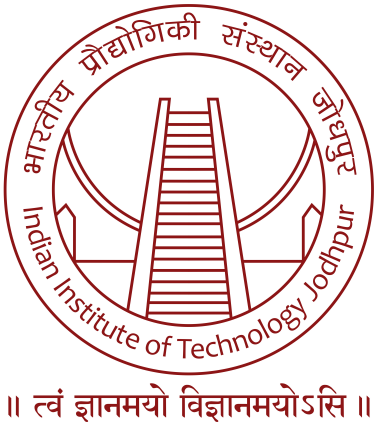
Indian Institute of Technology, Jodhpur



Creating a Segmentation model on top of Mobile Net Encoder

CSL7590: Deep Learning (Spring, 2024)

Assignment: 3

Due Date: March 19, 2024

Name:- Sahil
Roll No.:- M21MA210

M.Sc - M.Tech (Data and Computational Sciences)
Department of Mathematics

Contents

List of Figures

List of Tables

1 About ISIC 2016 Dataset

The ISIC (International Skin Imaging Collaboration) Dataset 2016 is a publicly available dataset primarily used for research in skin image analysis, particularly for melanoma detection and classification. It consists of a large collection of dermoscopic images, along with corresponding metadata and ground truth annotations. In our dataset, we have the following composition of data:

- 900 Training images.
- Training image's segmentation masks of same number.
- 379 Test images.
- Test image's segmentation masks of same number.

1.1 Pre-processing of Dataset

- The dataset has images in 'JPEG' format whereas mask in 'PNG' format. As instructed in the assignment, the images and masks are resized to 128×128.
- Custom Data loader was created to load the images along with their masks sequentially, which can be further employed during training,validation and testing phase.
- The training dataset is split for validation into ratio $0.8 : 0.2$, thus training and validation set has 720 & 180 samples respectively.
- Three datasets and data-loaders were created which serves purpose for training, validation and testing.
- Some Augmentations were performed like Horizontal flip, Vertical Flip, Color Jitters to a sample dataset.
- After pre-processing, it was verified whether the image is mapped to its actual segmentation mask or not.
- After pre-processing, the image size is [128,128,3] as [height,width,channels] and that of mask is [128,128,1].

1.2 Exploring and Analyzing the dataset

- Visualising some random images and its segmentation mask from the dataset gives the following results:

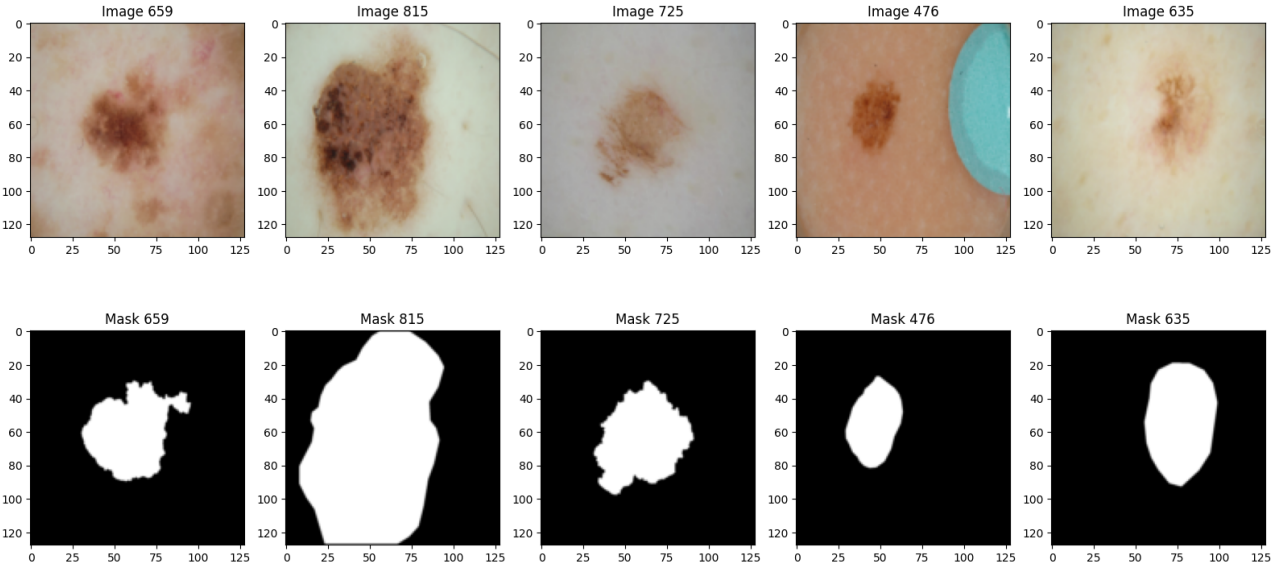


Figure 1: Sample Images and their Segmentation masks

- The mean pixel value in the images is: **0.6366** whereas the standard deviation of pixel values is: **0.1514**.
- Analysing the distribution of image sizes in the given dataset, we yield the following results:

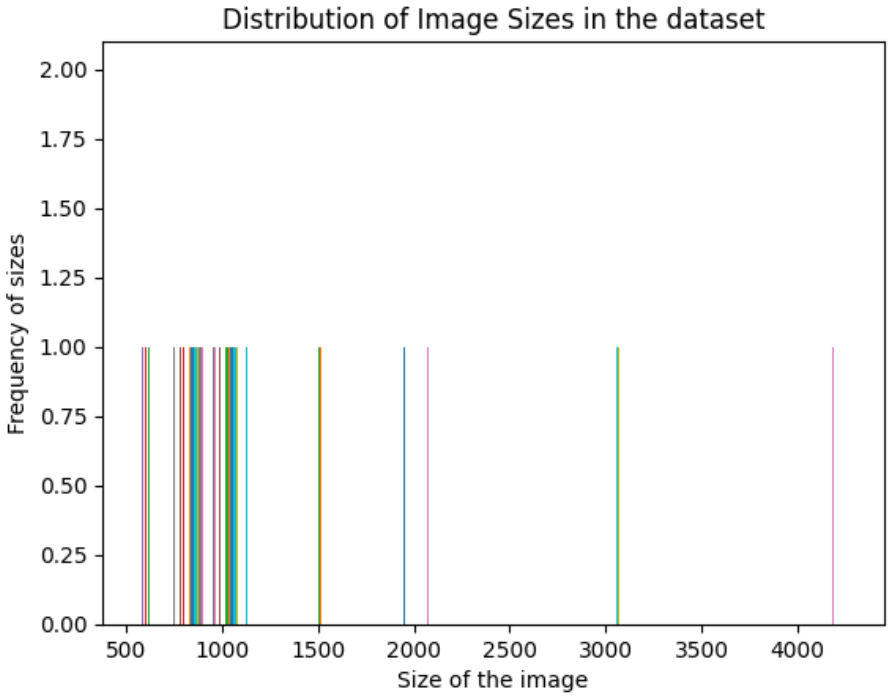


Figure 2: Distribution of image sizes

- Performed some augmentations as discussed above and visualised those and yield the following results:

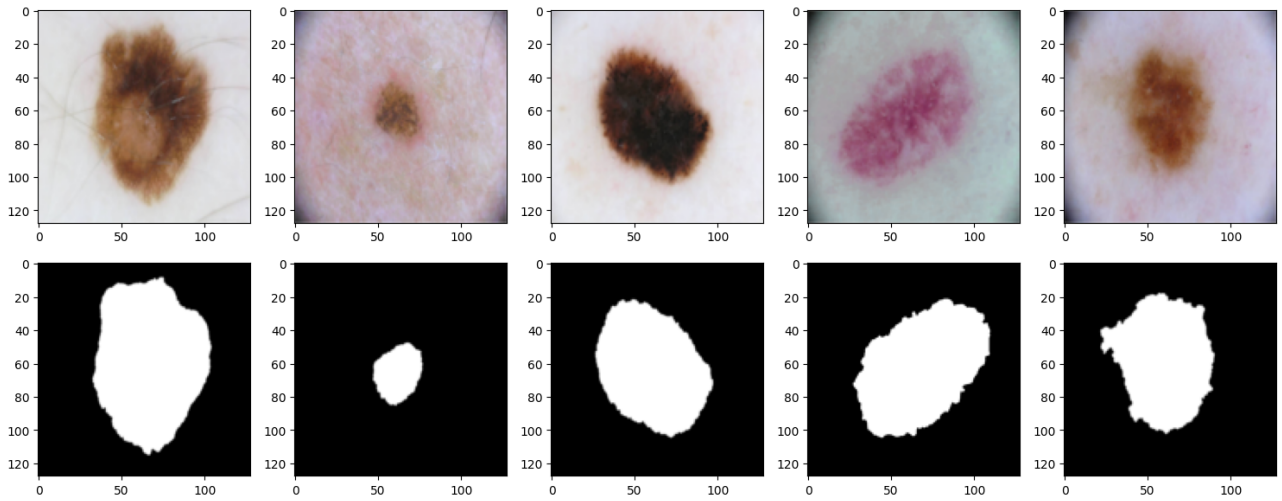


Figure 3: Visuals of augmentations on the dataset

2 Network Architecture

2.1 Layer-wise description

The Architecture used in the assignment is as follows:- Used a pre-trained Mobile-Net encoder trained on ImageNetV1 and then designed a custom decoder on top of this encoder for segmentation task. The decoder designed has the following architecture:

1. **2D transposed convolution Layer (I)**
   (a) Input channels: 1280
   (b) Output channels: 512
   (c) Kernel size: 4×4
   (d) Stride: 2
   (e) Zero-Padding: 1
2. **2D transposed convolution Layer (II)**
   (a) Input channels: 512
   (b) Output channels: 256
   (c) Kernel size: 4×4
   (d) Stride: 2
   (e) Zero-Padding: 1
3. **2D transposed convolution Layer (III)**

   (a) Input channels: 256
   (b) Output channels: 128
   (c) Kernel size: 4×4
   (d) Stride: 2
   (e) Zero-Padding: 1

4. **2D transposed convolution Layer (IV)**
   (a) Input channels: 128
   (b) Output channels: 64
   (c) Kernel size: 4×4
   (d) Stride: 2
   (e) Zero-Padding: 1

5. **2D transposed convolution Layer (V)**
   (a) Input channels: 64
   (b) Output channels: 32
   (c) Kernel size: 4×4
   (d) Stride: 2
   (e) Zero-Padding: 1

6. **2D convolution Layer**
   (a) Input channels: 32
   (b) Output channels: 1
   (c) Kernel size: 1×1
   (d) Stride: 1

7. 2D transposed convolutional Layer is employed for up sampling or increasing the spatial resolution of feature maps.

8. Zero Padding is used in it so that the dimension of the input images can be preserved.

9. In order to prevent over-fitting of the model during training phase, Dropout regularisation has been used.

10. For the first task, the encoder weights are frozen whereas for the second task, where we have to perform fine tuning, the encoder weights are made to learn by a very small learning rate.

11. Used AdamW optimiser with learning rate as 0.001 and loss function as **Binary Cross Entropy with Logits Loss**.

2.2 Methodology and Working of model

- Initially, the instances of encoder and decoder are created.
- For task 1, the encoder weights are set to freeze, whereas for task 2, the encoder weights are also made to learn but with very small learning rate of 0.0001, so that the weights would be updated only if their is a large change in the value of gradients.
- Then, the data is passed through the encoder and the features are extracted from it.
- The extracted features are then sent to decoder for segmentation task
- The decoder then produces the segmentation masks for the input images, compared them with the actual segmentation masks, computed loss and then gradients are calculated and loss is rectified using back propagation method.
- In this way, the complete model is trained for 20 epochs for both the tasks.
- For each epoch, all the losses *viz.* training, validation and test losses are calculated.
- When the model was validated during the training phase, for each epoch IOU score, Dice Score and pixel accuracy was calculated.
- The trained model is then tested on test dataset and its IOU, Dice Score and pixel wise accuracy was computed to know how the model performance.
- The loss, score and accuracy curves were plotted and total trainable and non trainable parameters were reported.
- At last, a text file created on Google Drive which records the value of decoder parameters in both the tasks.

