Assignment 3
Deep Learning

Submitted By:

Gyanendra Chaubey
M.Tech (AR-VR)
Roll No: M23AIR005

Submitted To:

Dr. Deepak Mishra
Assistant Professor
Dept. of CSE

1. Methodology:

In this work, segmentation of the dermoscopic lesion images has been done. For this work, MovilenetV2 has been used as the encoder network and a customised decoder has been created to efficiently produce the mask for the given lesion image. This dataset has been chosen from ISCI 2016 challenge on the skin. This dataset contains the 900 train images and 379 test images along with their masks. Originally, the image size is 1022x767 with their masks. The images are in RGB colour space format and masks are given in the GrayScale format. After loading the image, it has been ensured that images are correctly loaded in the RGB format. For the masks a thresholding of pixel values has been done between 127 and 255. Afterwards, augmentation has been applied for horizontal flip and vertical flip along with resizing of images and masks to 128x128. Then, pixel values have been normalized between 0 and 1.

After reading the data, I have plotted few samples from the test data as shown in Fig. 1.

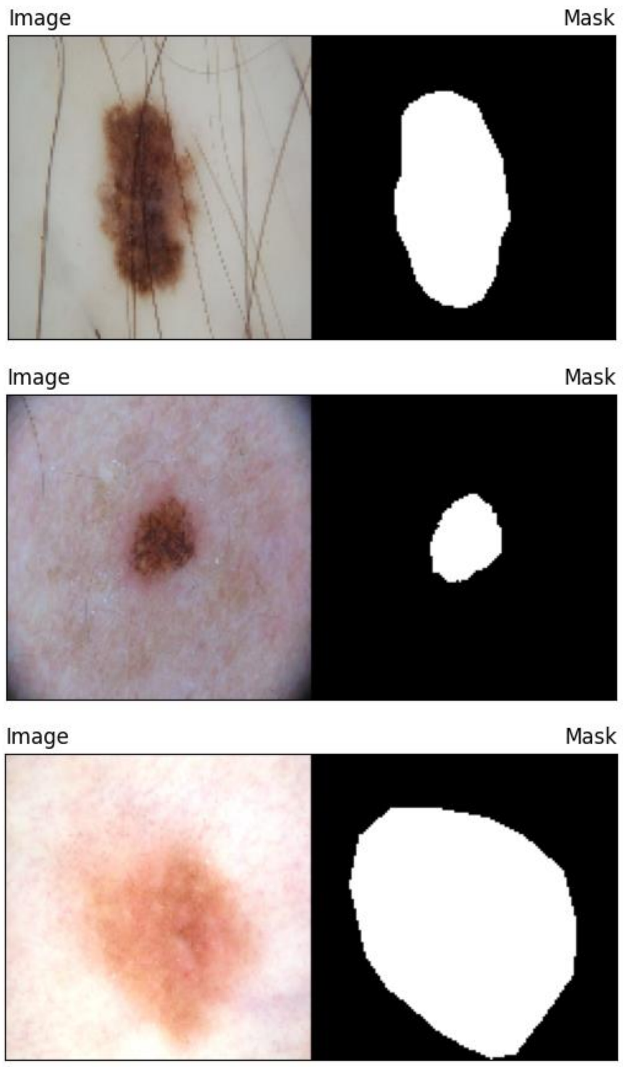

Fig. 1. Sample Images from Test Dataset

In this work, model has been trained for segmentation using two different approaches of transfer learning. Firstly, the model has been trained using the **Off-shelf training** strategy and then it has been trained using the **fine-tuning** methodology. But before discussing the training part lets first discuss about the designing of architecture of the model.

1.1 Designing the Network Architecture:

The model architecture for this segmentation task has been designed in the encoder decoder architecture format. For this task, pretrained layers of MobileNetV2 (trained on ImageNet) has been used as an encoder. The output of the pretrained architecture contains 1280 feature maps, these features serve as an input for the decoder of the model. Decoder has been designed using the 5 transpose convolution layers followed by Batch normalization, relu activations and dropout at rate of 0.5. At the last, a 2d convolution layer is added to generate the mask. The architecture of the decoder is given below in Table 1.

