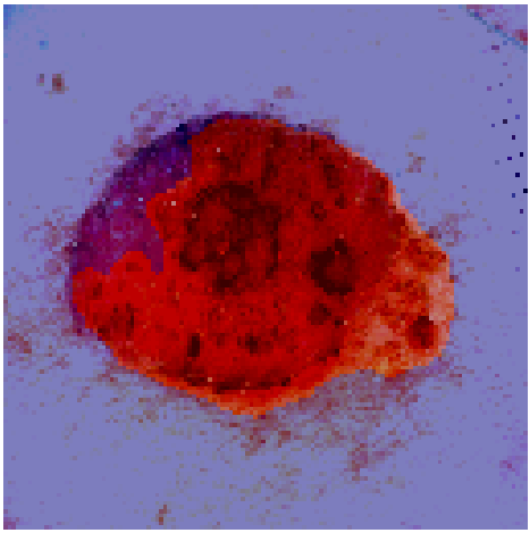
Overlay of Predicted Mask



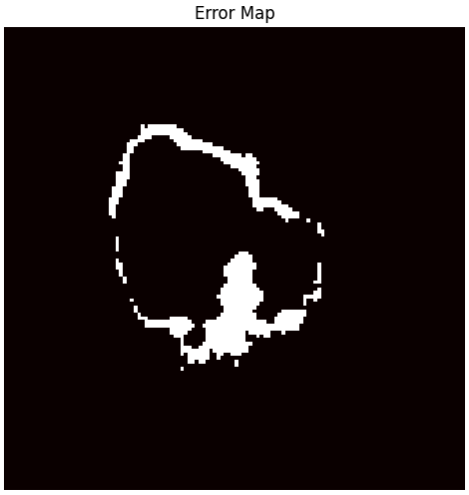
Overlay of Predicted Mask

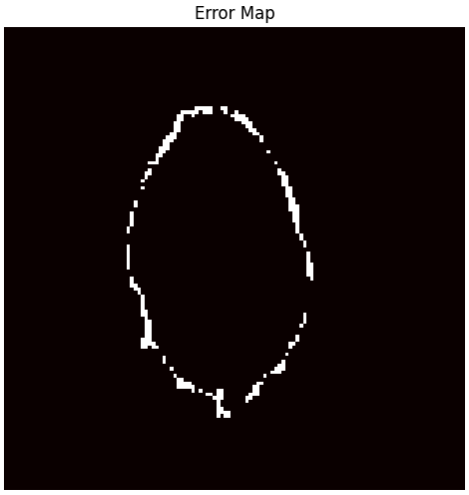


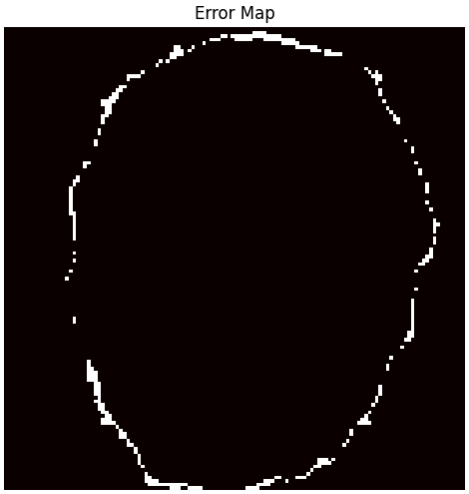
Overlay of Predicted Mask

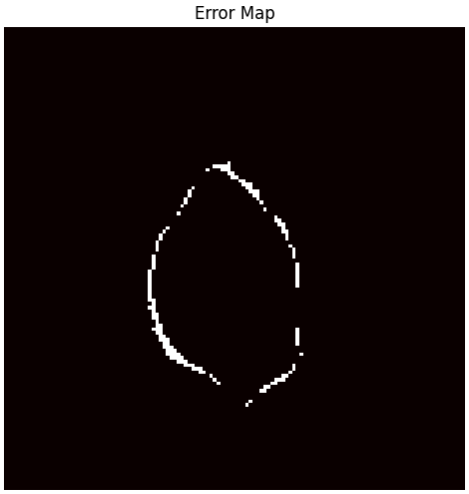


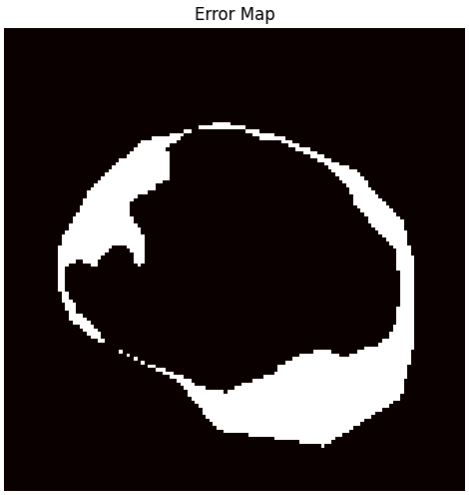
Visualizing Error Maps:











Result Analysis:

The above results were obtained while keeping the encoder layers freezed and using skip connections in our architecture(which we described earlier)

- Test Loss: 0.2583:
    - Indicates the model's predictions are closely aligned with the actual data.
    - Suggests high accuracy in segmenting images but highlights room for further optimization.

- Intersection over Union (IoU): 0.8716
    - Demonstrates the model's proficiency in accurately segmenting targeted objects within images.
    - A high IoU score reflects the effective identification of regions of interest.

- Dice Score: 0.9023
    - Shows the model's predictions are highly congruent with the actual shapes and boundaries of segmented objects.
    - Similar to IoU but provides a slightly different perspective on overlap and accuracy.

Experiment 6.2: FineTuning:

Following 6.1, we commenced with saving our model's data and the initialization of a new model reflecting its current state.

Subsequently, we engaged in the training of the segmentation model, implementing a fine-tuning of the encoder weights through the unfreezing of the encoder's stage 4 parameters.

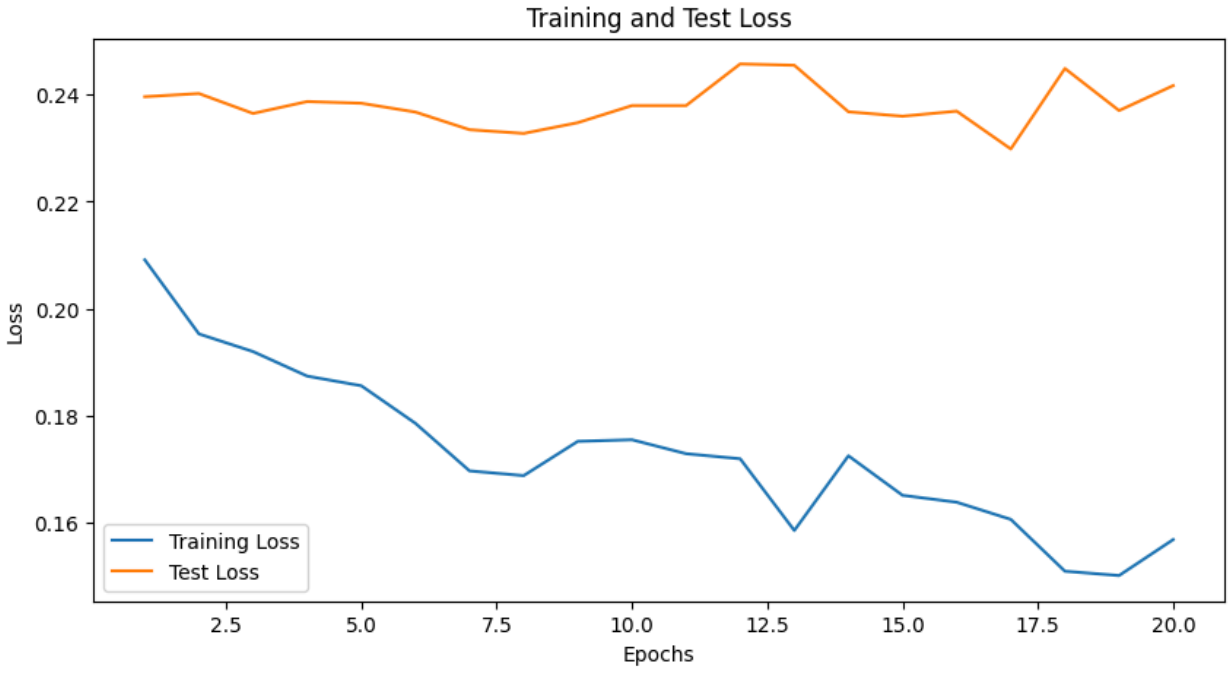
Results and Analysis:

Intersection over Union (IoU) and Dice score:

- **Test Loss: 0.2417**
- **IoU: 0.8799**
- **Dice: 0.9089**

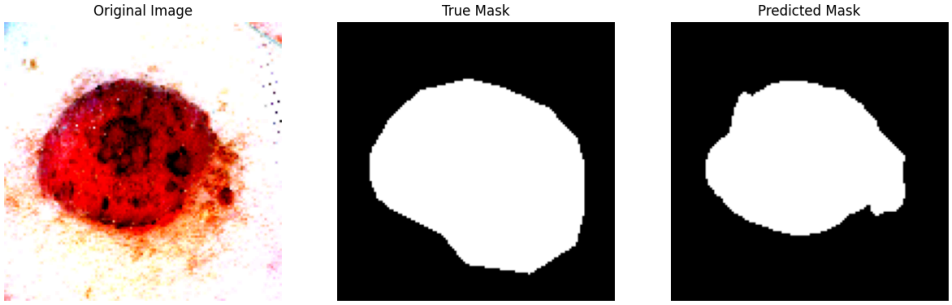
Training and Validation Loss Plots:

- With Training



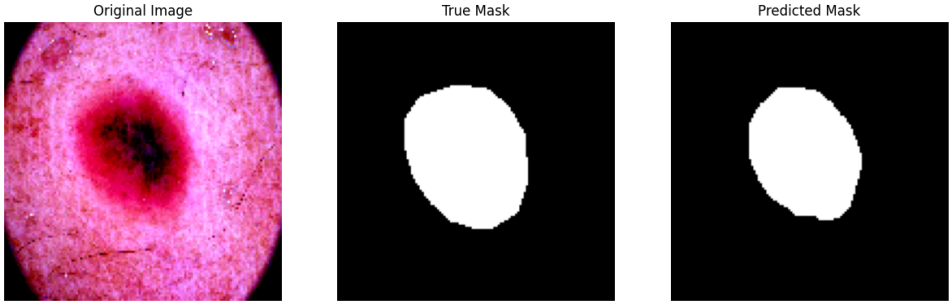
- Post Training

Visualizing samples alongside their generated masks and ground truth:

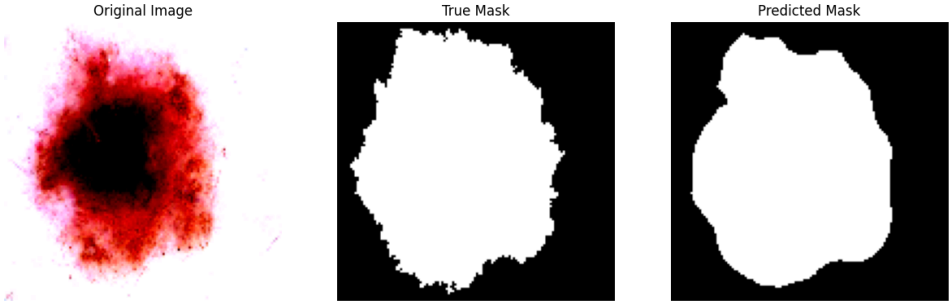


Original Image | True Mask | Predicted Mask

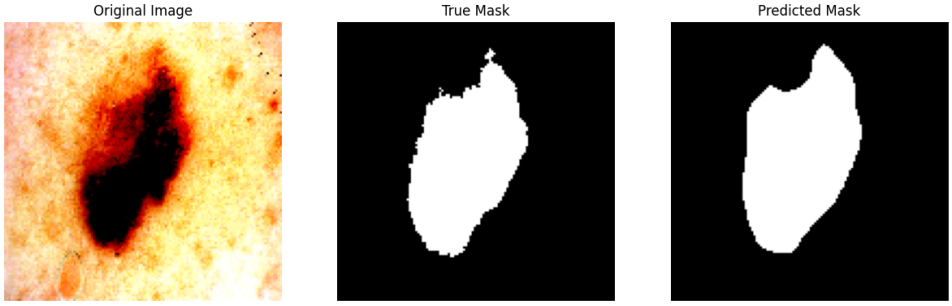
Original Image | True Mask | Predicted Mask



