Supervised Skin lesion Segmentation on HAM10000 Dataset

Sriram S M

January 9, 2023

1 Introduction

The task at hand is perform binary segmentation on skin lesion images to identify and localize the affected areas in the image. By performing segmentation, the results can further be used for downstream tasks like skin cancer classification and automatic diagnosis.

Segmentation is a technique used in computer vision to identify objects or boundaries to simplify an image and more efficiently analyze it. It is performed by dividing an image into different regions based on the characteristics of pixels. Binary Segmentation is used to divide 2 major regions. In our case, these regions are the cancer affected and unaffected parts of the skin in the image. The segmented images are 2D, are also called masks and can have pixel intensities of either high(1) or low(0).

2 Data Set

The dataset used for this task is the HAM10000 dataset, which consists of 10015 RGB images of skin cancer and corresponding type of cancer. It also includes the ground truth of segmentation maps or masks, localizing the affected region, which were provided by dermatologists. The skin images are of size (3,450,600) and 2D masks are of size (450,600).

For the current task, we use these masks to train our model in a supervised setting and test the model on a disjoint subset of the dataset. The type of cancer is not used for the segmentation task.

2.1 Data Cleaning

It was noticed that some of the ground truth masks had their pixels inverted. The unaffected areas had a pixel intensity of 1 and affected areas had 0. This was fixed by just inverting back the values of the pixels, i.e, 0 to 1 and 1 to 0.

3 Approach

A high level idea for the approach used is as follows:

- Model - To perform the task of binary segmentation a Convolution Neural Network is used. The specifics and architecture of the model is described later
- Input - The input to our CNN would be an RGB image of from the dataset resized to 128x128.
- Output - The output would be a 2D mask image of the size 128x128.
- Loss function - Dice loss is used as the loss function for this model. It is a common metric used in binary segmentation.
- Tracking metric - Jaccard Index/Intersection over Union is another popular metric in binary segmentation.

3.1 Model

The python library segmentation_models_pytorch was used to easily create and handle the CNN model.

segmentation_models_pytorch allows us to create a model by combining encoder and decoder architectures and pretrained weights from popular CNN architectures like ResNet, VGGNet, UNet, etc. The model which was used for the task had:

- Unet architecture: To exploit the strength of encoder-decoder framework and skip connections from encoder to decoder. The encoder converts the image to a concise lower dimensional shape by using a series of convolutional layers and max pooling layers. The decoder then converts the concise shape to the required output shape through a series of convolutional and upscaling layers. This framework is found to be effective because converting the information to a lower dimensional vector removes noise and helps better training on important features. The additional skip connections are found to increase the quality of the output.

- Resnet34 encoder and decoder with pretrained weights on ImageNet: To help with good initialization. ImageNet is a popular image dataset used in competitions and for comparing models. By using pretrained weights we are ensuring the model's ability to identify important features to start with. This helps us use lesser data and lesser training time since most of the work is done and only finetuning to our task is needed.