Table 1. Architecture of the decoder

```
(decoder): Sequential(
   (0): ConvTranspose2d(1280, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
   (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): ReLU(inplace=True)
   (3): Dropout(p=0.5, inplace=False)
   (4): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
   (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (6): ReLU(inplace=True)
   (7): Dropout(p=0.5, inplace=False)
   (8): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
   (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (10): ReLU(inplace=True)
   (11): Dropout(p=0.5, inplace=False)
   (12): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
   (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (14): ReLU(inplace=True)
   (15): Dropout(p=0.5, inplace=False)
   (16): ConvTranspose2d(64, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
   (17): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (18): ReLU(inplace=True)
   (19): Dropout(p=0.5, inplace=False)
   (20): Conv2d(32, 1, kernel_size=(1, 1), stride=(1, 1))
  )
```

1.2 Off Shelf Method

In this method, encoder of the model is kept at the learning rate of zero means feature extracting layers of mobilenetv2 has been freeze for any kind of weight update. Decoder has been given a learning rate of **0.0001** which serves the purpose for the off-shelf transfer learning. The model has been trained to the **20 epochs**. Binary Cross Entropy with logits has been used as a loss function for training and testing. Dice Score and Intersection over Union (IoU) has been used for the evaluation of the model in each step. Adam optimizer has been used for the optimization of model.

1.3 Fine Tuning of Model

In this method, encoder and decoder of the model has been given different learning rates to fine tune the model. Encoder has been given a learning rate of **0.001** and decoder has been given a learning rate of **0.001**. Here, in this case encoder has been given a higher learning rate so that model can be fitted at a good rate. The model has been trained to the **20 epochs**. Binary Cross Entropy with logits has been used as a loss function for training and testing. Dice Score and Intersection over Union (IoU) has been used for the evaluation of the model in each step. Adam optimizer has been used for the optimization of model.

1.4 Loss and Score functions

In this work, Binary Cross Entropy with logits has been used for the training of the model. The output of the decoder layer is the predicted mask but containing the feature map values hence an activation function like sigmoid is needed to get the actual mask for comparison. Since, here only two classes need to be compared one is actual mask and second one is predicted mask that's why chosen binary cross entropy. Although, Binary cross entropy works well with the uniformly distributed data. Also, for the evaluation of the model Dice Score and Intersection over Union (IoU) has been used. The equation for the BCEwithLogits, Dice Score and IoU score given as Eq.1, 2, 3 respectively.

$$H(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i * \log\big(\sigma(\hat{y})\big)\Big) + (1-y_i) * \big(1-\log\big(\sigma(\hat{y})\big)\big) \qquad (1)$$

$$Dice\ Score = 2 * \frac{y \cap \hat{y}}{y \cup \hat{y}} \qquad (2)$$

$$IoU\ Score = \frac{y \cap \hat{y}}{y \cup \hat{y}} \qquad (3)$$

2. Results

The results shown in the below two approaches are the best obtained results after several iterations of modifications either in the architecture or in the hyper parameters of the model. For each of the method, accuracy is given below.

2.1 Off-shelf Method

In this method, after training the model till 20 epochs the model seems to be converging to a good scale. In Fig. 2, it can be observed that the model is converging to a good scale in case of the training to the 20 epochs and will improve its performance even if there is increase in the epochs. But the loss curve is decreasing in the form of plateaus and is converging with very little difference to the training loss.

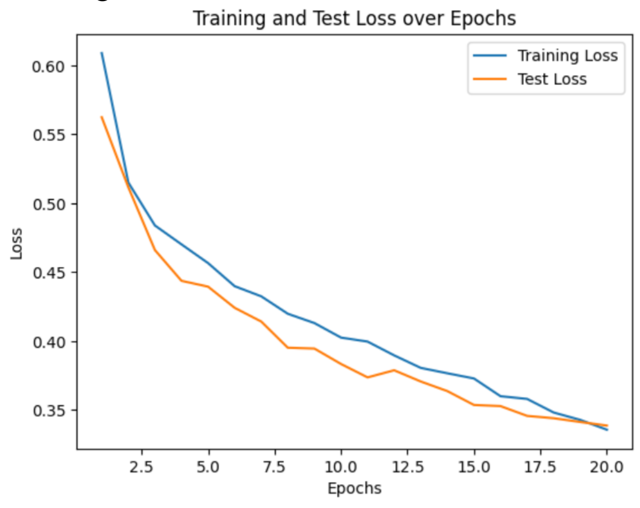

Fig. 2 Training and Testing loss

In the Fig. 3, Initially, at epoch 2.5, the Dice Score is around 0.35. As training progresses, the Dice Score rapidly improves and reaches approximately 0.65 by epoch 5. After epoch 5, the Dice Score fluctuates between approximately 0.55 and 0.65, indicating that the model's performance stabilizes. The rapid initial improvement suggests that the model is learning well from the training data. The subsequent oscillations indicate that the model might be fine-tuning its predictions without significant gains.