3 Results and Observations

3.1 Task 1: Feature Extraction

1. The following table contains the recorded information per epoch for 20 epochs:

Epoch	Training Loss	Validation Loss	Test Loss	IoU Score	Dice Score
1	0.6843	0.6004	0.6164	0.0726	0.1354
2	0.4922	0.3937	0.4597	0.3614	0.5309
3	0.3933	0.3657	0.4055	0.3776	0.5482
4	0.3727	0.3731	0.3893	0.5222	0.6861
5	0.3293	0.3485	0.3767	0.5200	0.6842
6	0.3131	0.3837	0.4726	0.2753	0.4317
7	0.2900	0.3200	0.3644	0.5048	0.6709
8	0.2746	0.3249	0.3659	0.4479	0.6187
9	0.2592	0.3297	0.3663	0.5257	0.6891
10	0.2635	0.3368	0.3664	0.5217	0.6857
11	0.2531	0.3571	0.3703	0.5497	0.7094
12	0.2530	0.3522	0.3882	0.4529	0.6234
13	0.2726	0.3638	0.4151	0.5089	0.6745
14	0.2469	0.3283	0.3592	0.5052	0.6713
15	0.2356	0.3530	0.3648	0.5134	0.6785
16	0.2289	0.3310	0.3575	0.5322	0.6947
17	0.2244	0.3310	0.3888	0.4520	0.6226
18	0.2272	0.3415	0.4259	0.4840	0.6523
19	0.2387	0.4111	0.4219	0.5585	0.7167
20	0.2090	0.3324	0.3789	0.4705	0.6399

Table 1: Task 1: Model Performance

2. The values of the metrics after training phase is:

Metric	Value
Mean Train Loss	0.3031
Mean Validation Loss	0.3639
Mean Test Loss	0.4027
Min IoU Score	0.0726
Max IoU Score	0.5585
Mean IoU Score	0.4578
Min Dice Score	0.1354
Max Dice Score	0.7167
Mean Dice Score	0.6182
Mean Pixel-wise Accuracy	0.8150