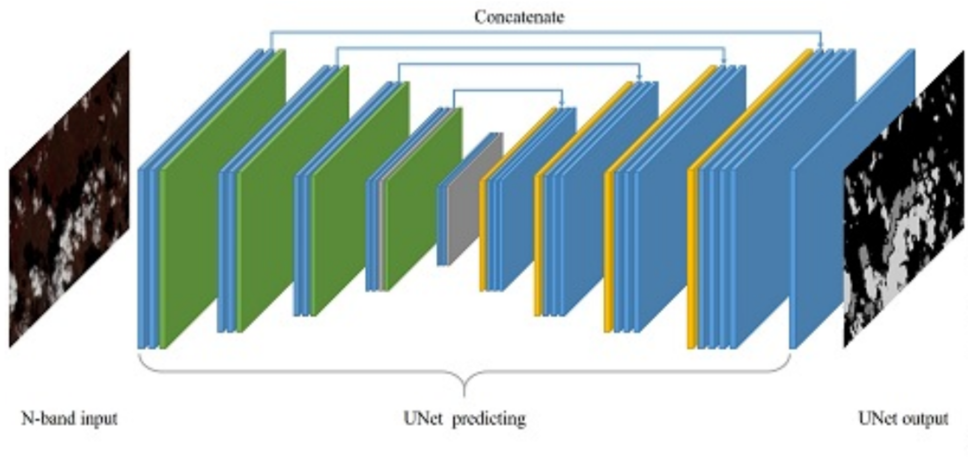


Figure 1: UNet architecture - Encoder Decoder with Skip connections

3.2 Input and Output

The input images of shape (3,450,650) were resized to (3,128,128) as the encoder-decoder architecture of CNNs easily handle sizes of power of 2. Higher quality images are computationally expensive, but downsampling too much will lose important features. So 128 was chosen to balance the trade off between image quality and computational time.

The output masks and the ground truth masks too were set to the same size (3,128,128).

3.3 Loss function and metrics

The quality of results from a machine learning model is highly influenced by what metric we use to evaluate and assess the model. Different evaluation metrics and loss functions can be used to test our model performance. In any prediction task, for every data point, we have a ground truth which is the expected correct output and a corresponding prediction that the model outputs. The purpose of an evaluation metric is to give us an idea of how close the prediction is to the ground truth. Now that we have a qualitative intuition behind what the purpose of a metric is, we use the prediction and ground truth to make up a quantitative formula. Let us suppose a classification task of 2 output classes. So for every data point, the output can be either of the 2 classes, say 0 (Negative class) and 1 (Positive class). Now for every data point, we have a corresponding ground truth and a prediction. We can split the dataset into 4 parts:

- True Positives (TP): The number of data points where both ground truth and prediction was class 1.
- True Negatives (TN): The number of data points where both ground truth and prediction was class 0.
- False Positives (FP): The number of data points where ground truth was class 0 and prediction was class 1.
- False Negatives (FN): The number of data points where ground truth was class 1 and prediction was class 0.

Just by looking at these measures we see that our aim would be to maximize TP and TN and minimize FP and FN.

'Accuracy' is a well known metric and is used in a lot of prediction applications. Accuracy can be simply defined as the percentage of correct predictions out of all the predictions. Using the above mentioned terms, the formula for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

It looks all plain and straightforward till now! But we now face 2 problems which we present and solve sequentially.

The first problem is that the task at hand is of binary segmentation and not binary classification. For every data point the output is not a binary class but an image! We are tasked with comparing a ground truth image with a predicted image. An image is just an array of pixels intensity values. In our case, the output images are masks in which pixel intensities can either be 0 or 1. So now we just have both ground truth and predicted values of an array of pixels whose values are either 0 or 1. This looks very similar to the binary classification stated above. Now we can just use TP, TN, FP, FN over all the pixel values and calculate accuracy.

The second problem we face is that of choice of metric. Accuracy can prove to be ineffective in some applications. For our task, we are interested in predicting the area of affected skin, i.e, we focus more on getting the predictions of one of the classes right, the class being 1 let's say. In this case there are metrics that give better insights.

Jaccard Index or Intersection Over Union (IoU) is simply the ratio of the number activated pixels in the intersection of ground truth G and predicted image S and the union.

$$JaccardIndex(S, G) = \frac{|S \cap G|}{|S \cup G|} \tag{2}$$

where $|.|$ denotes the no.of of activated pixels.

We can rewrite the same in terms of TP,TN,FP,FN as:

$$JaccardIndex = \frac{TP}{TP + FP + FN} \tag{3}$$

Since only TP is there in the numerator and not TN, only correct predictions of class 1 will increase the score. This metric is very intuitive because ideally we want to have as much intersection between ground truth and prediction as possible and we want the union to be as close to the intersection as possible.

A closely related metric to the Jaccard Index is the Dice Score. The Dice Score is a spatial overlap metric which is computed as follows for predicted image S and ground truth G.

$$DiceScore(S, G) = \frac{2 * |S \cap G|}{|S| + |G|} \tag{4}$$

$$DiceLoss = 1 - DiceScore \tag{5}$$

This can be rewritten as:

$$DiceScore = \frac{2TP}{2TP + FP + FN} \tag{6}$$

Either of these metrics can be used for binary segmentation. We have used the Dice Loss for training the model. The model minimizes the Dice Loss thereby maximizing the Dice Score. We have used Jaccard Index for final evaluation.