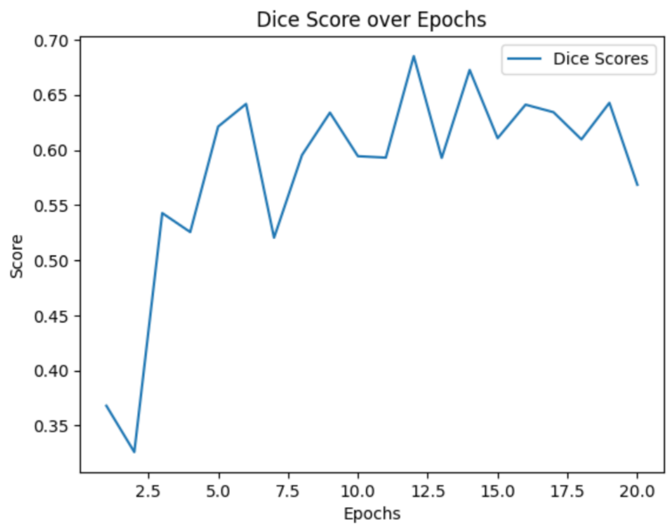

Fig. 3 Dice Score for Off-shelf learning

In Fig. 4, Initially, at epoch 0, the IoU Score is around 0.30. As training progresses, the IoU Score rapidly improves and reaches approximately 0.55 by epoch 5. After epoch 5, the IoU Score fluctuates but generally stays between values of approximately 0.45 and 0.55.

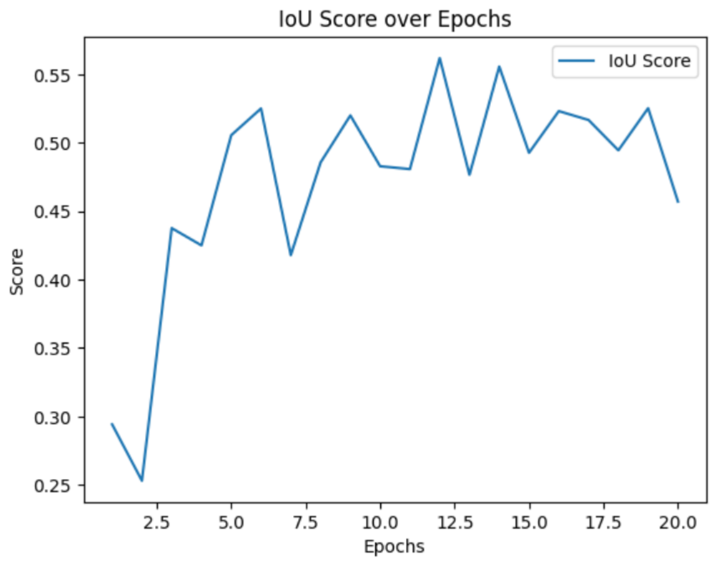

Fig. 4 IoU Score for Off-shelf learning.

Segmentation results of the off-shelf learning are shown in the Fig. 5.