Original Image | True Mask | Predicted Mask

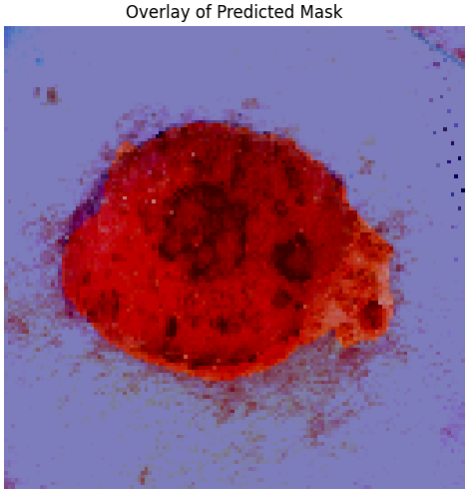


Original Image | True Mask | Predicted Mask

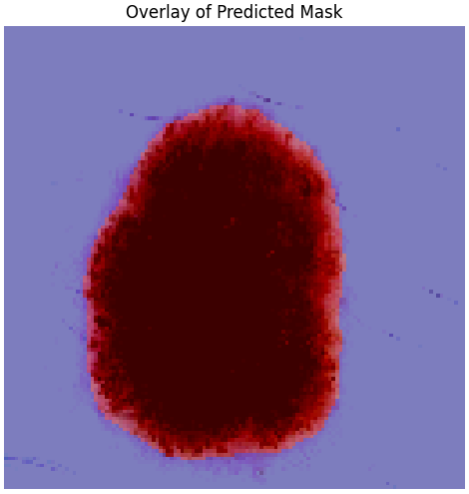


Visualizing Overlay of Predicted Masks:

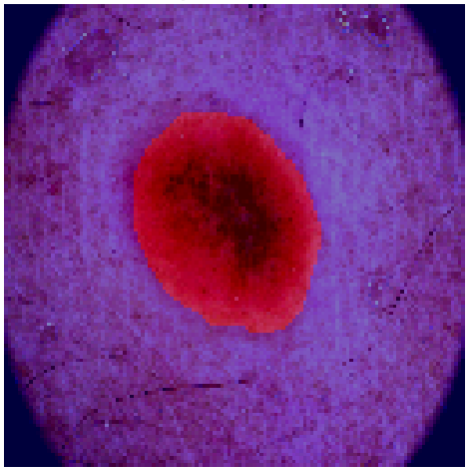
Overlay of Predicted Mask



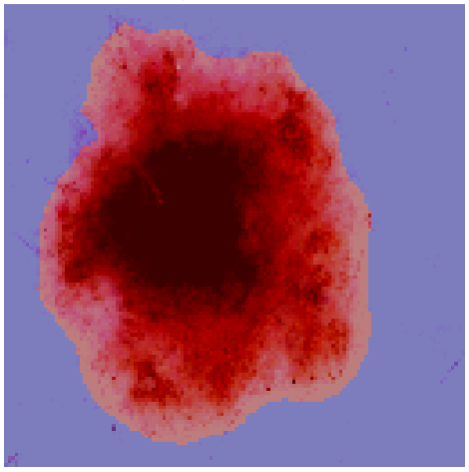
Overlay of Predicted Mask



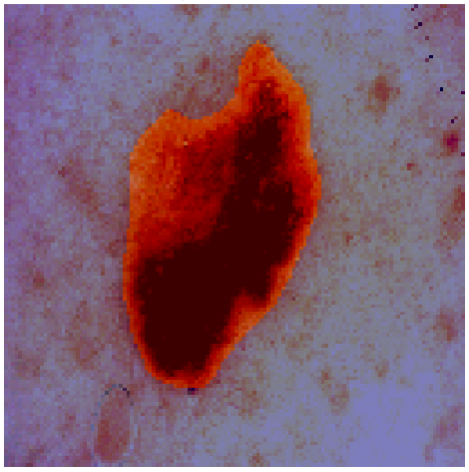
Overlay of Predicted Mask



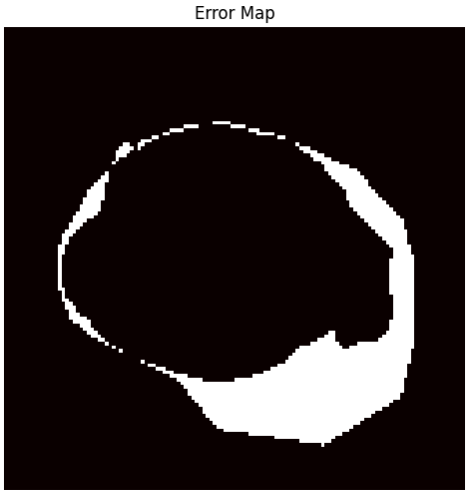
Overlay of Predicted Mask

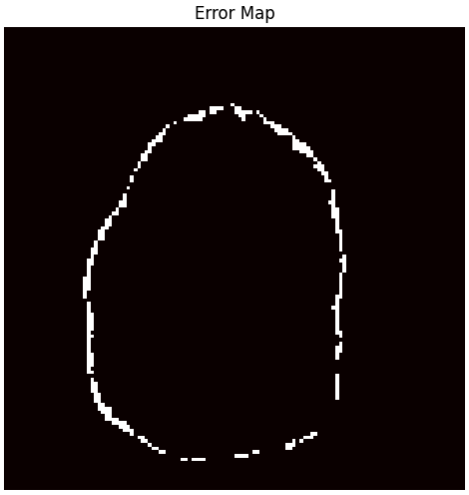


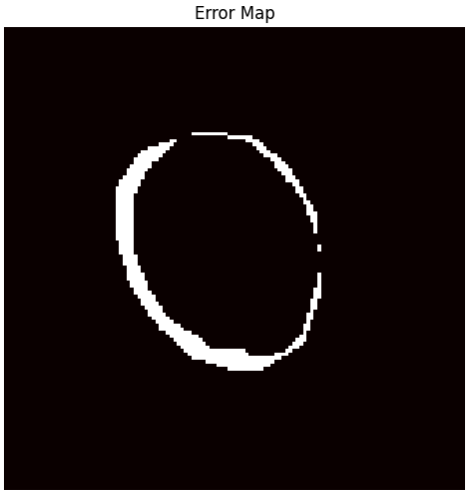
Overlay of Predicted Mask

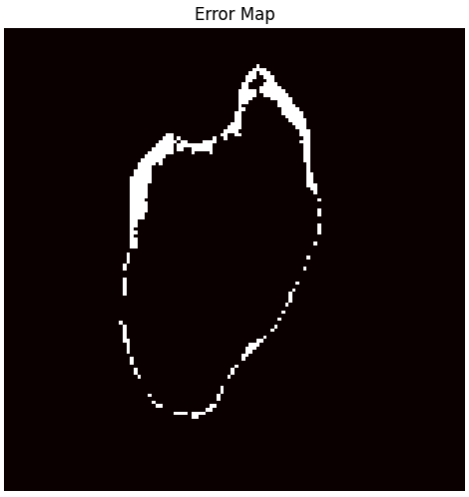


Visualizing Error Maps:









Result Analysis:

The above results were obtained while keeping the encoder layers freezed except stage 4 and using skip connections in our architecture(which we described earlier)

- Fine-tuning, specifically the unfreezing of the encoder's last stage, has directly contributed to the model's enhanced segmentation accuracy.
- The adjustments allow the model to leverage more detailed features, significantly improving segmentation precision.
- We also experimented with unfreezing 2 stages and even unfreezing all the layers of encoder but we observed that the best performance was given by unfreezing all layers as expected.
- But improvement gain is not that significant if we consider the number of parameters involved.

Comparative Analysis:

Model	Learning Rate	Test Loss	IOU	DICE
Without Fine Tune	0.001	0.2583	0.8716	0.9023
With Fine Tune(step 4)	0.0001	0.2417	0.8799	0.9089
With Fine Tune (all layers)	0.0001	0.2366	0.8834	0.9121

Parameter Analysis:

Without Fine tune:

Parameter	**Value**
Total parameters	6,995,457
Trainable parameters	4,771,585
Non-trainable parameters	22,23872

With Fine tune:

Parameter Type	Value
Total parameters	6995457
Trainable parameters	6452929
Non-trainable parameters	542528

Appendix A.1: Decoder Architecture

```
decoder: AdvancedDecoder
decoder.up1: ConvTranspose2d
decoder.conv1: DilatedConvBlock
decoder.conv1.conv: Conv2d
decoder.conv1.bn: BatchNorm2d
decoder.conv1.relu: ReLU
decoder.up2: ConvTranspose2d
decoder.conv2: DilatedConvBlock
decoder.conv2.conv: Conv2d
decoder.conv2.bn: BatchNorm2d
decoder.conv2.relu: ReLU
decoder.up3: ConvTranspose2d
decoder.conv3: DilatedConvBlock
decoder.conv3.conv: Conv2d
decoder.conv3.bn: BatchNorm2d
decoder.conv3.relu: ReLU
decoder.final_conv: Conv2d
decoder.upsample_to_target: Upsample
```