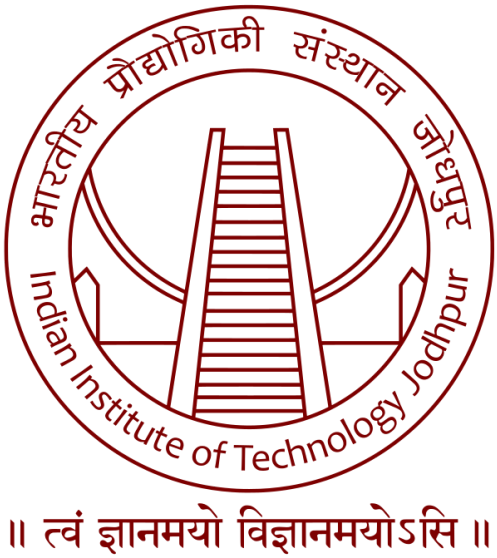
# Assignment 3

## Deep Learning




**Submitted By**

**Ratnesh Kumar Tiwari**
**(M23MAC011)**

**To**

**Dr. Deepak Mishra**

## Introduction:

Segmentation is a crucial task in computer vision that involves dividing an image into meaningful regions or segments. This report presents the results of experiments conducted on the ISIC 2016 dataset using a MobileNet V2 feature extractor pre-trained on ImageNetV1. The goal is to implement and evaluate the performance of two approaches:

a. Training the custom decoder with fixed encoder
b. Training the custom decoder along with fine-tuning the encoder weights.

## Dataset:

The provided dataset, known as ISIC 2016, is structured into the following components:

**Training Images (900 images):**
This subset comprises 900 training images. These images serve as the primary data for training machine learning models, particularly for segmentation tasks. Each image represents a unique instance from the dataset, likely containing skin lesions or other dermatological features.

**Training Masks (900 masks):**
Corresponding to the training images, this subset includes segmented masks. These masks serve as ground truth annotations for the training images. Each mask delineates the regions of interest within the corresponding training image, assisting in training supervised learning models, such as segmentation networks.

**Test Images (379 images):**
Consisting of 379 images, this subset serves as the validation set while training the models at every epoch. These images are distinct from the training set and are unused for the training. They are utilized to assess the generalization and performance of models in the training phase like overfitting.

**Test Masks (379 masks):**
Similar to the training masks, this subset includes segmented masks corresponding to the validation images. These masks serve as ground truth annotations for the validation images and are used for evaluating the effectiveness of segmentation models on the validation set.

## Data Preprocessing:

The ISIC 2016 dataset was preprocessed as follow:

**Training Set:**

a. Resizing images to 128x128 pixels

b. Applying random rotation with probability 10 percent with a maximum rotation of 15°
c. Random horizontal flip with probability of 50 percent
d. Normalization of images

**Validation Set:**

a. Resizing images to 128x128 pixels,
b. Normalization of images

Random rotation and flip was done to make our model more generalized in extracting the features and normalization was done for ease of computation.

## Architectures

### MobileNet V2:

MobileNet V2 is a neural network architecture designed specifically for efficient and lightweight deep learning on mobile and embedded devices. It builds upon the success of its predecessor, MobileNet V1, by introducing several improvements to further enhance performance while maintaining low computational complexity. Below is an overview of the key components and design principles of MobileNet V2:

**Depthwise Separable Convolution:**
MobileNet V2 primarily employs depthwise separable convolutions, which consist of two consecutive layers: depthwise convolution and pointwise convolution. This architecture significantly reduces the number of parameters and computations compared to traditional convolutional layers, making it well-suited for resource-constrained environments.

**Inverted Residuals with Linear Bottlenecks:**
MobileNet V2 introduces the concept of inverted residuals, where the input and output of each bottleneck block have the same dimensions. This design helps preserve information flow through the network while allowing for efficient channel-wise feature transformations. Linear bottlenecks further enhance the representational power of the network by applying a linear activation function within the bottleneck blocks.

**Expansion and Squeeze-Excitation:**
Each bottleneck block in MobileNet V2 begins with an expansion convolution layer that increases the number of channels followed by a squeeze-and-excitation operation. The squeeze operation reduces spatial dimensions via global average pooling, while the excitation operation learns channel-wise importance weights to emphasize informative features.

**Multiplier and Width Multiplier:**
MobileNet V2 introduces two hyperparameters: the multiplier and width multiplier. The multiplier adjusts the number of channels in each layer, providing a trade-off between model size and

accuracy. The width multiplier scales the entire network uniformly, allowing for efficient model scaling without changing the network architecture.

**Inverted Residual Block with Shortcut Connection:**
The core building block of MobileNet V2 is the inverted residual block, which consists of a sequence of depthwise separable convolutions, expansion convolution, and squeeze-and-excitation operations. Additionally, each block incorporates a shortcut connection to facilitate gradient flow and ease the training of deeper networks.

**Final Classification Layer:**
MobileNet V2 typically concludes with a global average pooling layer followed by a fully connected layer and softmax activation function for classification tasks. However, for tasks such as semantic segmentation or object detection, additional layers may be appended to adapt the network output to the specific task requirements.

Overall, the MobileNet V2 architecture strikes a balance between model efficiency and performance, making it a popular choice for various computer vision tasks, especially on mobile and edge devices where computational resources are limited. Its lightweight design and robust features enable deployment in real-world applications, ranging from image classification to object detection and beyond.

| Input | Operator | no of output channels | repeating number | stride |
|---|---|---|---|---|
| 128 x 128 x 3 | Conv 2D | 32 | 1 | 2 |
| 64 x 64 x 32 | Bottleneck | 16 | 1 | 1 |
| 64 x 64 x 16 | Bottleneck | 24 | 2 | 2 |
| 32 x 32 x 24 | Bottleneck | 32 | 3 | 2 |
| 16 x 16 x 32 | Bottleneck | 64 | 4 | 2 |
| 8 x 8 x 64 | Bottleneck | 96 | 3 | 1 |
| 8 x 8 x 96 | Bottleneck | 160 | 3 | 2 |
| 4 x 4 x 160 | Bottleneck | 320 | 1 | 1 |
| 4 x 4 x 320 | Covn 2D 1x1 | 1280 | 1 | 1 |

## Custom Decoder:

The CustomDecoder architecture is designed as a decoder component for a segmentation task. It takes feature maps from a preceding encoder, typically pretrained MobileNet V2 in this context, and upsamples them to generate segmentation masks. Below is an elaboration of its key components and operations:

**Transposed Convolution Layers:**
The decoder begins with a series of transposed convolution layers (ConvTranspose2d). These layers upsample the feature maps, gradually increasing the spatial resolution of the input.

**Batch Normalization Layers:**
Batch normalization (BatchNorm2d) layers are employed after each transposed convolution layer to stabilize and accelerate the training process. They normalize the input across the batch dimension, helping to mitigate issues related to internal covariate shift.

**Activation Functions:**
The rectified linear unit (ReLU) activation function (nn.ReLU) is applied after each convolutional operation, introducing non-linearity to the network and enabling the model to learn complex patterns within the data. Additionally, a sigmoid activation function (nn.Sigmoid) is used at the output layer to produce pixel-wise probability scores indicating the presence of the target object class.

**Dropout Layer:**
Dropout (nn.Dropout) is utilized to prevent overfitting by randomly setting a fraction of input units to zero during training. This regularization technique helps improve the generalization capability of the model.

**Residual Connections:**
Residual connections are incorporated within the decoder architecture to facilitate gradient flow and alleviate the vanishing gradient problem. These connections skip over several convolutional layers and directly add the input to the output of subsequent layers, allowing the model to learn residual mappings.

**Configurable Expansion and Compression:**
The architecture allows for configurable expansion and compression of feature maps through the use of convolutional layers with varying output channel sizes. This flexibility enables the model to capture and represent features at different levels of abstraction.

