AMRESH KUMAR
M23CSA004
ASSIGNMENT_03

Code Link:https://colab.research.google.com/drive/1ggLn1toE6rZMyH42hvUhxgn8h3HIFYiW?usp=sharing

Experiment 1: Feature Extraction (Frozen Encoder)

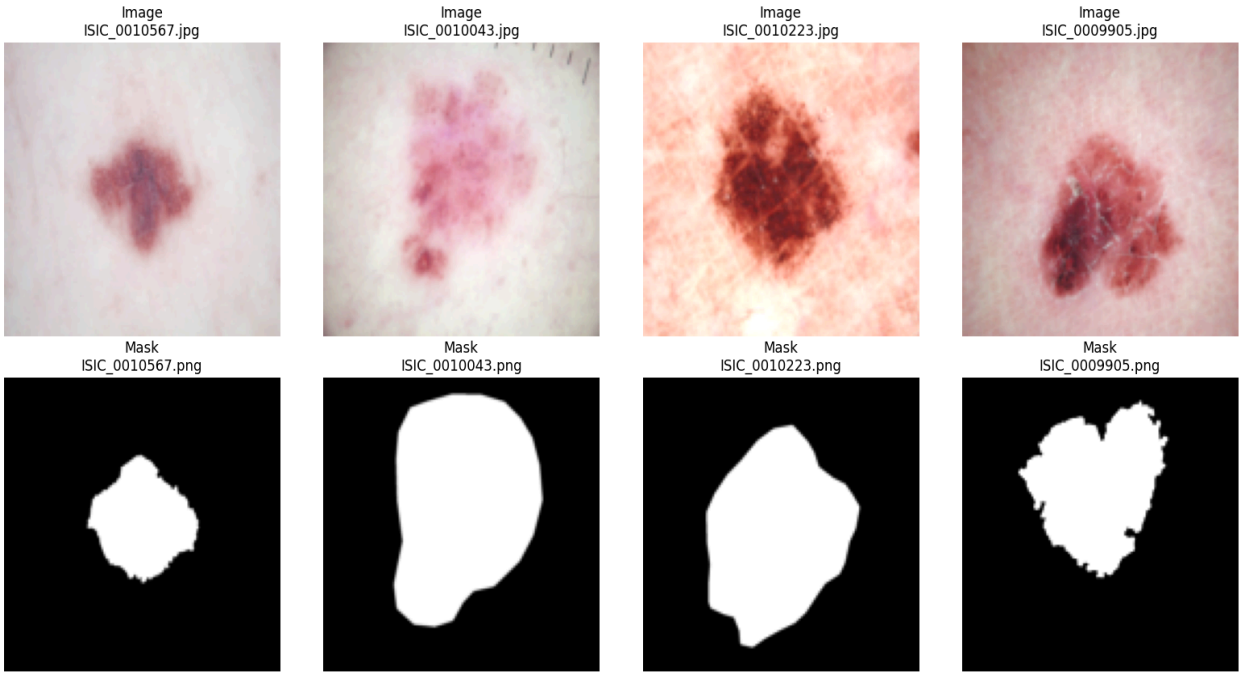
Methods
Dataset Pre-processing(Same for both Experiments):

Loaded the ISIC 2016 dataset.
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.

Original Image Shape and Rationale for Resizing:
Sample Original Image Shapes:
Original Image Dimensions: **1022 x 767**
Original Image Dimensions: **2272 x 1704**
Rationale for Resizing:
It can be seen that the images are of different sizes, so , we need to resize ,
Also imagenet is pretrained on 128*128 image sizes.
Resizing the images to a smaller size (128x128) helps in reducing computational complexity during training while still preserving essential features.

Pre-processed the dataset by resizing images to 128x128 and applying **data augmentation** techniques like **random horizontal** and **vertical flips**, as well as **color jittering.**
Created **custom dataloaders** for both training and test datasets.

Original Image and Mask Visualization:



Sample original images and their corresponding masks from the dataset were visualized.

Model Architecture:

Utilized a pre-trained MobileNet encoder trained on ImageNetV1.
Designed a custom decoder on top of the MobileNet encoder for the segmentation task.
The custom decoder is designed to take feature maps from the encoder and perform segmentation by gradually upsampling and reducing the number of channels until reaching the desired number of classes.