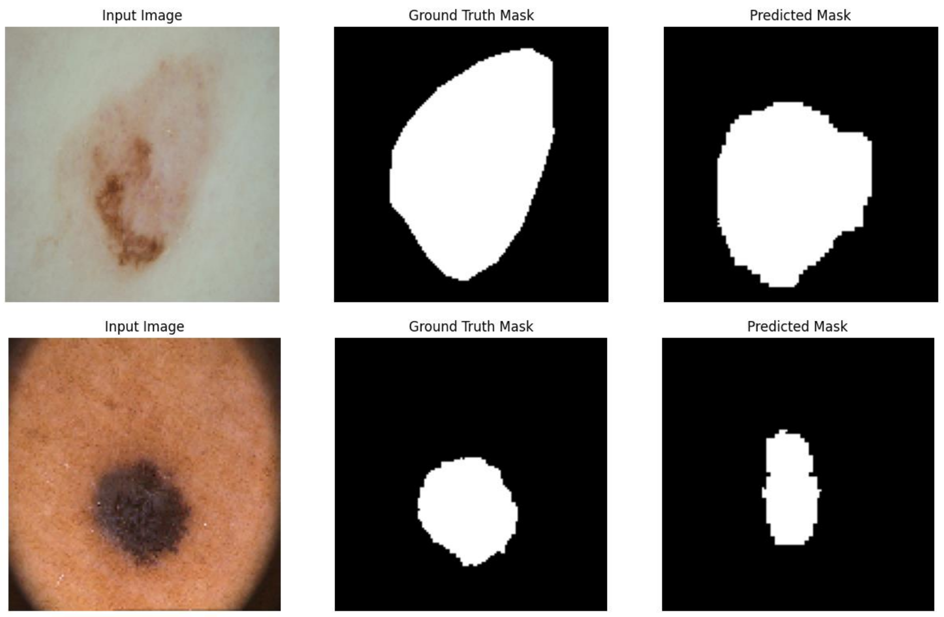

Fig. 5 Results of segmentation in Off-Shelf Learning

2.2 Fine-tuning Method

After applying the finetuning of the model, the loss converges at faster rate. In Fig. 6, it is shown that initially the testing loss was bit high but gradually decrease and matches the training loss. As the number of epochs (training iterations) increases, the training loss decreases. There is some fluctuation in the testing loss can be seen but model getting converged. If the model keeps on training for a greater number of epochs model may go for overfitting.

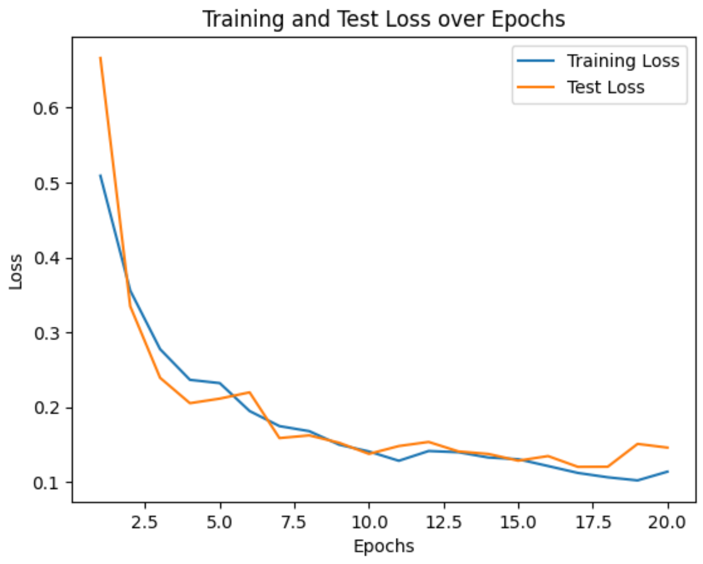

Fig. 6 Training and testing loss for fine-tune method

In Fig. 7, dice score is converging at a good scale. After running till the 20 epochs model is able to achieve the about 90% of score.

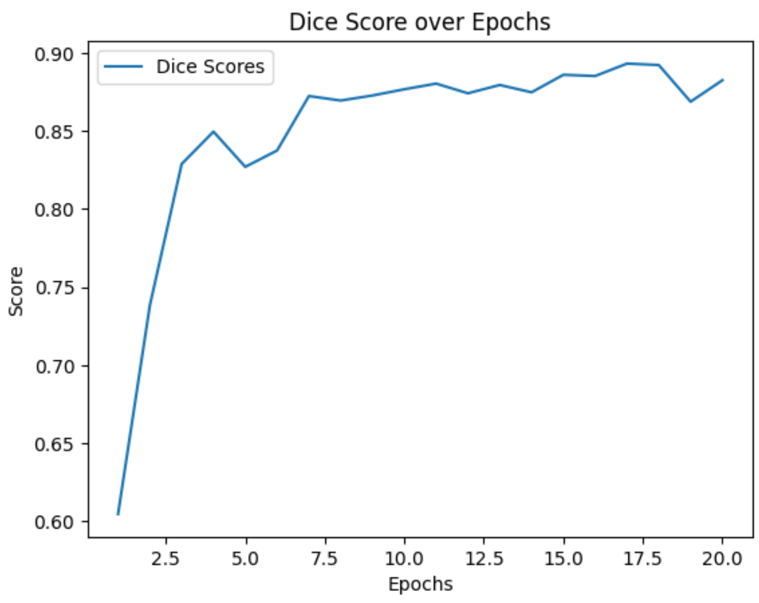

Fig. 7 Dice Score for fine-tune method

In Fig. 8, IoU score is converging at a good scale. After running till the 20 epochs model can achieve the about 80% of score.