**Final Convolution and Upsampling:**
The decoder concludes with a convolutional layer followed by upsampling operations (ConvTranspose2d) to generate the final segmentation mask. The output mask is produced with the same spatial dimensions as the input image, containing pixel-wise predictions for the target object class.

**Parameter Sharing and Regularization:**
Parameter sharing is enforced across layers, reducing the number of trainable parameters and promoting parameter efficiency. Furthermore, regularization techniques such as dropout are employed to prevent overfitting and enhance model generalization.

| Input | Operator | no of output channels | repeating number | stride |
|---|---|---|---|---|
| 4 x 4 x 1280 | Conv Transpose 2D | 512 | - | 2 |
| 8 x 8 x 512 | Conv Transpose 2D | 256 | - | 2 |
| 16 x 16 x 256 | Conv Transpose 2D | 128 | - | 2 |
| 32 x 32 x 128 | Conv Transpose 2D | 64 | - | 2 |
| 64 x 64 x 64 | Conv Transpose 2D | 32 | - | 2 |
| 128 x128x 32 | Conv 2D | 1 | - | 1 |

## Experiments

### Experiment 1:

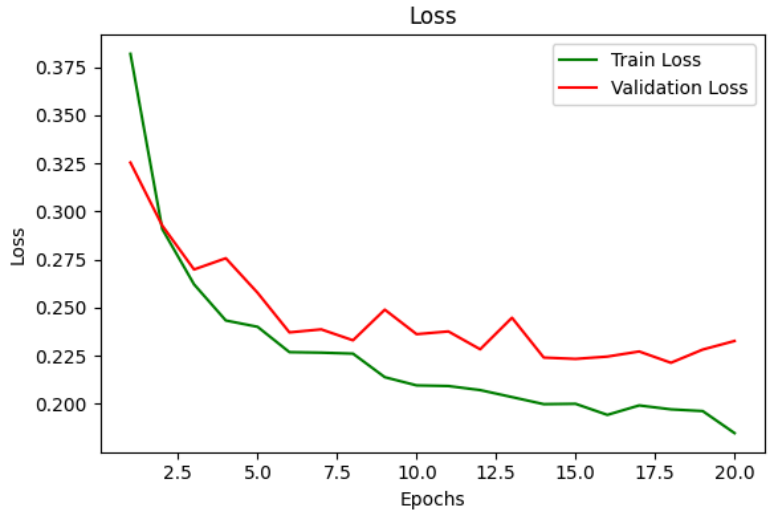
In this experiment, a pre-trained MobileNet V2 encoder was deployed, which had been previously trained on the ImageNetV1 dataset. Subsequently, a custom decoder was designed specifically for the segmentation task at hand. During training, the encoder weights were frozen, ensuring that only the decoder parameters were updated. Various configurations were explored, including extracting features from both the last layer of the encoder and multiple intermediate layers. This experimentation aimed to evaluate potential performance enhancements by leveraging different levels of abstraction within the encoder's feature hierarchy.

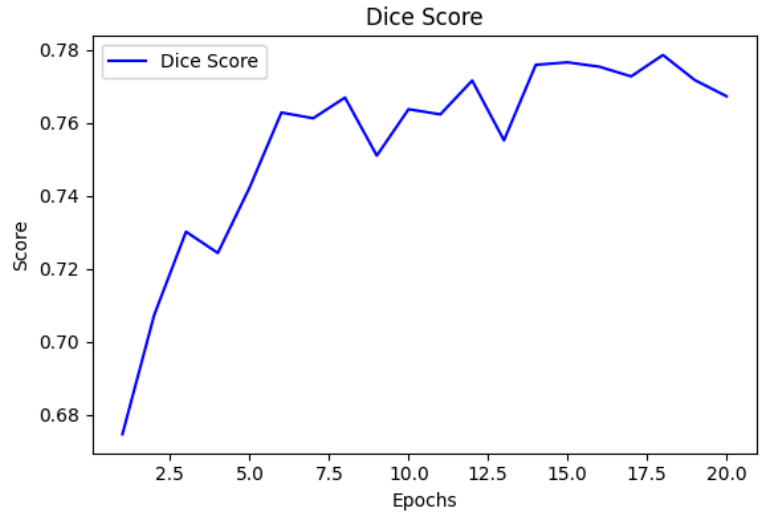
### i. Results:

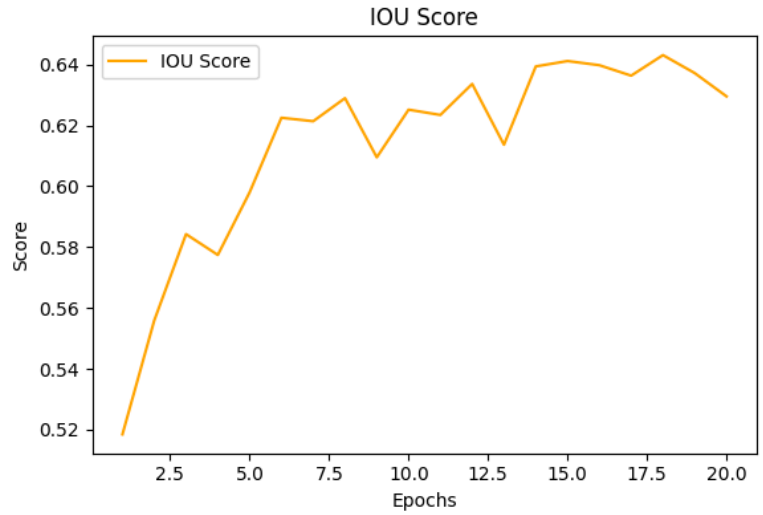
| Epoch | IOU Score | Dice Score | Training Loss | Validation Loss |
|---|---|---|---|---|
| 01/20 | 0.5185 | 0.6746 | 0.3819 | 0.3254 |
| 02/20 | 0.5559 | 0.7073 | 0.2909 | 0.2927 |
| 03/20 | 0.5843 | 0.7301 | 0.2621 | 0.2699 |
| 04/20 | 0.5774 | 0.7243 | 0.2434 | 0.2757 |
| 05/20 | 0.5980 | 0.7423 | 0.2401 | 0.2577 |
| 06/20 | 0.6225 | 0.7628 | 0.2269 | 0.2372 |
| 07/20 | 0.6214 | 0.7612 | 0.2267 | 0.2388 |
| 08/20 | 0.6289 | 0.7669 | 0.2262 | 0.2331 |
| 09/20 | 0.6095 | 0.7510 | 0.2139 | 0.2490 |
| 10/20 | 0.6252 | 0.7637 | 0.2097 | 0.2363 |
| 11/20 | 0.6234 | 0.7623 | 0.2093 | 0.2377 |
| 12/20 | 0.6336 | 0.7716 | 0.2072 | 0.2284 |
| 13/20 | 0.6136 | 0.7552 | 0.2036 | 0.2448 |
| 14/20 | 0.6394 | 0.7759 | 0.1999 | 0.2241 |
| 15/20 | 0.6411 | 0.7766 | 0.2001 | 0.2234 |
| 16/20 | 0.6398 | 0.7754 | 0.1943 | 0.2246 |
| 17/20 | 0.6363 | 0.7727 | 0.1992 | 0.2273 |

| | | | | |
|---|---|---|---|---|
| 18/20 | 0.6431 | 0.7786 | 0.1972 | 0.2214 |
| 19/20 | 0.6372 | 0.7717 | 0.1963 | 0.2283 |
| 20/20 | 0.6295 | 0.7673 | 0.1849 | 0.2327 |