Input: Feature maps from the encoder with input_channels channels = 1280.
Output: Segmentation mask with num_classes channels.
Layers:
Convolutional Layer (conv1):
Input: 1280
Output: 128 channels
Kernel Size: 3x3
Padding: 1
Activation Function: ReLU
Convolutional Layer (conv2):
Input: 128 channels
Output: 1
Kernel Size: 1x1
Upsampling Layer (upsample):
Upsample to the desired output size (128x128) using bilinear interpolation.

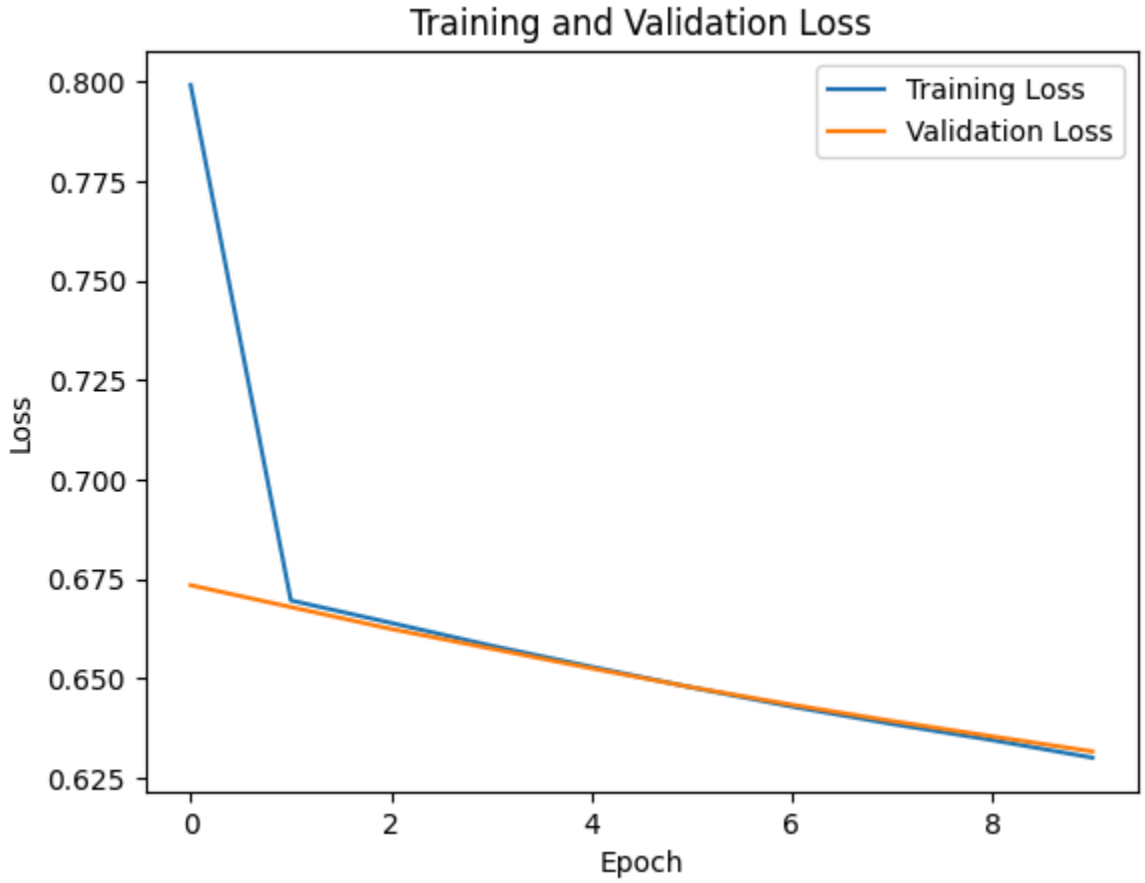
Froze the encoder parameters to prevent them from being updated during training.
Defined a custom loss function using BCEWithLogitsLoss and utilized Adam optimizer for optimization.

Training and Evaluation:

Trained the decoder while keeping the encoder frozen for 10 epochs.
Monitored training and validation loss during each epoch.
Evaluated the model on the test dataset to calculate Intersection over Union (IoU) and Dice score.
Visualized several samples alongside their generated masks and ground truth for qualitative analysis.(BUT was not able to show predicted mask so i have not shown here)

Results:

Intersection over Union (IoU): **0.2862**(This comes when i run only for 1 epoch)
Dice Score: **0.4105**(This comes when i run only for 1 epoch)
NOTE: On running for 10 or 5 epoch i am able to get values as zero
Loss Curves:(Calculated for 10 epoch and working fine)



Training Loss decreased over epochs.
Validation Loss slightly increased after a certain number of epochs, indicating potential overfitting.

