Medical Image Segmentation

EE 610: Image Processing

Guide: Prof. Amit Sethi

Anand Bhaskar

200100024

Objective:

The early identification of cancer cells through the analysis of medical images. To accomplish this, the project leverages the UNet model, a robust architecture recognized for its proficiency in image segmentation tasks. The primary aim is to develop a personalized UNet model from the ground up, aiming to precisely identify cancerous cells within medical images sourced from the MoNuSeg Grand Challenge dataset. By addressing this challenge, the project seeks to enhance early diagnosis and facilitate more effective treatment strategies for individuals affected by cancer.

Data Preprocessing:

The project utilized the MoNuSeg Grand Challenge dataset, which presented images of varying sizes. To standardize the input, all images were resized to a consistent 64x64 size. Subsequently, the images were normalized by dividing by 255 and converted to grayscale, given the absence of utility in the RGB channels. The data was then transformed into tensors using TensorFlow, facilitating compatibility with the chosen deep learning framework.

UNet Architecture:

The UNet architecture was meticulously constructed to leverage its segmentation capabilities for cancer cell detection. The core components of the UNet model were designed using three essential functions: Down Block, Up Block, and Bottleneck.

- Down Block Function: This function is responsible for downsampling the input image. It applies two consecutive Conv2D layers with BatchNormalization, followed by a MaxPool2D layer. The result is both the image post-downsampling and the image before pooling, which are essential for subsequent upsampling.
- Up Block Function: The up_block function performs the upsampling operation. It involves concatenating the upsampled feature map with the corresponding feature map from the downsampling path. This concatenated feature map is then processed through two Conv2D layers with BatchNormalization. The final output is obtained after this processing.

- Bottleneck Function: The bottleneck function represents the central layer of the UNet model. It employs two Conv2D layers with BatchNormalization, contributing to the compression of spatial information while preserving important features.

UNet Model Definition:

The UNet model is constructed by sequentially arranging the designed components. It comprises an Input Layer, followed by four down-block layers, a bottleneck layer, four up-block layers, and a final Conv2D layer with a sigmoid activation function.

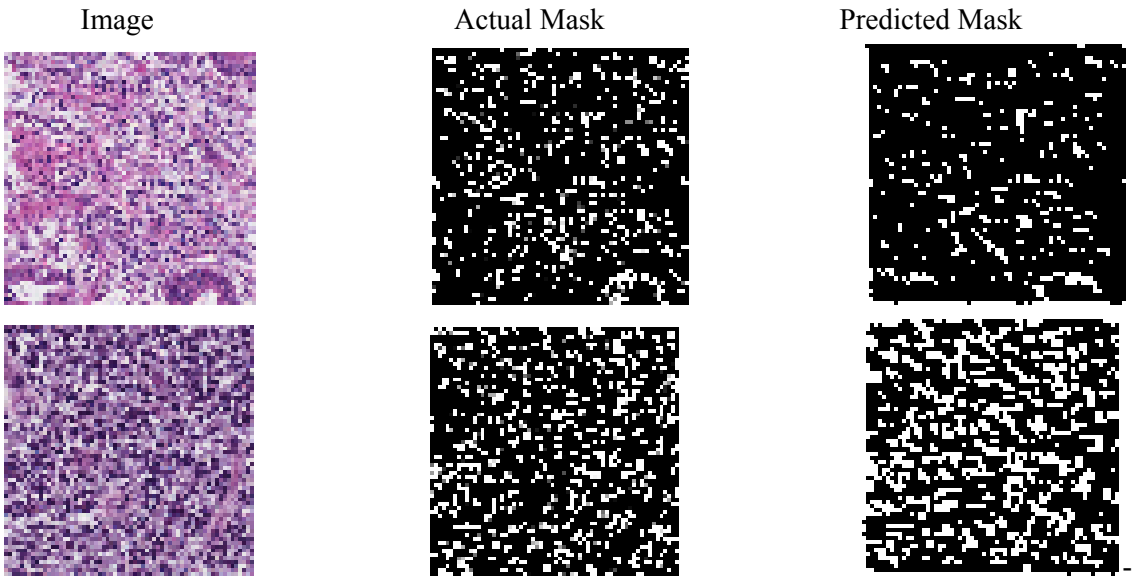
Model Compilation and Training:

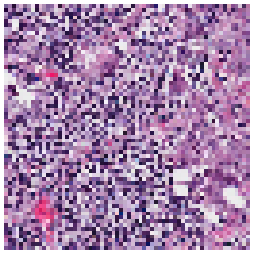
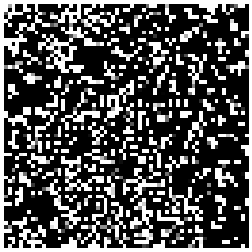
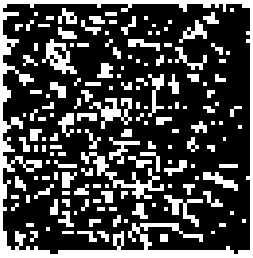
The compiled UNet model utilizes the AdaGrad optimizer and the Dice loss function. Additionally, the F1 score is employed as a metric to gauge the model's performance during training. The model is then trained on the preprocessed dataset, enabling it to learn the complex patterns and characteristics of cancerous cells.

Results:

Upon training completion, the UNet model is evaluated using the test dataset. The model exhibits a promising F1 score of 0.6353, reflecting its proficiency in cancer cell detection within medical images.

Some of the predicted images:



Conclusion:

The project demonstrates the successful implementation of a custom UNet model for cancer cell detection in medical images. The meticulous design of the UNet architecture, along with the incorporation of essential functions for down-sampling, up-sampling, and bottleneck processing, enables accurate segmentation of cancerous cells. The project underscores the potential of AI-powered models in revolutionizing cancer diagnosis, offering the medical community a robust tool to enhance early detection and improve patient outcomes. The achieved F1 score of 0.6353 on the test data showcases the effectiveness of the developed UNet model in this critical medical application.

References:

- To load the dataset from a .zip file
- https://towardsdatascience.com/an-informative-colab-guide-to-load-image-datasets-from-github-kaggle-and-local-machine-75cae89ffa1e
- For using segmentation_models module https://segmentation-models.readthedocs.io/en/latest/api.html
- For finding the proper loss function https://github.com/qubvel/segmentation_models/blob/master/segmentation_models/losses.py
- Tensorflow API https://www.tensorflow.org/api_docs/python/tf/keras/