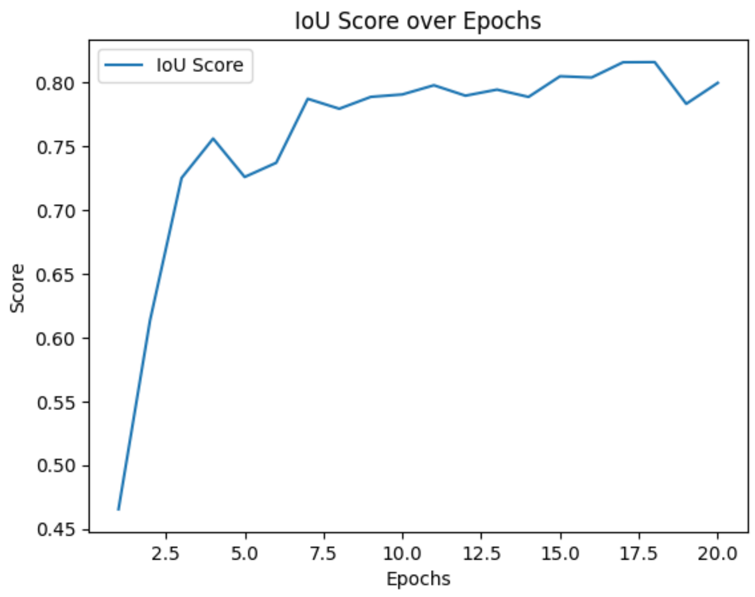

Fig. 8 IoU Score for fine-tune method.

Segmentation results of the off-shelf learning are shown in the Fig. 9.

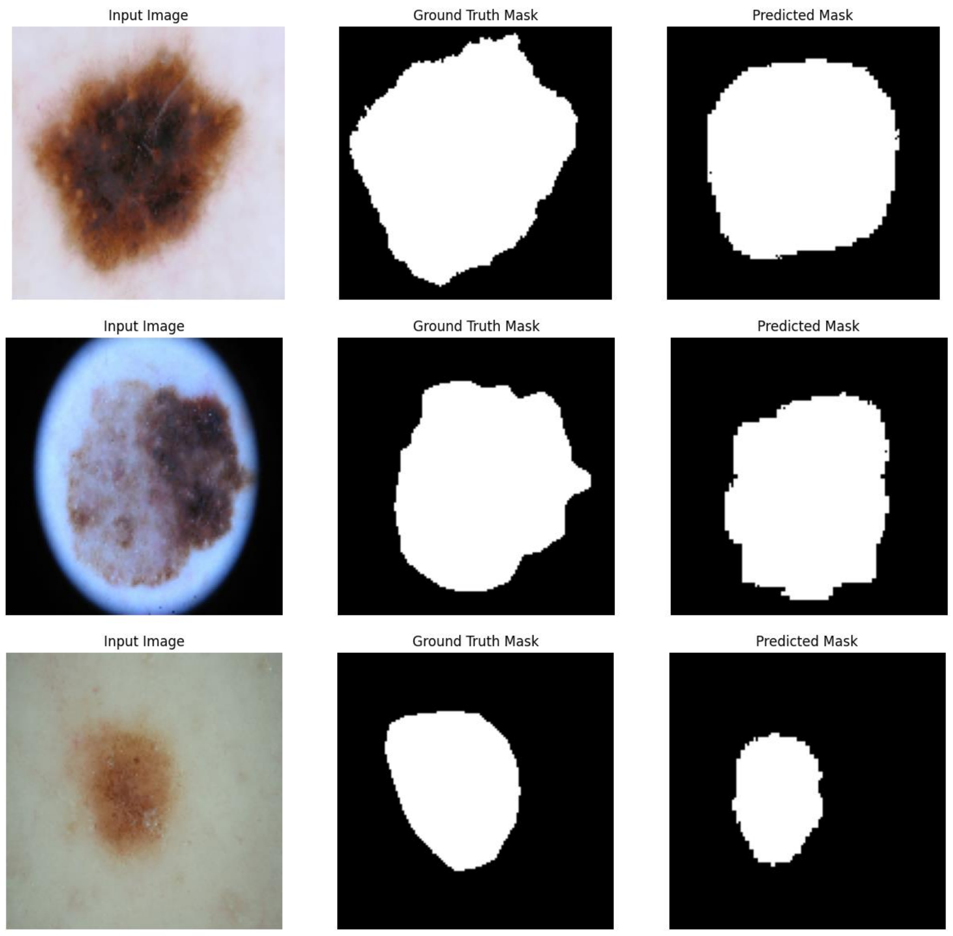

Fig. 9 Results of segmentation in Fine tune Learning

3. Discussion and Conclusions:

This work is intended to do the segmentation of the lesion images using the encoder decoder model. There is need to implement the off shelf and fine tune transfer learning methods. Both the methods have been used and a good dice score of about 90% on the fine tune method is achieved. Although, off shelf method is not able to perform well.

Colab link: **Colab**