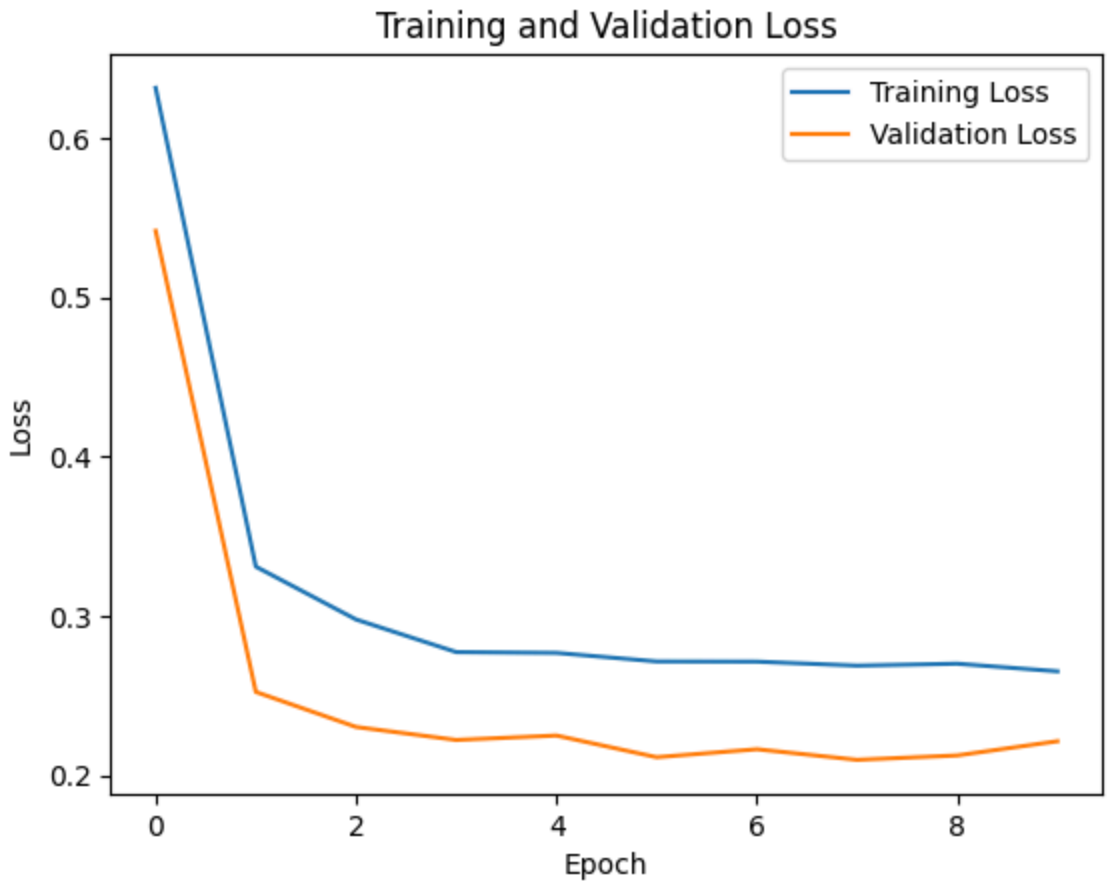
Observations:
Model achieved satisfactory performance metrics, but there is room for improvement, particularly in addressing overfitting.

Experiment 2: Fine-tuning (Unfrozen Encoder)
Methods:

Model Architecture and Training:

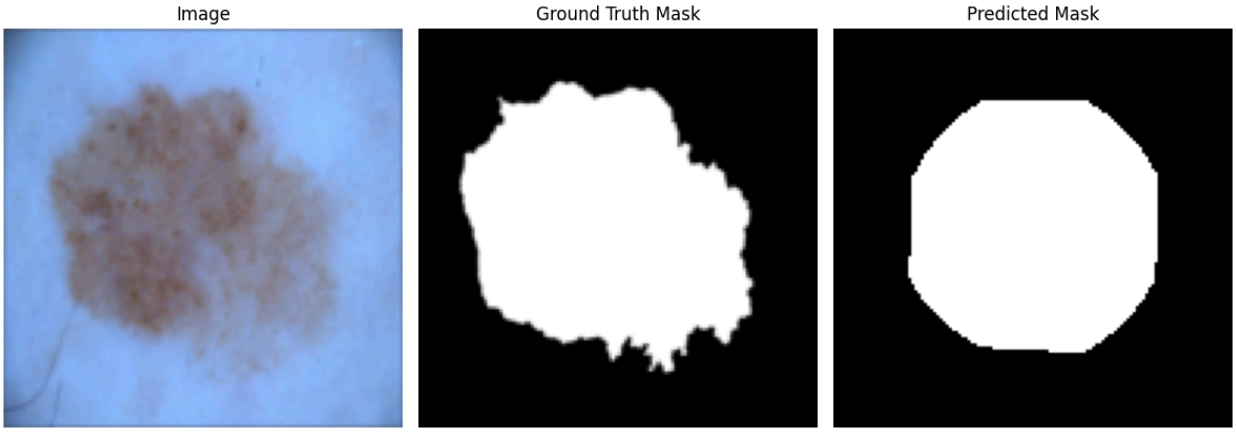
Utilized the same architecture as in Experiment 1.
Trained the segmentation model while fine-tuning the encoder weights.
All the layers of encoder are trained.
Used the pre-trained MobileNet encoder and a custom decoder.
Defined loss function, optimizer.
Training and Evaluation:

Trained the model for 10 epochs, monitoring training and validation loss.
Evaluated the model on the test dataset to calculate Intersection over Union (IoU) and Dice score.
Visualized several samples alongside their generated masks and ground truth for qualitative analysis.
Results:(Results taken safer running 10 epoch)
Intersection over Union (IoU): **0.5715**
Dice Score: **0.7118**
Loss Curves:



Training Loss showed a decreasing trend.
Validation Loss slightly increased after a certain number of epochs, suggesting potential overfitting.