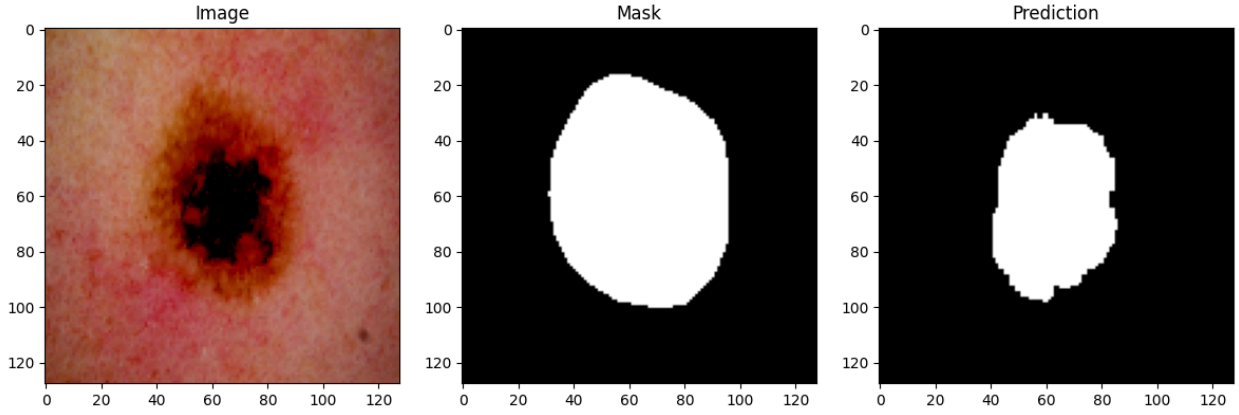
## Graphs:

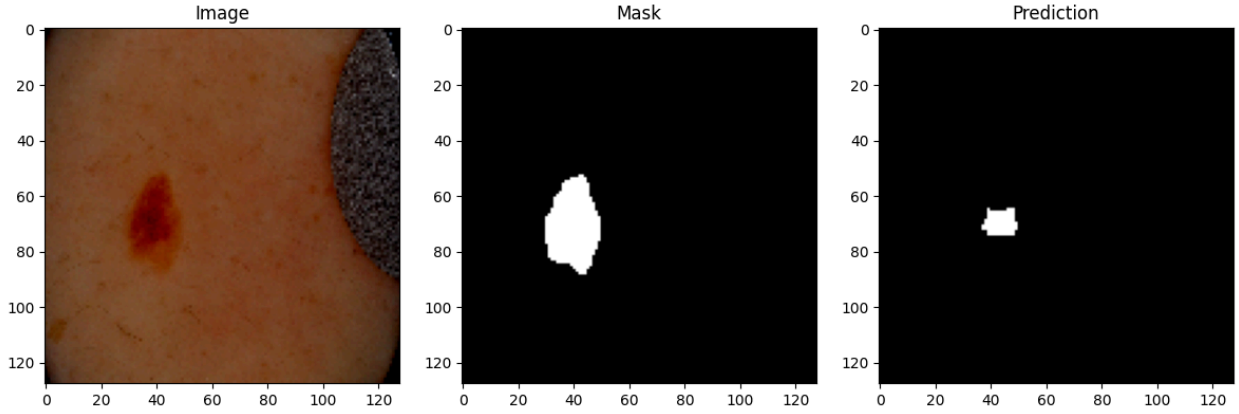

Loss
Train Loss
Validation Loss
Loss
Epochs
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Dice Score
Dice Score
Score
Epochs
0.78
0.76
0.74
0.72
0.70
0.68
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


IOU Score
IOU Score
Score
Epochs
0.64
0.62
0.60
0.58
0.56
0.54
0.52
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