4 Training

The python library pytorch_lightning was used for training our model.

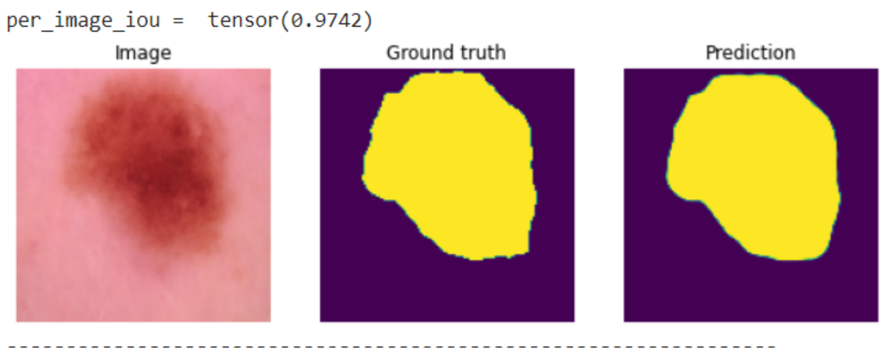
Due to RAM and computational constraints 4000 images were used. Out of these, 95% (3800) was used for training and 2.5% (100) was used each for validating and testing.

The following hyperparameters were chosen after tuning by trial and error:

- Batch size: In batchwise training the model is updated after every batch of data points based on the loss computed using the datapoints of the batch. The batch size was selected to be 16.
- Optimizer: Adam was used as optimizer with initial learning rate of 0.0001. Adam optimizer helps in better convergence by using momentum based gradient descent and adaptive learning rate. It can handle sparse gradients on noisy problems.
- Epochs: This is the number of iterations the model trains through the whole dataset. More the number of epochs the better it fits to the training data. It should be such that the model neither underfits nor overfits. No.of epochs was set to 20.

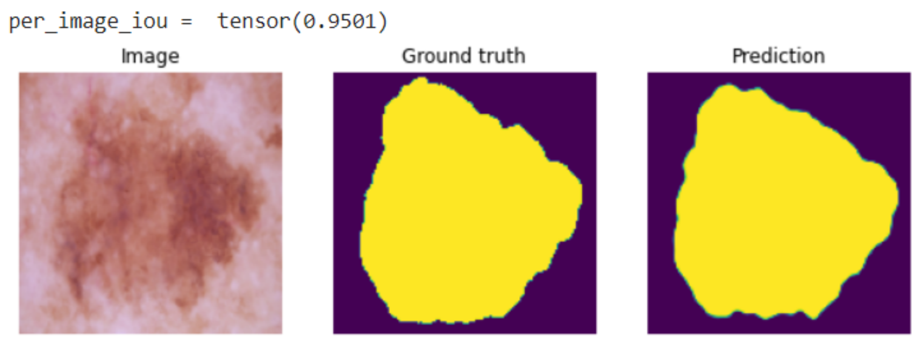
5 Results

After training, the test set was evaluated using the trained model. Jaccard Index was calculated for each sample and averaged over the whole test set. **The mean Jaccard Index was found to be 0.8828**. Some results are visualized in figures 1-4.



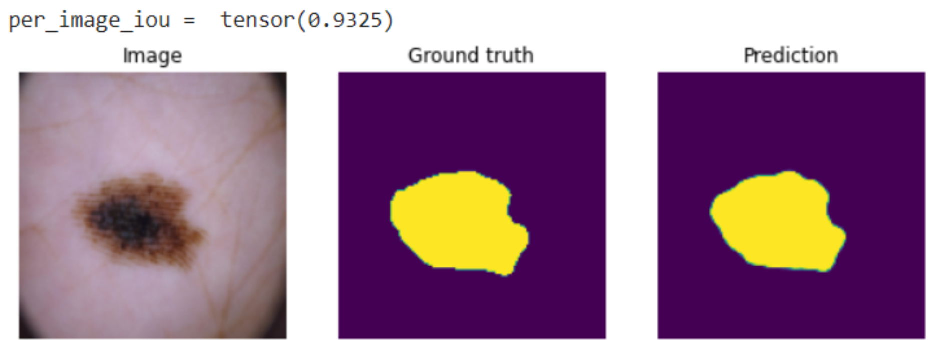
```
per_image_iou =  tensor(0.9742)
```

Figure 2:



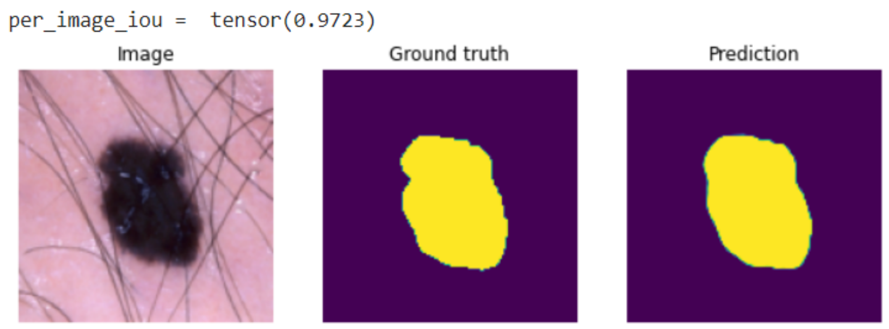
```
per_image_iou =  tensor(0.9501)
```

Figure 3:



```
per_image_iou =  tensor(0.9325)
```

Figure 4:

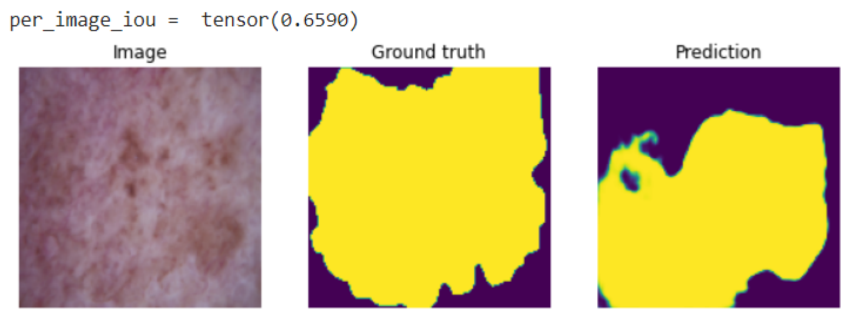


```
per_image_iou =  tensor(0.9723)
```

Figure 5:

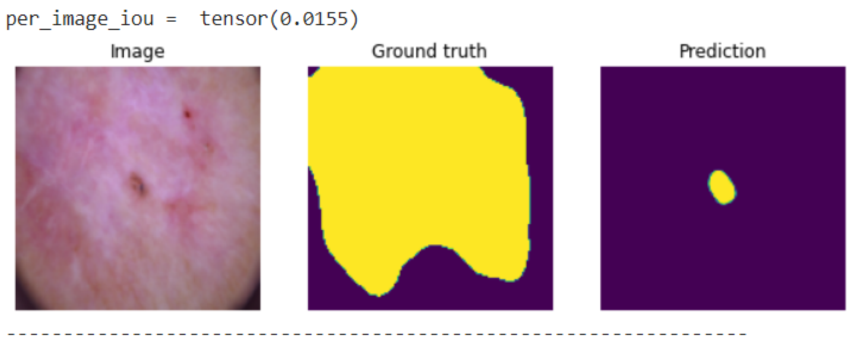
6 Failure cases and shortcomings

Though our model achieved a high Jaccard Index on the test data, the average score was pulled down due to a few specific scenarios. Some of them are mentioned in figures 5-8.