Table 2: Metrics of Model on Task 1

3. The plot of training, validation and test losses is as below:

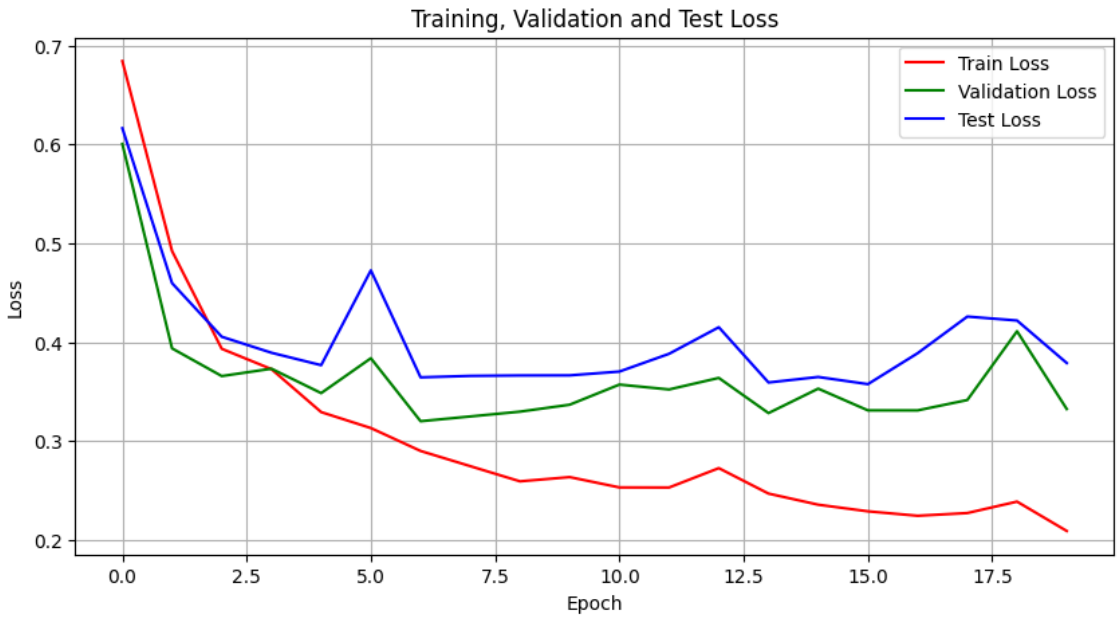


Figure 4: Plot showing Training, Validation and Test Loss

4. The plot of IOU score per epoch:

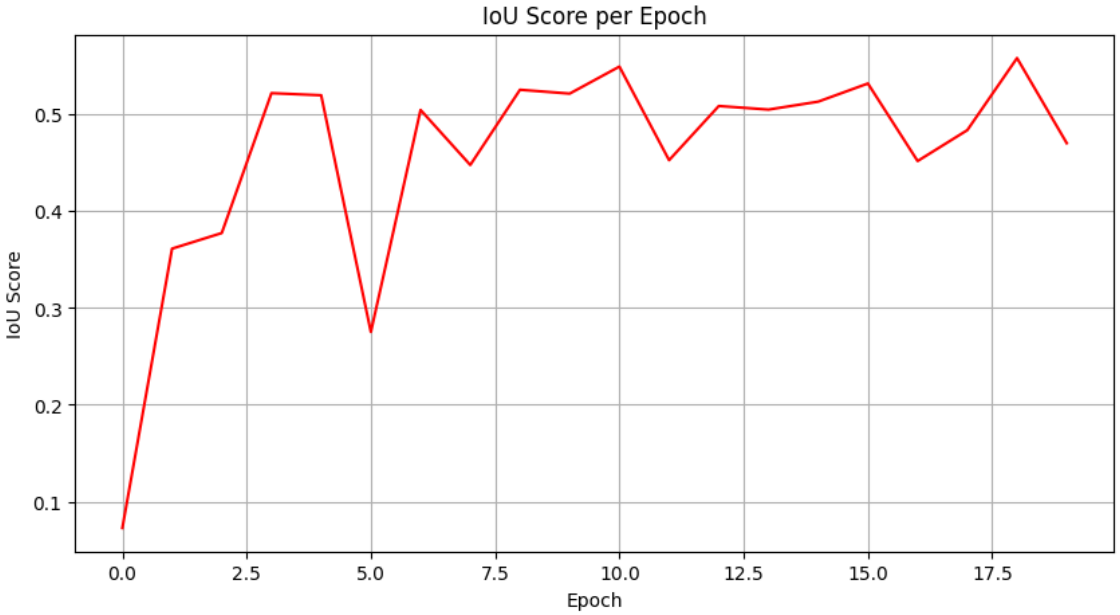


Figure 5: IOU Score per epoch

5. The plot of Dice score per epoch:

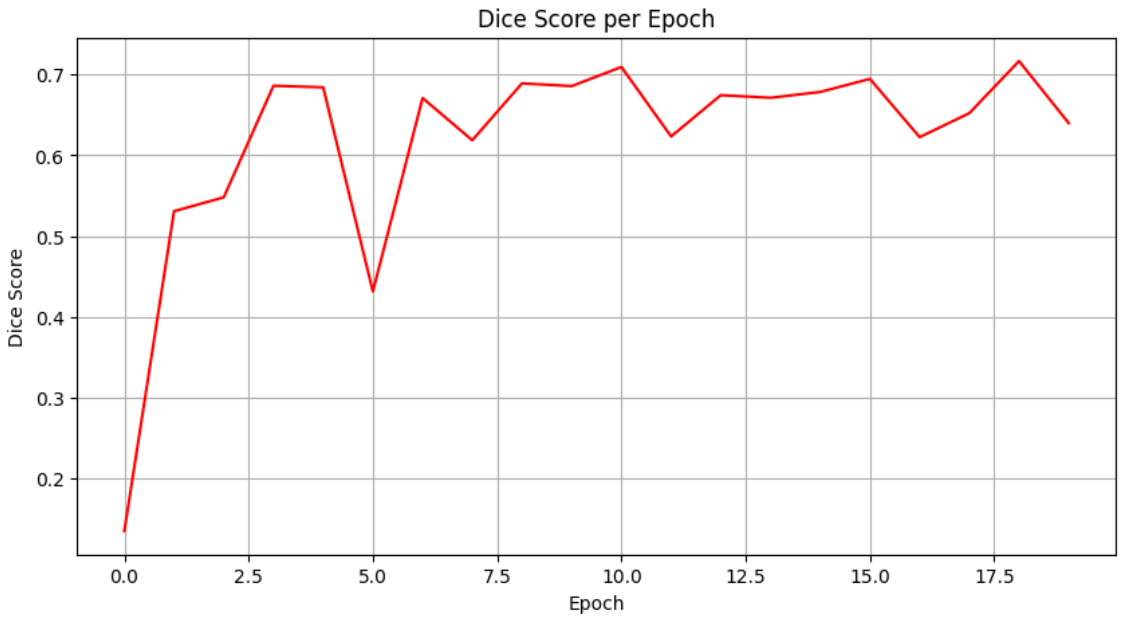


Figure 6: Dice Score per epoch

6. The plot of pixel wise accuracy:

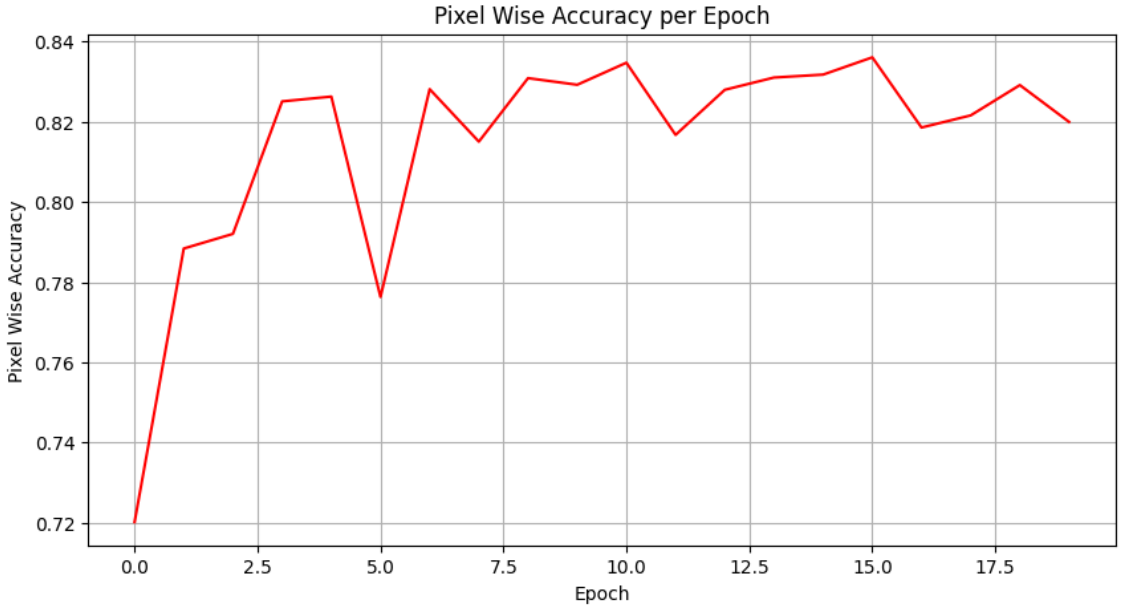


Figure 7: Pixel Wise Accuracy