## Few Samples  with ground and predicted mask:



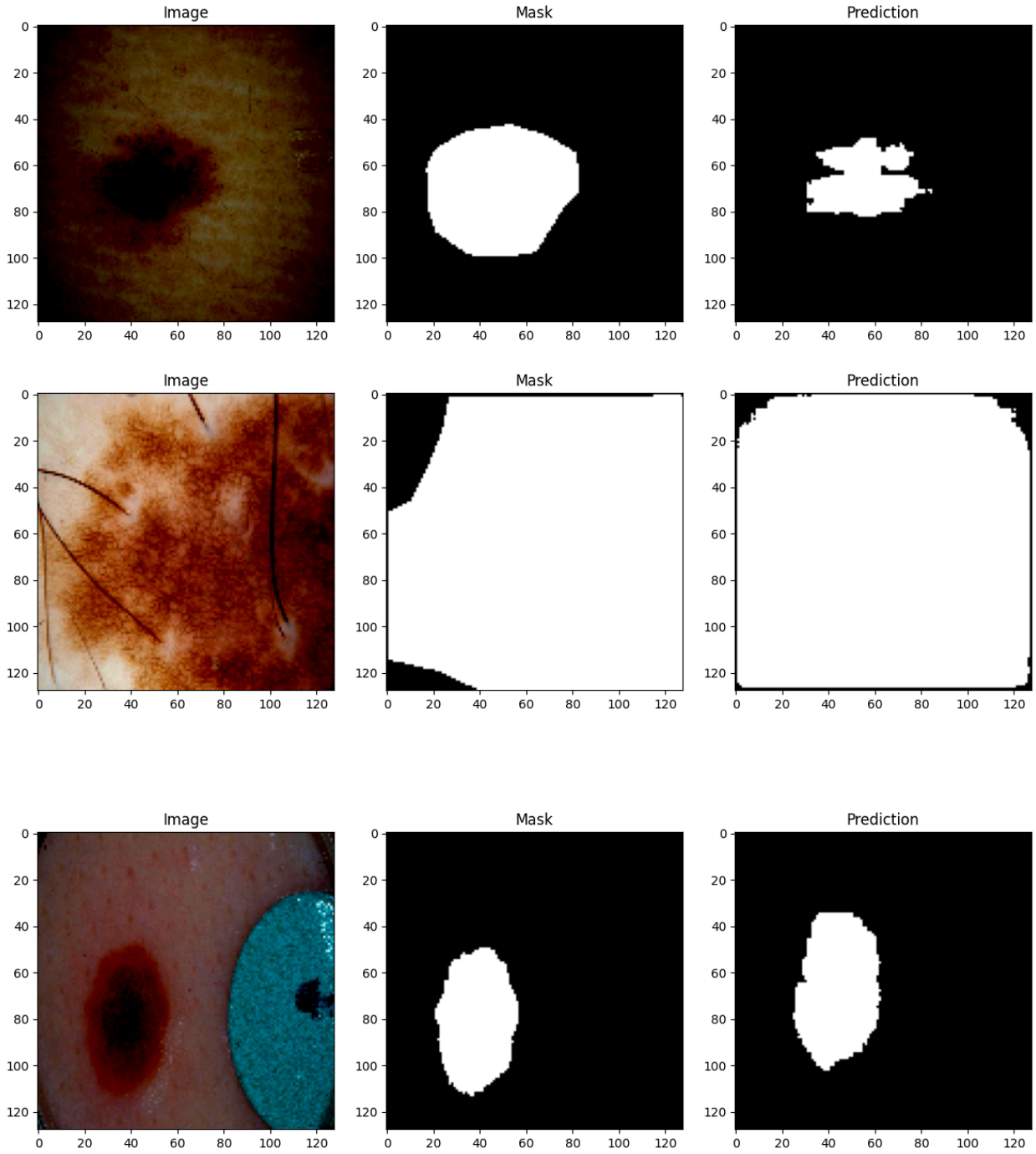


Image
Mask
Prediction
Image
Mask
Prediction
Image
Mask
Prediction

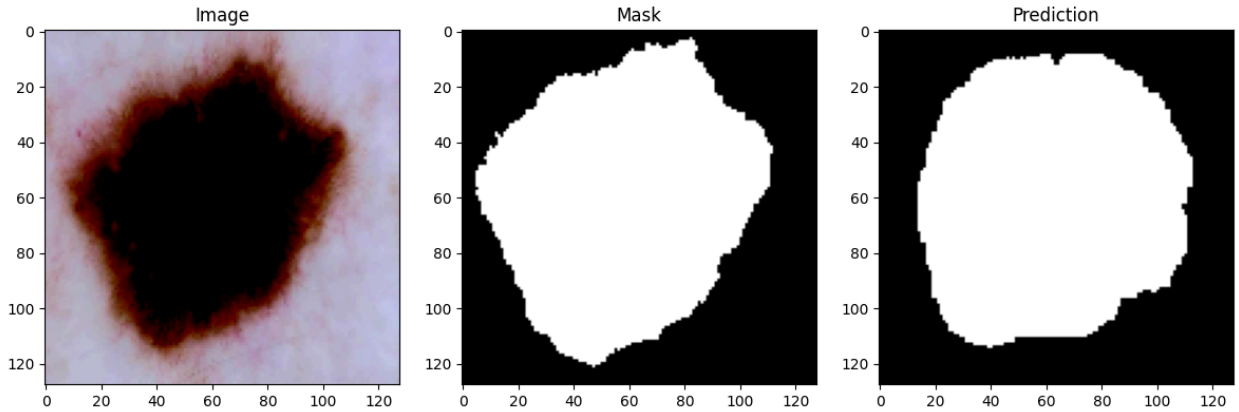


Image
Mask
Prediction

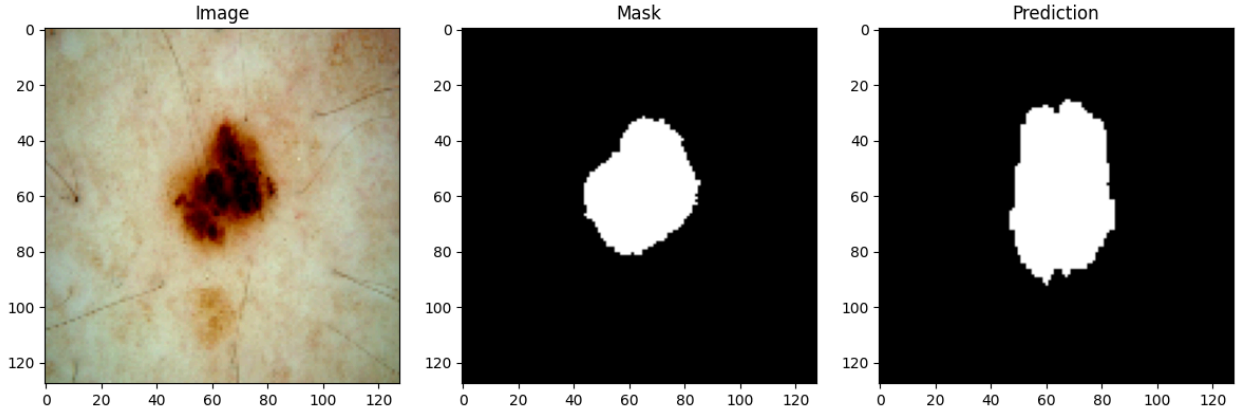


Image
Mask
Prediction

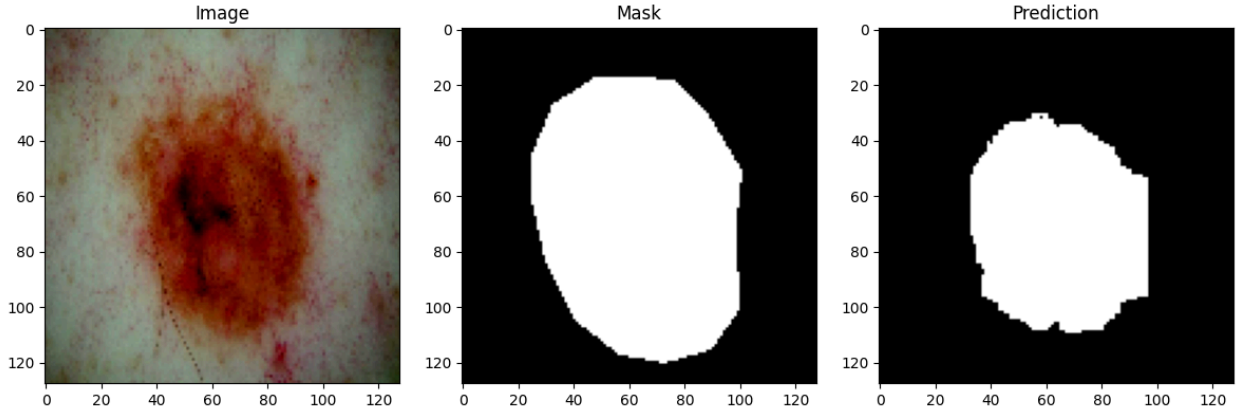
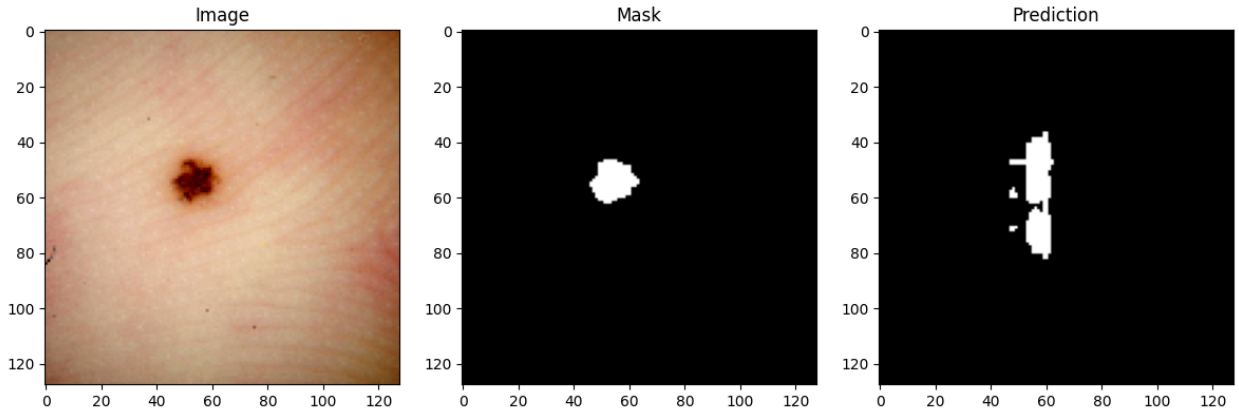
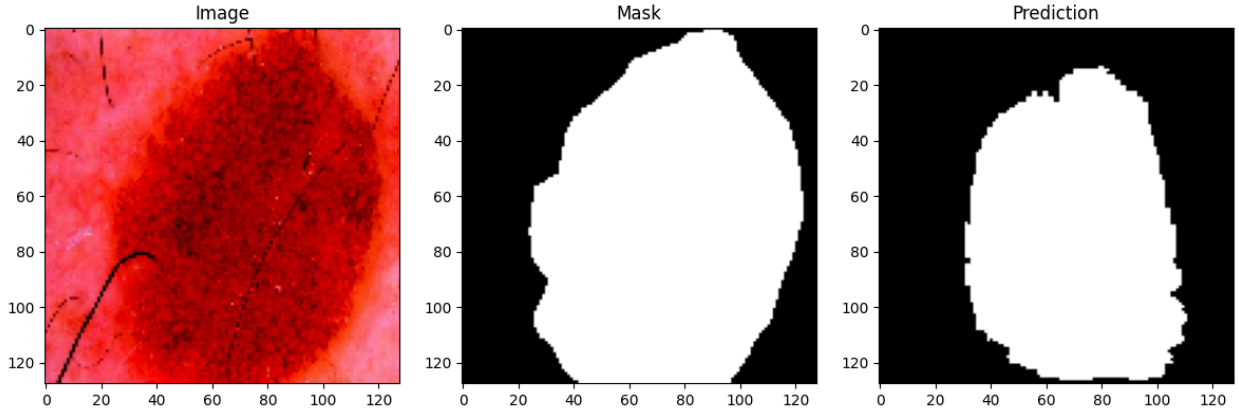


Image
Mask
Prediction




## ii. Observation:

**Encoder Features Utilization:** Experimenting with input from both the last layer of the encoder and multiple intermediate layers yielded insights into the importance of feature extraction depth. It was observed that features extracted from deeper layers of the encoder tended to capture more abstract and high-level representations, potentially leading to better segmentation performance.