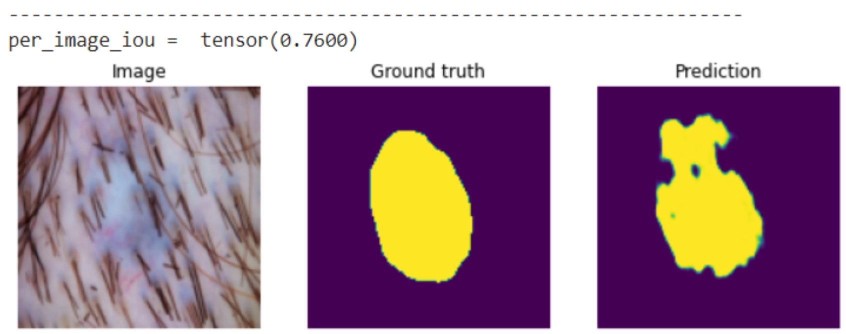
```
per_image_iou =  tensor(0.6590)
```

Figure 6: Almost whole image is the affected part according to ground truth but model predicts partial area



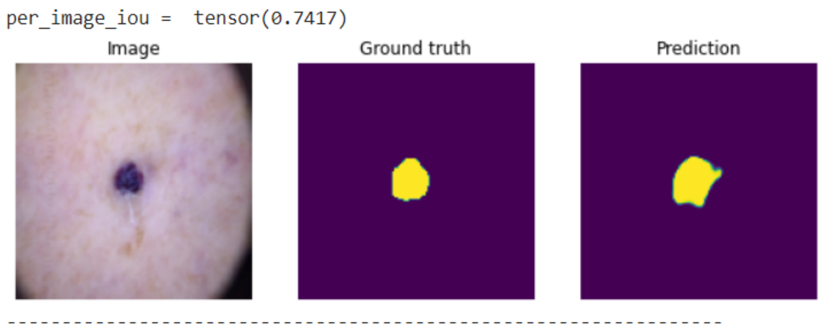
```
per_image_iou =  tensor(0.0155)
```

Figure 7: Almost whole image is the affected part according to ground truth but model predicts area that looks more affected than others



```
per_image_iou =  tensor(0.7600)
```

Figure 8: Presence of hair in the image affects prediction



```
per_image_iou =  tensor(0.7417)
```

Figure 9: A below average score even though prediction looks accurate due to the nature of the metric. If the total no.of activated pixels is low it is more prone to achieve a low score

7 Improvements

The presented model is a base model for the binary segmentation task on the given dataset. Due to time and logistic constraints like RAM and GPU, more exploration and trials were difficult to perform. Higher RAM will allow us to use the complete dataset instead of partially. We can also train the model for a higher no.of epochs with higher model complexity if more time is available.

Further, there are various methods presented in research papers that help increase the score significantly.

One unique and interesting method is presented in 'MFSNet: A Multi Focus Segmentation Network for Skin Lesion Segmentation' by Hritam Basak, Rohit Kundu, Ram Sarkar. They make use of inpainting as a preprocessing step to remove hair or other irrelavant objects from the image. Their approach makes use of attention modules - 2 boundary attention modules and 2 reverse attention modules. Attention mechanisms are widely known for boosting the performance of CNN-based models. In addition to this, instead of a standard decoder with plain upsampling, they use a Parallel Partial Decoder. This helps capture both low level and high level semantics efficiently.

8 References

- https://www.kaggle.com/datasets/surajghuwalewala/ham1000-segmentation-and-classification
- Basak, H., Kundu, R., Sarkar, R. (2022). MFSNet: A multi focus segmentation network for skin lesion segmentation. Pattern Recognition, 128, 108673. https://doi.org/10.1016/j.patcog.2022.108673
- Pham, H. N., Koay, C. Y., Chakraborty, T., Gupta, S., Tan, B. L., Wu, H., Vardhan, A., Nguyen, Q. H., Palaparthi, N. R., Nguyen, B. P., H. Chua, M. C. (2019). Lesion Segmentation and Automated Melanoma Detection using Deep Convolutional Neural Networks and XGBoost. 2019 International Conference on System Science and Engineering (ICSSE). https://doi.org/10.1109/icsse.2019.8823129