7. Visualisation of the predicted segmentation masks with the corresponding image and ground truth i.e. Actual mask.

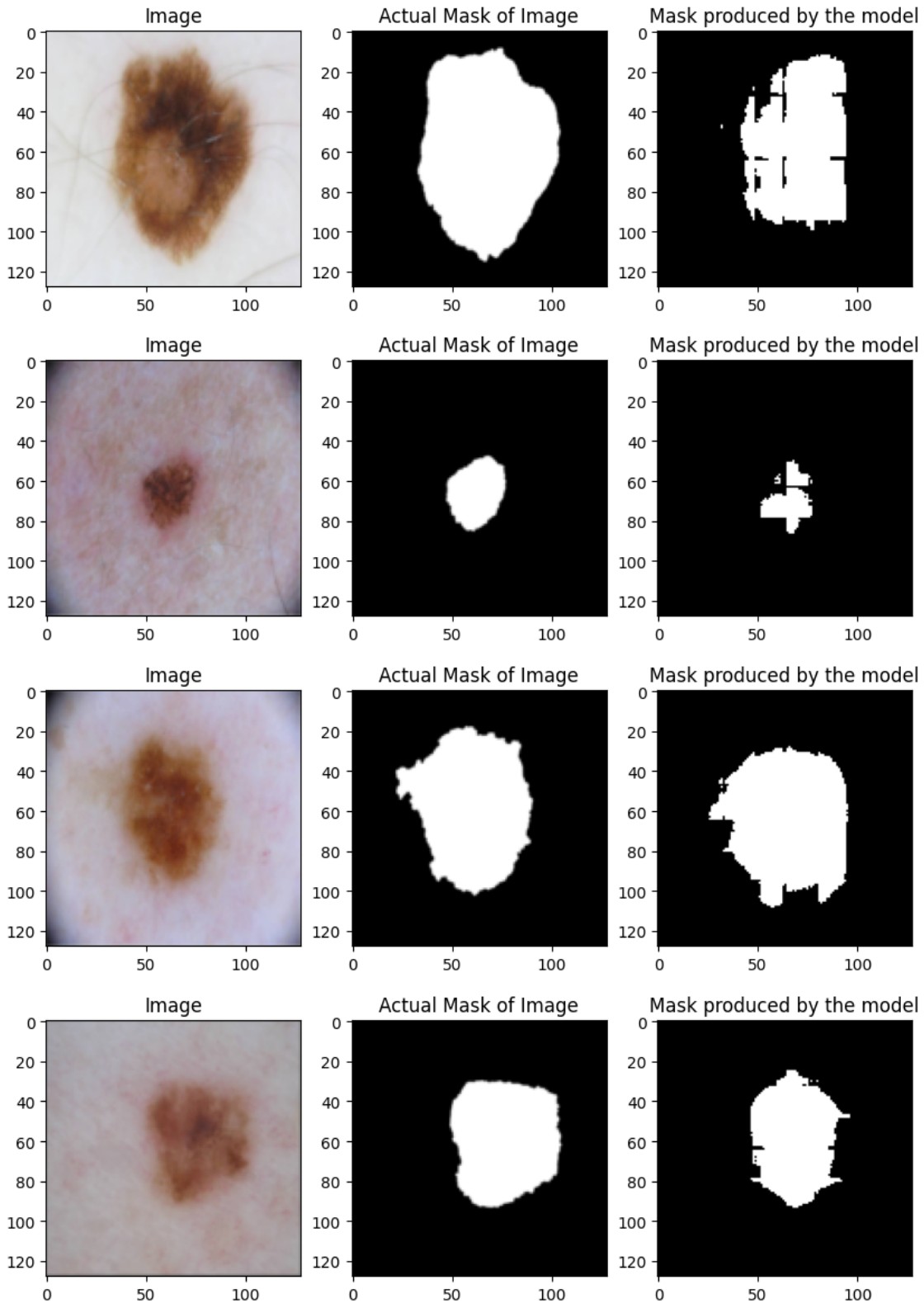
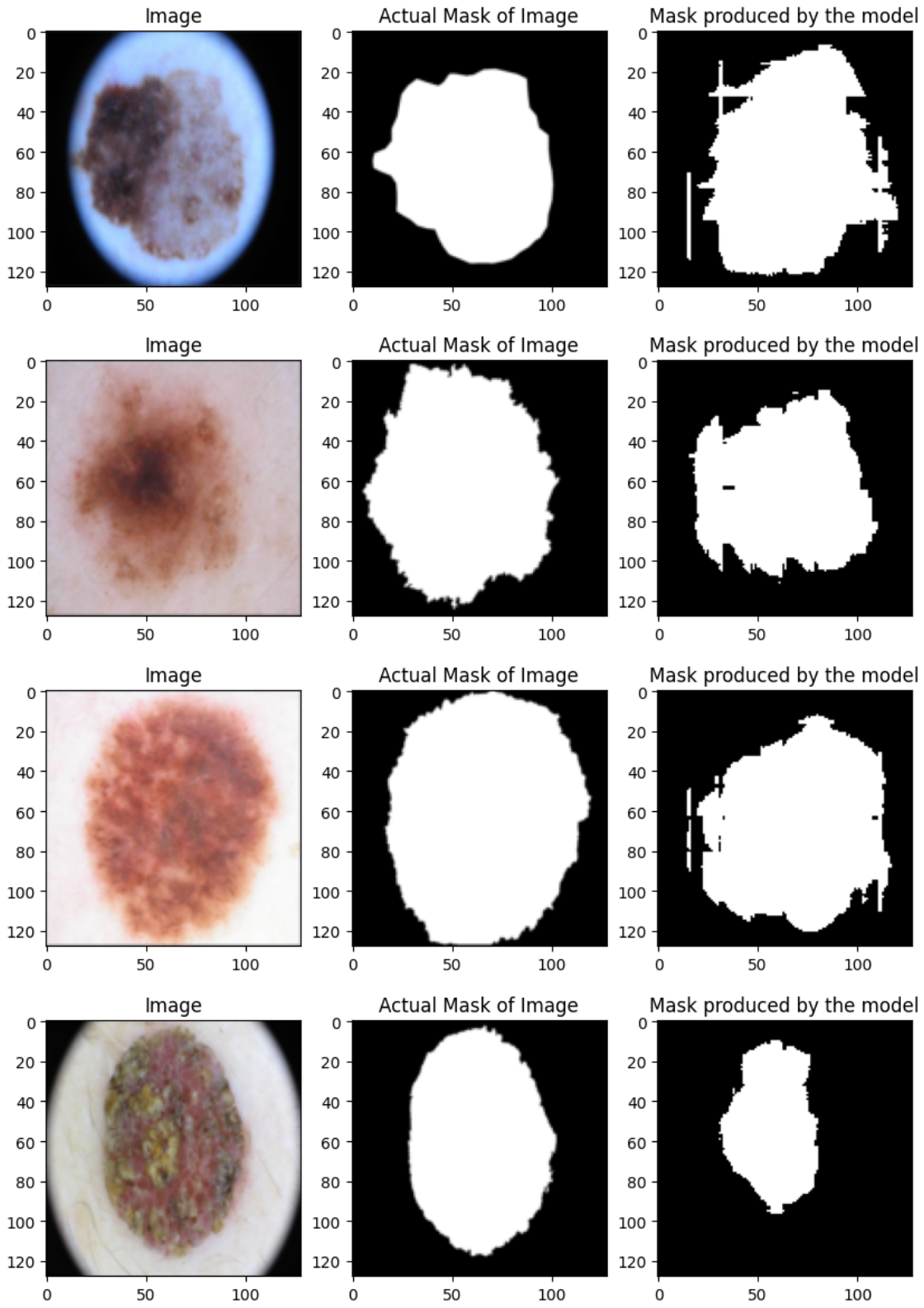


Figure 8: Actual Masks v/s Predicted Masks for Task 1



8. Number of parameters:

   - The number of parameters (involved in our task) in Encoder are:

Parameter Type	Value
Total Parameters	2223872
Trainable Parameters	0
Non-trainable Parameters	2223872

Table 3: Encoder Parameters for Task 1

- The number of parameters in Decoder are:

Parameter Type	Value
Total Parameters	13272065
Trainable Parameters	13272065
Non-trainable Parameters	0