**Decoder Training:** Training the custom decoder while keeping the encoder frozen allowed for efficient utilization of pre-trained features. This approach demonstrated the effectiveness of leveraging pre-trained encoders for segmentation tasks, as the decoder could focus on learning task-specific features without the need for extensive training data.

**Segmentation Performance:** The segmentation model achieved reasonable performance in terms of Intersection over Union (IoU) and Dice score. However, the choice of input features from the encoder and the design of the decoder architecture significantly influenced the segmentation accuracy.

**Computational Efficiency:** Utilizing a pre-trained MobileNet V2 encoder contributed to computational efficiency, as the model could leverage the learned representations from ImageNetV1 for feature extraction without the need for extensive computational resources.

## Experiment 2:

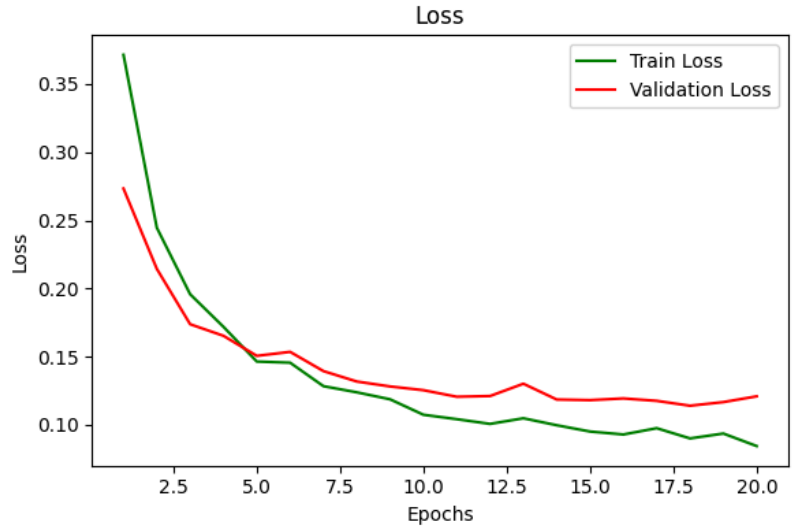
In Experiment 2, the same architecture as utilized in Experiment 1, was maintained, comprising a pre-trained MobileNet V2 encoder coupled with a custom decoder tailored for the segmentation task. However, unlike Experiment 1, where only the decoder was trained while keeping the encoder frozen, in this experiment, a different approach was adapted. Here, we fine-tuned the pre-trained MobileNet V2 encoder while simultaneously training the custom decoder. Fine-tuning involves updating the parameters of the pre-trained encoder alongside training the decoder during the optimization process. By fine-tuning the encoder weights, we aimed to adapt the learned representations of the encoder to better suit the segmentation task, while also allowing the decoder to learn task-specific features. This comprehensive training strategy aimed to leverage the strengths of both the pre-trained encoder and the custom decoder, ultimately enhancing the segmentation model's performance and its ability to accurately segment regions of interest within the images.

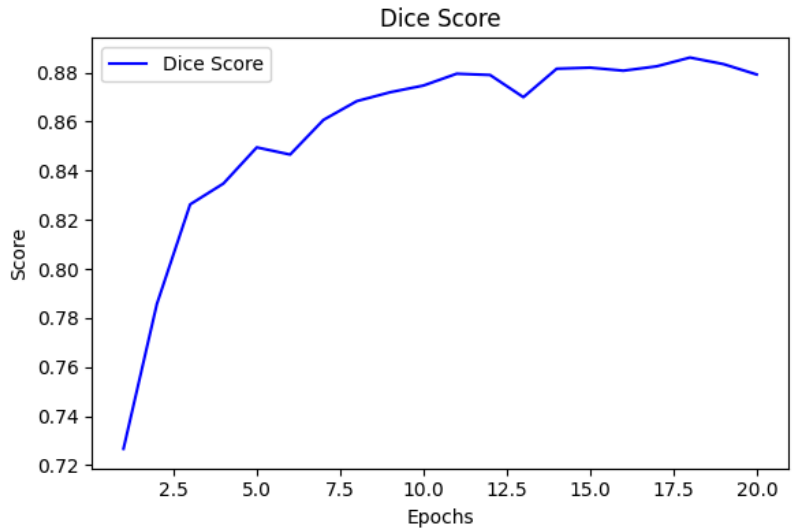
### i. Results:

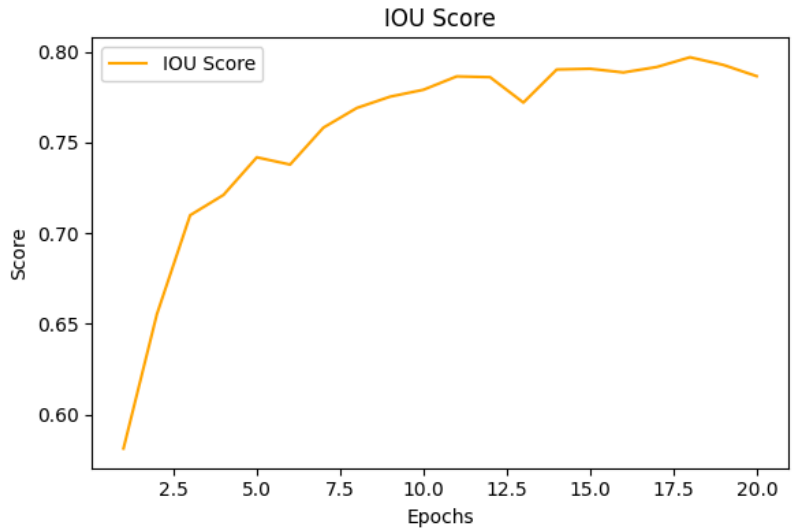
| Epoch | IOU Score | Dice Score | Training Loss | Validation Loss |
|---|---|---|---|---|
| 01/20 | 0.5814 | 0.7267 | 0.3715 | 0.2733 |
| 02/20 | 0.6553 | 0.7856 | 0.2446 | 0.2144 |
| 03/20 | 0.7099 | 0.8263 | 0.1957 | 0.1737 |

| | | | | |
|---|---|---|---|---|
| 04/20 | 0.7211 | 0.8348 | 0.1718 | 0.1652 |
| 05/20 | 0.7419 | 0.8495 | 0.1463 | 0.1505 |
| 06/20 | 0.7379 | 0.8466 | 0.1455 | 0.1534 |
| 07/20 | 0.7582 | 0.8607 | 0.1283 | 0.1393 |
| 08/20 | 0.7691 | 0.8683 | 0.1237 | 0.1317 |
| 09/20 | 0.7753 | 0.8719 | 0.1186 | 0.1281 |
| 10/20 | 0.7791 | 0.8747 | 0.1073 | 0.1253 |
| 11/20 | 0.7865 | 0.8795 | 0.1040 | 0.1205 |
| 12/20 | 0.7860 | 0.8790 | 0.1006 | 0.1210 |
| 13/20 | 0.7721 | 0.8699 | 0.1047 | 0.1301 |
| 14/20 | 0.7903 | 0.8815 | 0.0996 | 0.1185 |
| 15/20 | 0.7907 | 0.8820 | 0.0950 | 0.1180 |
| 16/20 | 0.7886 | 0.8807 | 0.0928 | 0.1193 |
| 17/20 | 0.7916 | 0.8825 | 0.0974 | 0.1175 |
| 18/20 | 0.7970 | 0.8861 | 0.0899 | 0.1139 |
| 19/20 | 0.7928 | 0.8834 | 0.0935 | 0.1166 |
| 20/20 | 0.7866 | 0.8792 | 0.0843 | 0.1208 |