After Prediction



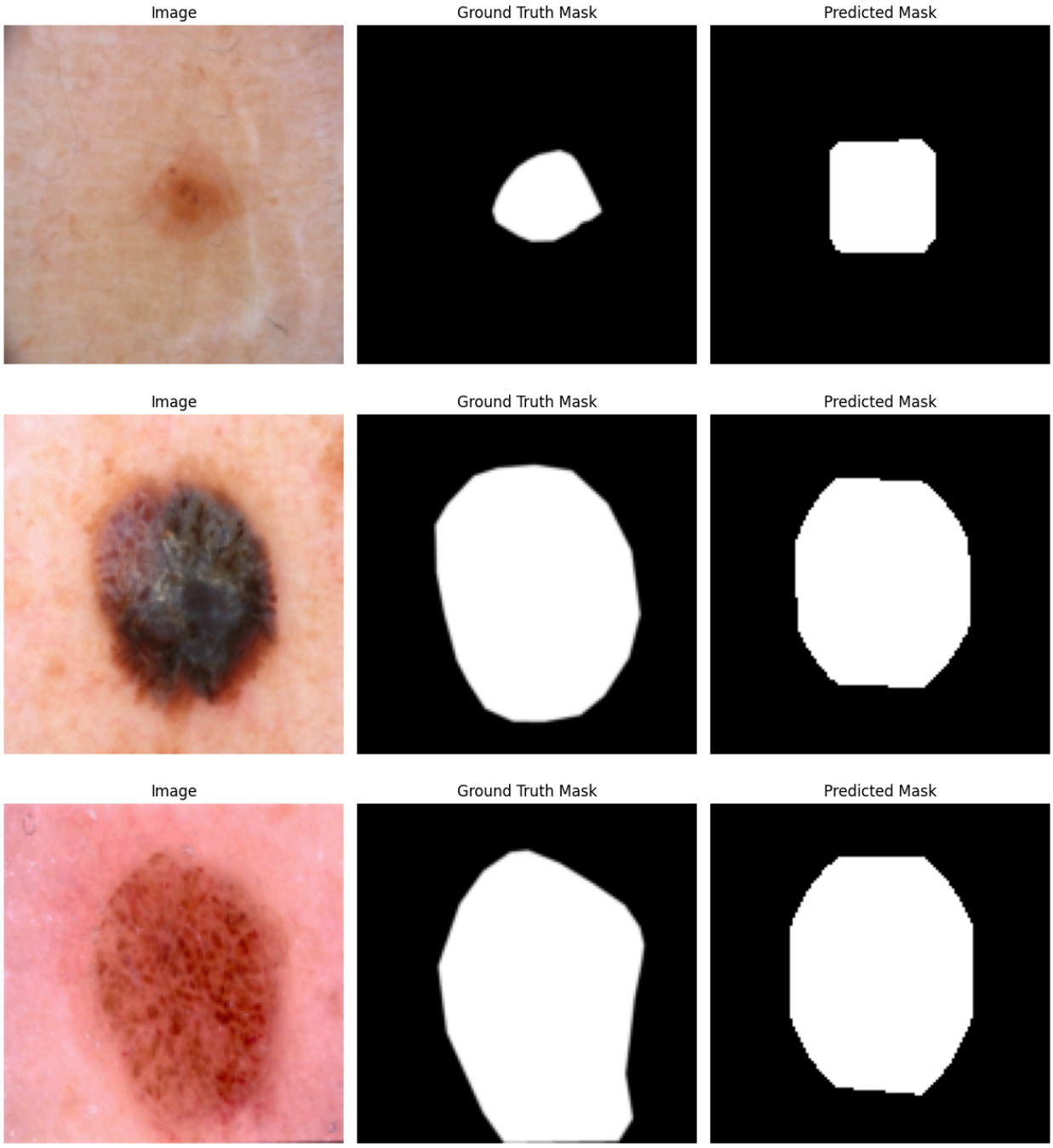
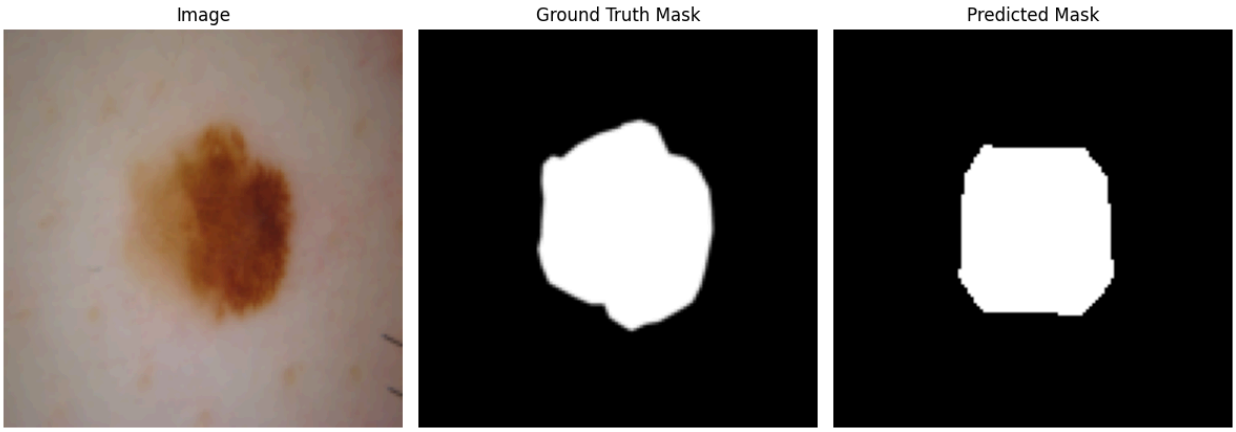


Image
Ground Truth Mask
Predicted Mask
Image
Ground Truth Mask
Predicted Mask
Image
Ground Truth Mask
Predicted Mask



Observations:

Model achieved **Better** performance metrics with Experiment 2, with **slightly Better** IoU and Dice scores.

Comparative Analysis:

Experiment	IoU	Dice Score	Final Training Loss	Final Validation Loss
Feature Extraction (Frozen Encoder)	**0.2862**	**0.4105**	**0.6301**	**0.6316**
Fine-tuning (Unfrozen Encoder)	**0.5715**	**0.7118**	**0.265**	**0.2212**

Both experiments utilized the same dataset and architecture.
Feature extraction with an unfrozen encoder(fine Tuning) resulted in marginally better IoU and Dice scores compared to frozen encoder.
Both experiments showed a similar trend in loss curves, with training loss decreasing over epochs and validation loss slightly increasing, indicating potential overfitting.
Fine-tuning the encoder significantly improves segmentation performance, suggesting that feature extraction with an unfrozen encoder is not sufficient for this task according to my observation as I was not able to perform complete with frozen encoder.

REFERENCES:

1.Upsample — PyTorch 2.2 documentation
2.Encoder Decoder | Sequence-to-Sequence Architecture | Deep Learning | CampusX (youtube.com)

3. MobileNet, MobileNetV2, and MobileNetV3 (keras.io)