Table 4: Decoder Parameters for Task 1

3.2 Task 2: Fine Tuning the Encoder weights

1. The following table contains the recorded information per epoch for 20 epochs:

Epoch	Training Loss	Validation Loss	Test Loss	IoU Score	Dice Score
1	0.5996	0.3378	0.3388	0.5764	0.7313
2	0.2936	0.2436	0.2712	0.6152	0.7618
3	0.2098	0.2199	0.2661	0.6504	0.7882
4	0.1656	0.2254	0.2280	0.6900	0.8165
5	0.1501	0.1902	0.2310	0.6637	0.7979
6	0.1358	0.1907	0.2044	0.6940	0.8194
7	0.1235	0.1797	0.2030	0.7053	0.8272
8	0.1207	0.2518	0.2485	0.6712	0.8033
9	0.1167	0.1800	0.2094	0.7013	0.8245
10	0.1070	0.2040	0.2458	0.7044	0.8266
11	0.1061	0.2243	0.3013	0.6998	0.8234
12	0.1031	0.1780	0.2134	0.6845	0.8127
13	0.0908	0.1694	0.2121	0.6969	0.8214
14	0.1049	0.1872	0.2148	0.7063	0.8279
15	0.0976	0.1564	0.1895	0.7076	0.8288
16	0.0926	0.1818	0.2294	0.7121	0.8319
17	0.0946	0.1691	0.1896	0.7143	0.8334
18	0.0908	0.1772	0.2021	0.7165	0.8348
19	0.0747	0.2304	0.2386	0.7083	0.8293
20	0.0866	0.2117	0.3027	0.7018	0.8248

Table 5: Task 2: Model Performance

2. The values of the metrics after training phase is:

Metric	Value
Mean Train Loss	0.1482
Mean Validation Loss	0.2054
Mean Test Loss	0.2370
Min IoU Score	0.5764
Max IoU Score	0.7165
Mean IoU Score	0.6860
Min Dice Score	0.7313
Max Dice Score	0.8348
Mean Dice Score	0.8132
Mean Pixel wise Accuracy	0.8973