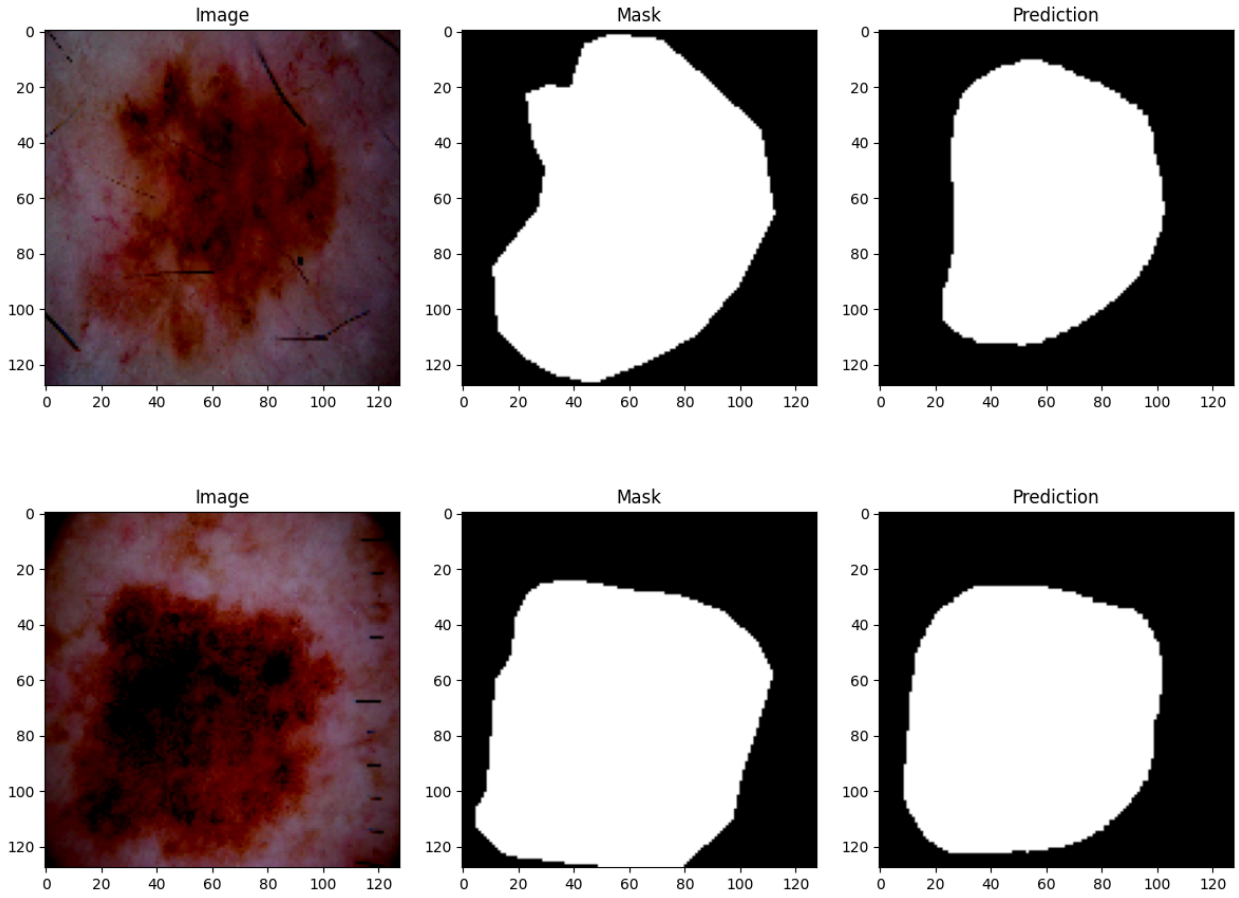
## Graphs:


Loss
Train Loss
Validation Loss
Loss
Epochs
0.35
0.30
0.25
0.20
0.15
0.10
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Dice Score
Dice Score
Score
Epochs
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.72
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


IOU Score
IOU Score
Score
Epochs
0.80
0.75
0.70
0.65
0.60
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

## Few Samples  with ground and predicted mask:



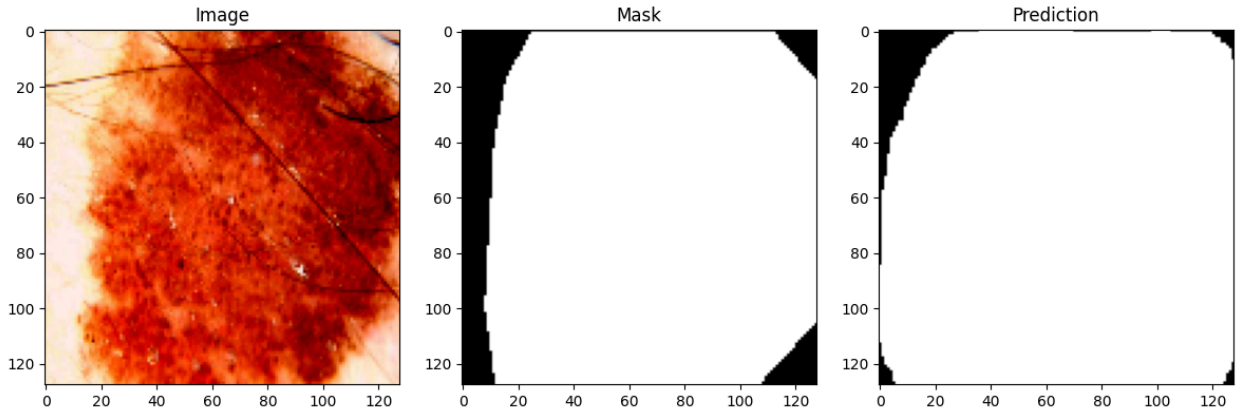


Image
Mask
Prediction

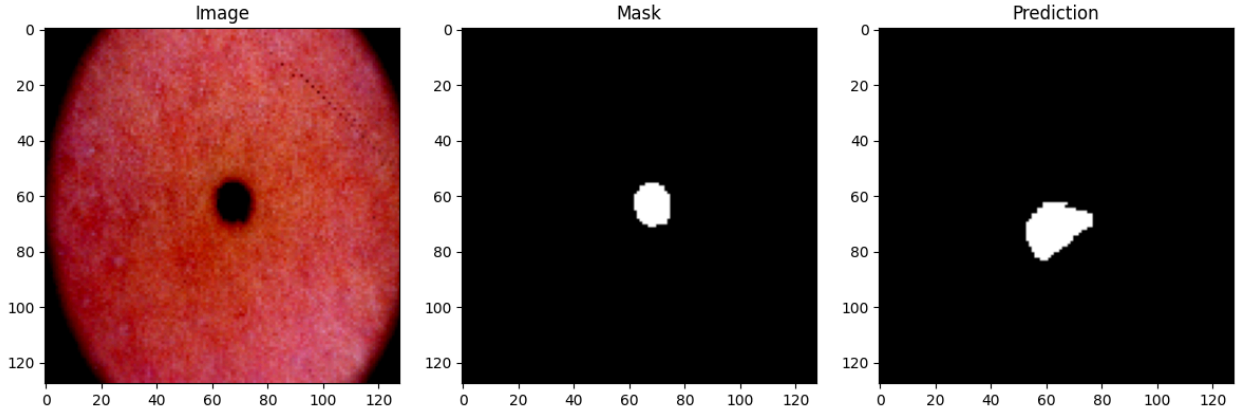


Image
Mask
Prediction

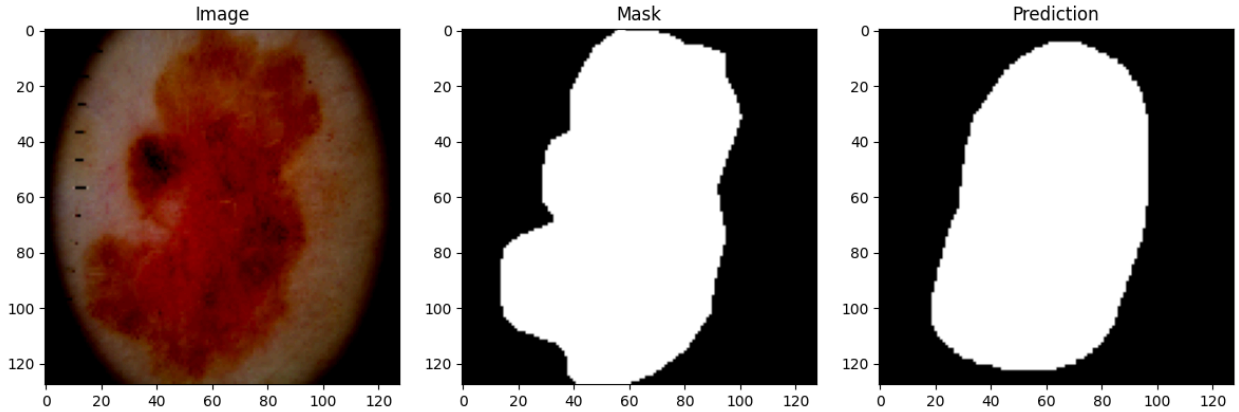


Image
Mask
Prediction

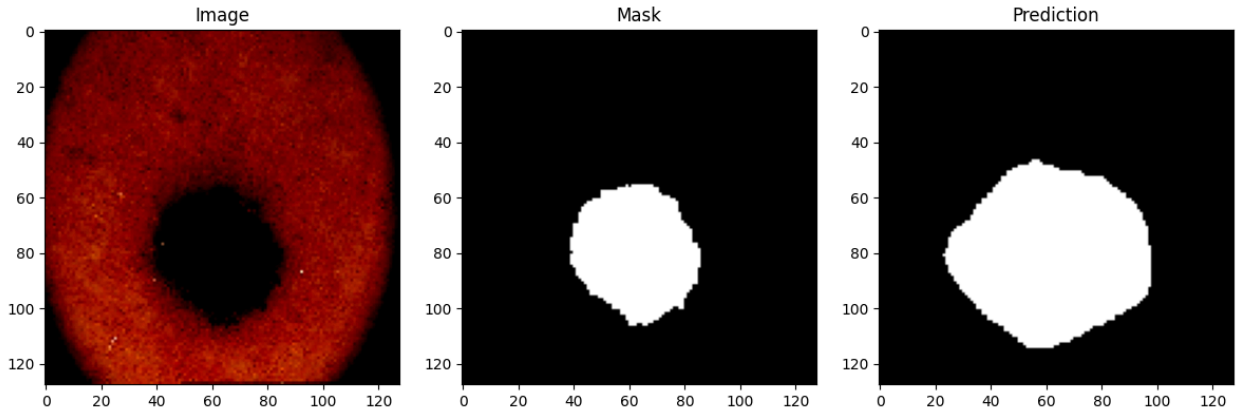


Image
Mask
Prediction

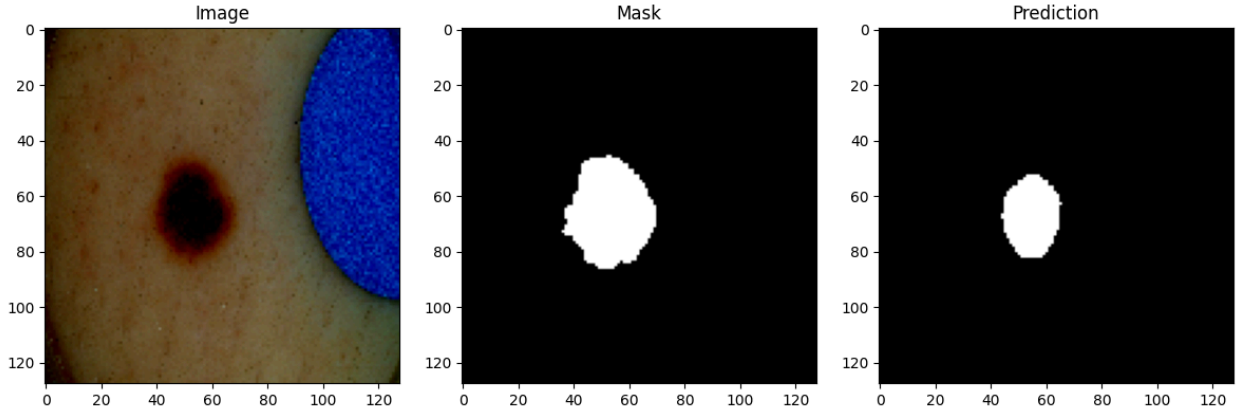


Image
Mask
Prediction

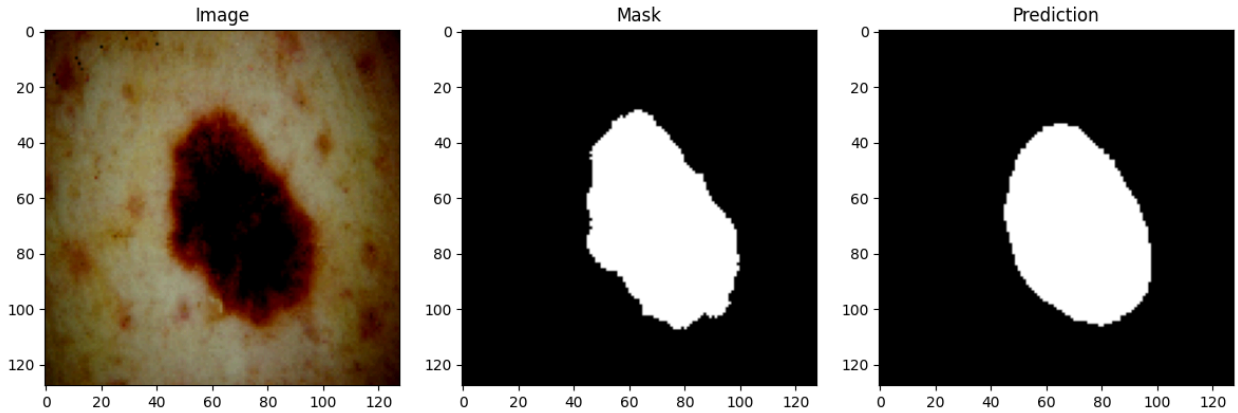
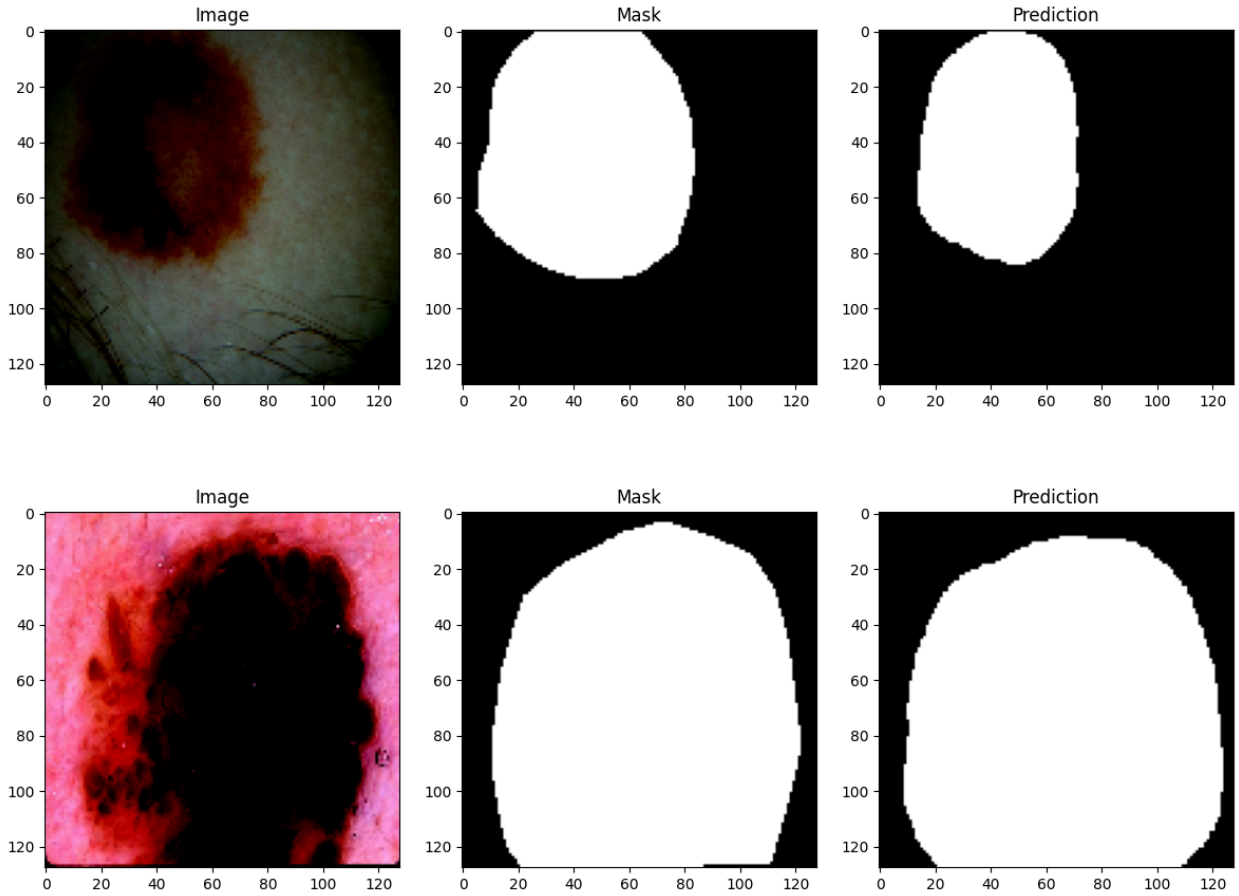


Image
Mask
Prediction




## ii. Observation:

**Encoder Adaptation:** Fine-tuning the pre-trained MobileNet V2 encoder alongside training the custom decoder allowed for adaptability to the segmentation task. The encoder could refine its learned representations to better suit the characteristics of the dataset, potentially improving segmentation performance.

**Model Generalization:** Fine-tuning the encoder weights facilitated better generalization of the segmentation model to the target dataset. By updating the encoder parameters, the model could learn dataset-specific features while retaining the knowledge gained from the pre-training on ImageNetV1.

**Training Dynamics:** Fine-tuning introduced additional training dynamics, as both the encoder and decoder parameters were optimized simultaneously. This process required careful

management of learning rates and regularization techniques to prevent overfitting and ensure stable convergence.

**Performance Improvement:** Fine-tuning the encoder led to improvements in segmentation accuracy, as indicated by higher IoU and Dice scores compared to Experiment 1. The adaptability of the encoder to the segmentation task contributed to enhanced feature representations, resulting in better segmentation performance on the test dataset.

## Comparison of Scores of both experiment

| Epoch | IOU Score 1 | Dice Score 1 | IOU Score 2 | Dice Score 2 |
|---|---|---|---|---|
| 01/20 | 0.5185 | 0.6746 | 0.5814 | 0.7267 |
| 02/20 | 0.5559 | 0.7073 | 0.6553 | 0.7856 |
| 03/20 | 0.5843 | 0.7301 | 0.7099 | 0.8263 |
| 04/20 | 0.5774 | 0.7243 | 0.7211 | 0.8348 |
| 05/20 | 0.5980 | 0.7423 | 0.7419 | 0.8495 |
| 06/20 | 0.6225 | 0.7628 | 0.7379 | 0.8466 |
| 07/20 | 0.6214 | 0.7612 | 0.7582 | 0.8607 |
| 08/20 | 0.6289 | 0.7669 | 0.7691 | 0.8683 |
| 09/20 | 0.6095 | 0.7510 | 0.7753 | 0.8719 |
| 10/20 | 0.6252 | 0.7637 | 0.7791 | 0.8747 |
| 11/20 | 0.6234 | 0.7623 | 0.7865 | 0.8795 |

| | | | | |
|---|---|---|---|---|
| 12/20 | 0.6336 | 0.7716 | 0.7860 | 0.8790 |
| 13/20 | 0.6136 | 0.7552 | 0.7721 | 0.8699 |