Table 6: Metrics of Model on Task 2

3. The plot of training, validation and test losses is as below:

Figure 9: Plot showing Training, Validation and Test Loss

4. The plot of IOU score per epoch:

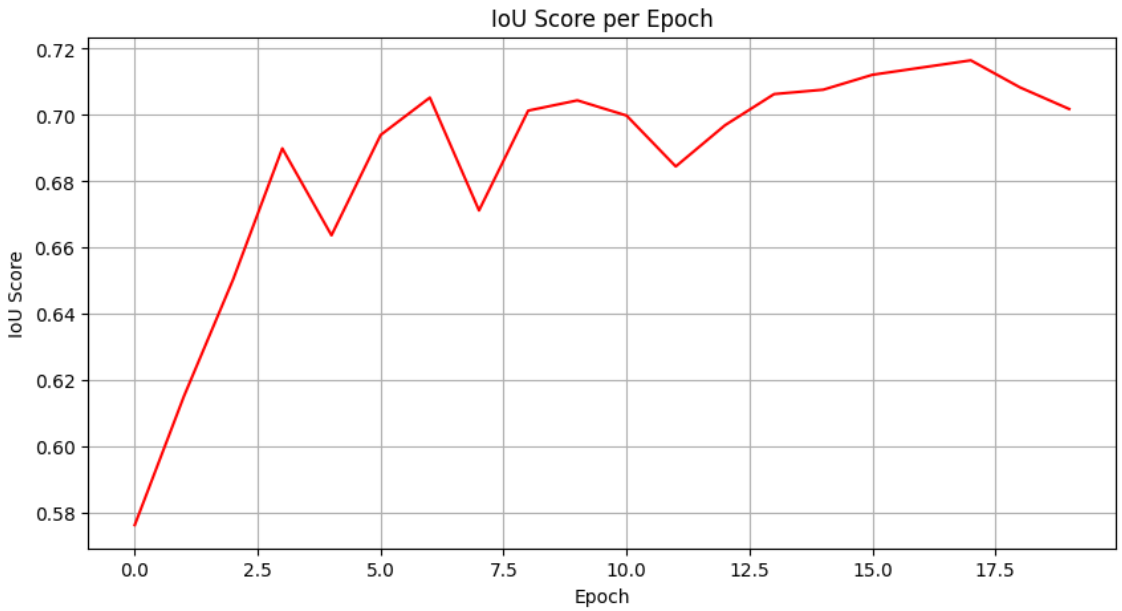


Figure 10: IOU Score per epoch

5. The plot of Dice score per epoch:

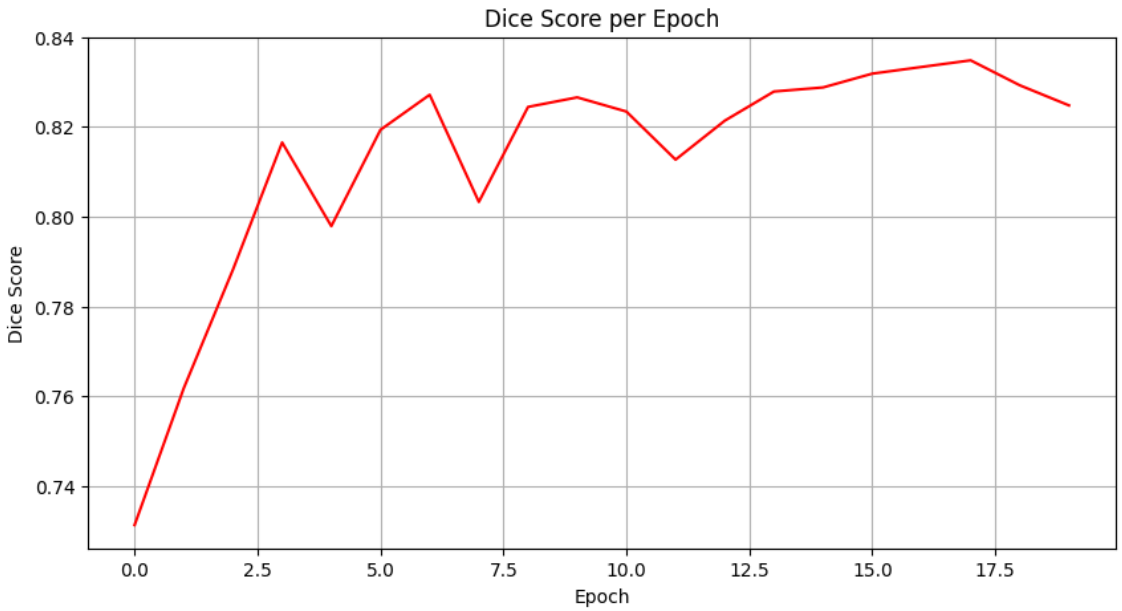


Figure 11: Dice Score per epoch

6. The plot of pixel wise accuracy:

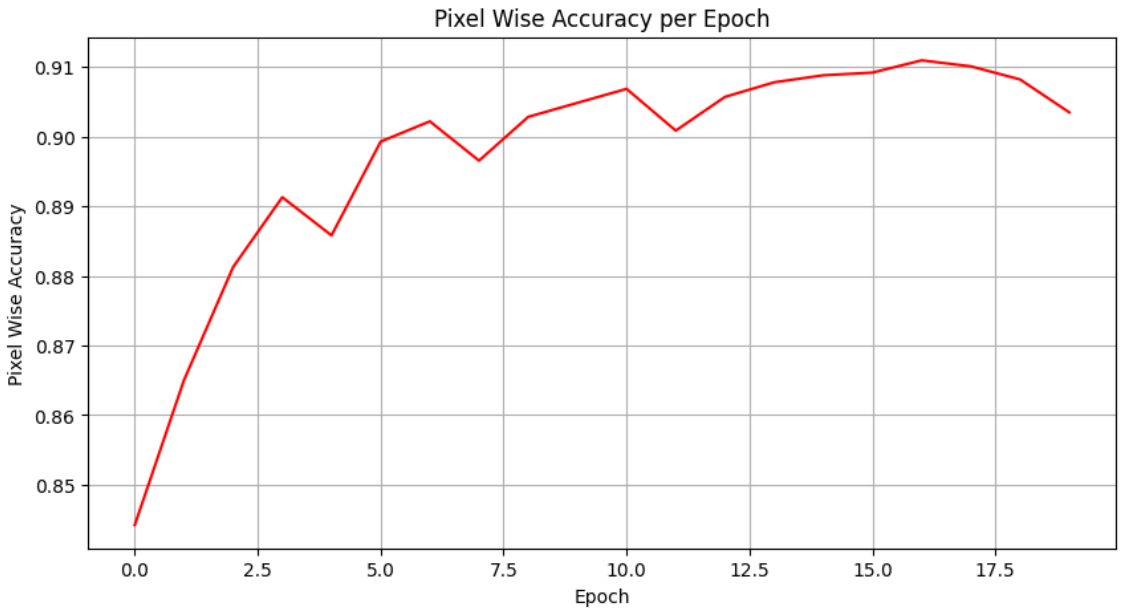


Figure 12: Pixel Wise Accuracy

7. Visualisation of the predicted segmentation masks with the corresponding image and ground truth i.e. Actual mask.

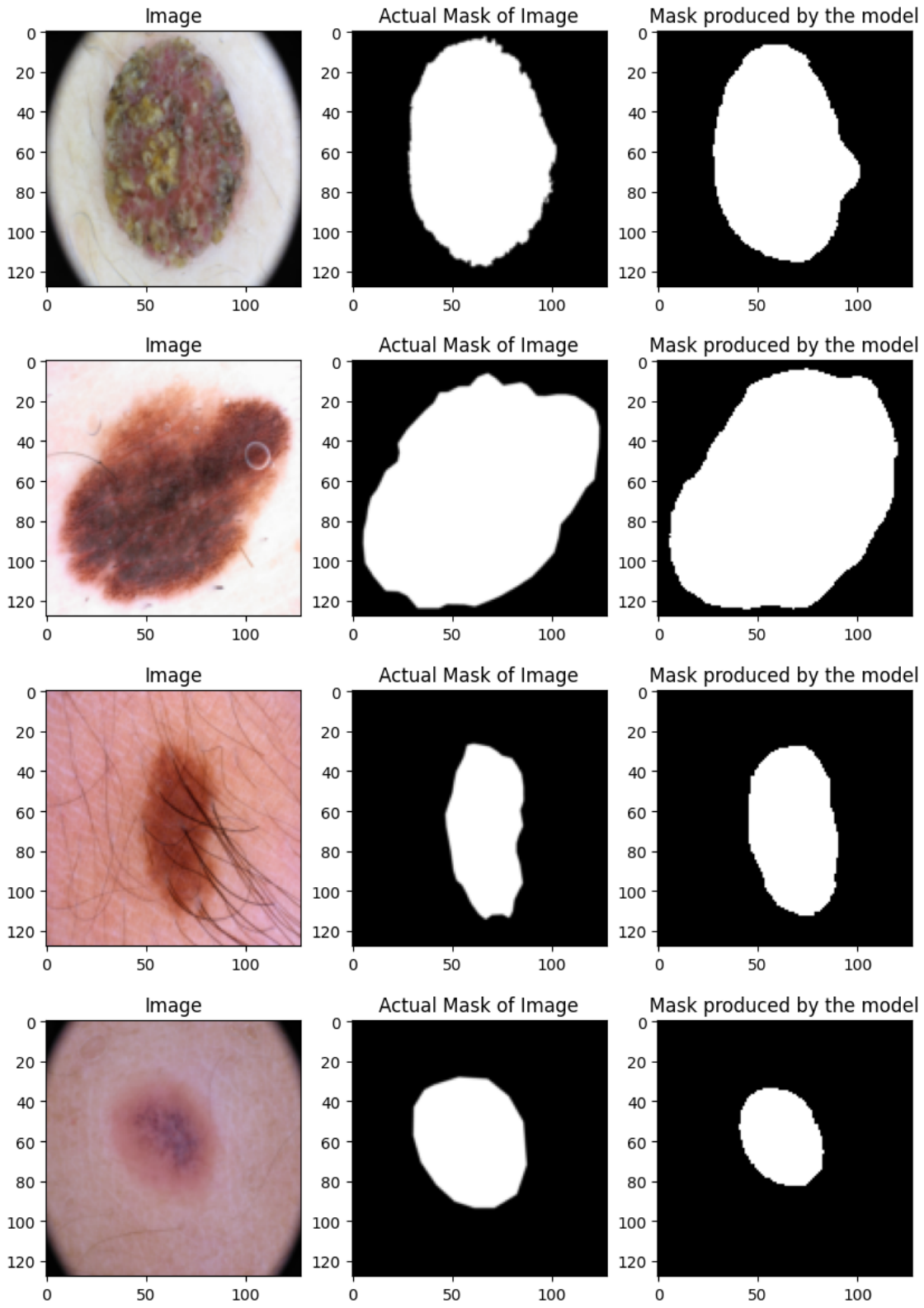
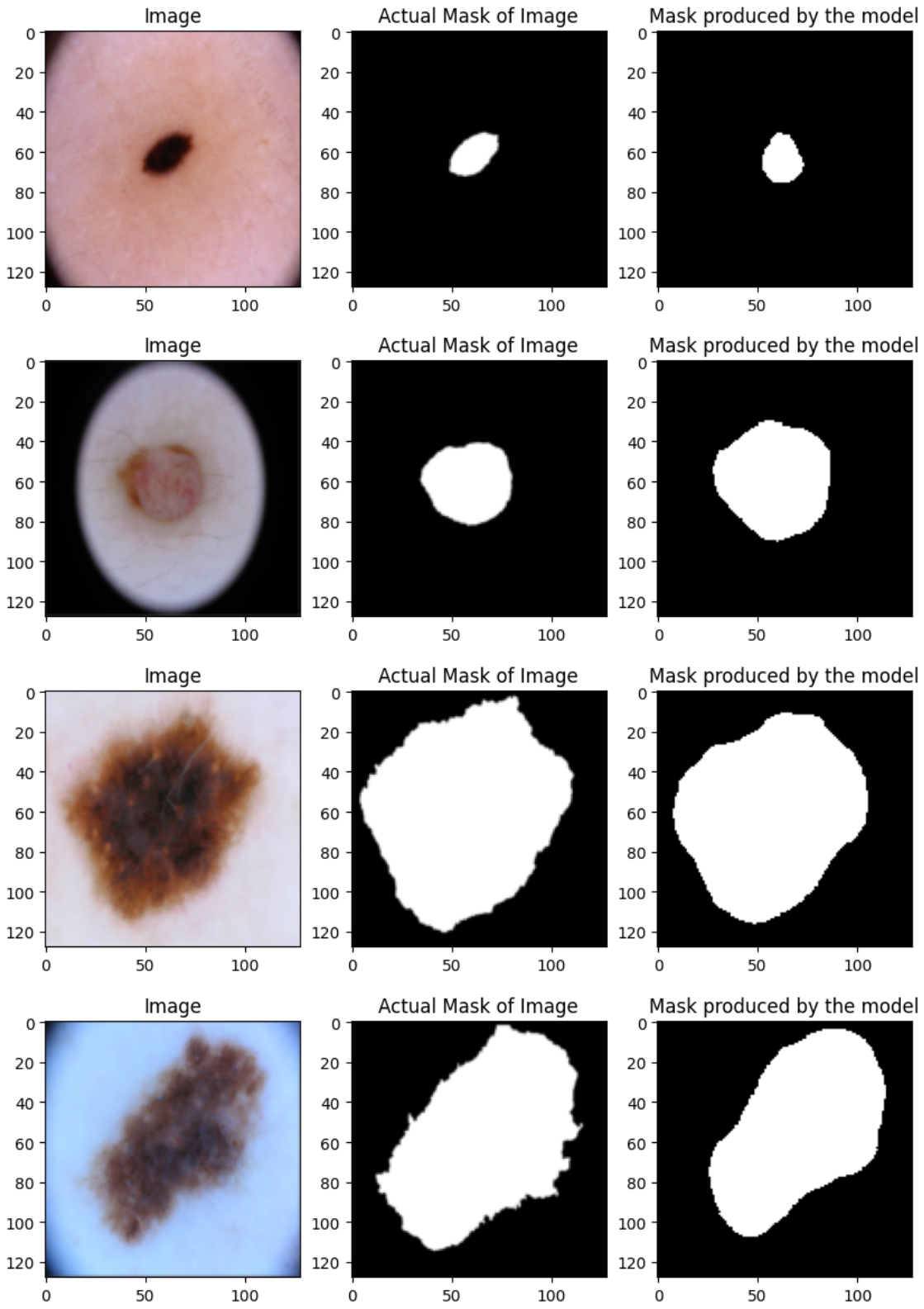


Figure 13: Actual Masks v/s Predicted Masks



8. Number of parameters:

   - The number of parameters (involved in our task) in Encoder are:

Parameter Type	Value
Total Parameters	2223872
Trainable Parameters	2223872
Non-trainable Parameters	0

Table 7: Encoder Parameters for Task 2

- The number of parameters in Decoder are:

Parameter Type	Value
Total Parameters	13272065
Trainable Parameters	13272065
Non-trainable Parameters	0

Table 8: Decoder Parameters for Task 2

4 Comparative study and Analysis of Results

- The performance of both the architectures can be compared based on the following metrics

Metric	Architecture 1	Architecture 2
IoU Score	0.5585	0.7165
Dice Score	0.7167	0.8348
Accuracy	0.8150	0.8973

Table 9: Comparison of Architectures

- In the task 1, we are using mobile net as encoder and using its pre trained weights, we are not training the weights of encoder, thus in this way we are transferring the learned knowledge of Mobile Net trained on ImagenetV1 to our model.
- This acts as base model for our architecture which is responsible for extracting the features from our input images and then we designed a decoder on top of encoder which understands the extracted features and creates our task of segmentation.
- Since the mask are of one channel, gray scale, pixels needs to be classified as 0 or 1.
- This technique is also called as ”Off the shelf method”.
- Since encoder model is trained on different dataset for classification task and our task is also different, so the complete model is not performing well when we are just taking the pre trained weights without updating them.

- Hence the IOU and dice scores for 1st task is less compared to 2nd task, where we allow encoder weights to update with small learning rate.
- This technique is called" Fine tuning".
- Thus by above analysis and results obtained for both the architectures we can observe that **fine tuning is more better than off the shelf method**.
- Updating the encoder weights enables to produce high quality segmentation masks.
- Though there are several fluctuations in the scores during training, still mean scores seems to be quite acceptable.
- Pixel wise accuracy also increases when we adapt 2nd architecture.
- We can utilize the plots above, for comparing the performance of both the architectures per epoch.
- The losses are keep on decreasing per epoch except at some instances.
- Overall, second architecture outperforms first.

5 Link to Notebook

- **Google Collab Notebook link:-** Click here to navigate to Google Colab Notebook
- **File containing all parameter values for both the tasks:-**
  Click here to navigate to text file containing all the parameters values

6 Resources Used

- https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html
- https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html
- https://segmentation-models-pytorch.readthedocs.io/en/latest/models.html
- Lecture Slides provided by the instructor

********** THE END **********