| 14/20 | 0.6394 | 0.7759 | 0.7903 | 0.8815 |
| 15/20 | 0.6411 | 0.7766 | 0.7907 | 0.8820 |
| 16/20 | 0.6398 | 0.7754 | 0.7886 | 0.8807 |
| 17/20 | 0.6363 | 0.7727 | 0.7916 | 0.8825 |
| 18/20 | 0.6431 | 0.7786 | 0.7970 | 0.8861 |
| 19/20 | 0.6372 | 0.7717 | 0.7928 | 0.8834 |
| 20/20 | 0.6295 | 0.7673 | 0.7866 | 0.8792 |

Analyzing the trends between the two sets of data for IOU (Intersection over Union) scores and Dice scores reveals several interesting insights:

Initial Performance: In the early epochs (e.g., 01/20 to 05/20), the second set of data consistently outperforms the first set in both IOU and Dice scores. This indicates that the model trained with the second set of data initially achieves better segmentation accuracy and overlap with ground truth masks compared to the first set.

Mid-Training Performance: As training progresses (e.g., from 06/20 to 15/20), both sets of data show improvements in IOU and Dice scores, although the rate of improvement varies. Interestingly, there are instances where the first set catches up and even surpasses the second set in performance (e.g., 12/20 and 13/20 for IOU Score 1, and 14/20 for Dice Score 1). This suggests that the first set of data might have certain characteristics or features that are advantageous for model learning in later epochs.

Stability and Convergence: Towards the end of training (e.g., 16/20 to 20/20), both sets of data demonstrate similar trends in performance, with slight fluctuations in scores. This indicates that the models trained with both datasets converge to a stable performance level, although there might be minor differences in their final performance metrics.

Overall Comparison: Despite fluctuations and occasional crossovers in performance, the second set of data consistently maintains a slightly higher performance level throughout most epochs. However, the differences in scores between the two sets are relatively small, suggesting that both datasets are effective for training the segmentation model, albeit with slight variations in performance.

In conclusion, while the second set of data generally exhibits slightly better performance in terms of IOU and Dice scores, both sets demonstrate improvements over training epochs and ultimately converge to similar performance levels. Understanding these trends can provide valuable insights for optimizing dataset selection and model training strategies in medical image segmentation tasks.

**Please find the code below:**

https://colab.research.google.com/drive/10OtVXrOZnq0xYo8J499205UDIbfNUZvx?usp=sharing