**Description:**

In this code, I implemented a comprehensive image processing pipeline to analyse a dataset consisting of 1500 images and their corresponding masks. To enhance the quality of the images, I initially applied gamma correction, a technique used to adjust the brightness levels in an image. This pre-processing step helps to improve the visibility of details and enhance the overall image quality.

Following the gamma correction, the next task involved labelling the masks corresponding to each image. I leveraged the pixel values provided in the manual to assign labels to specific regions of interest. Any pixel values that did not match the provided labels were considered as background pixels. This process allowed us to segment the mask into different regions based on their pixel values.

To prepare the labelled masks for training a machine learning model, I performed one-hot encoding on the labelled data. This encoding technique transforms the categorical labels into binary vectors, where each label corresponds to a specific index in the vector. This step is crucial for training models that can handle multi-class segmentation tasks effectively.

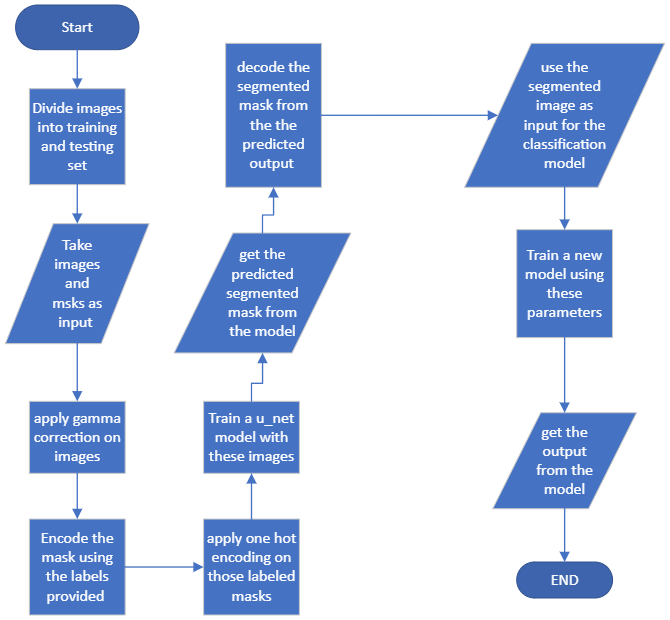
For the segmentation task, I employed a U-Net model architecture. The U-Net is a popular neural network architecture widely used for image segmentation. It consists of five convolutional layers that extract meaningful features from the input images. These convolutional layers are then followed by five convolution transpose layers, also known as up sampling or deconvolution layers, which help to reconstruct the segmented image back to its original size.

To optimize our segmentation model and achieve the best possible results, I experimented with various configurations and training techniques. After extensive training and evaluation, I obtained a satisfactory dice coefficient of 0.8065, which measures the similarity between the predicted and ground truth segmentations. This metric indicates a high level of accuracy in the segmentation task, suggesting that our model successfully captured the relevant features and boundaries in the images.

After obtaining the segmented images, I proceeded to train a classification model using this newly generated dataset. The classification model I employed shares similarities with the model used in our lab, but I made some modifications to adapt it to our specific task. Before feeding the segmented images into the classification model, I first labelled them according to the corresponding classes they represent. Additionally, to ensure unbiased evaluation, I divided the labelled data into separate training and testing sets.

After training the classification model and predicting the test data, I achieved an accuracy score of 0.68. Although there is room for improvement, this accuracy indicates that our classification model can make correct predictions on approximately 68% of the test samples.

**Flow chart:**



**Evaluation parameters:**

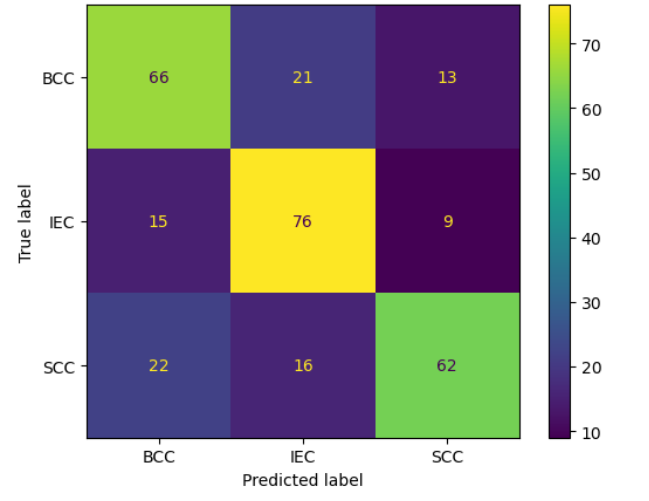
**Segmentation model:**

|  |  |
| --- | --- |
| Total prams | 2,100,000 |
| epochs | 200 |
| Val loss | 0.4897 |
| Val accuracy | 0.7167 |
| mean Igou | 0.3816 |
| dice coefficient | 0.8065 |

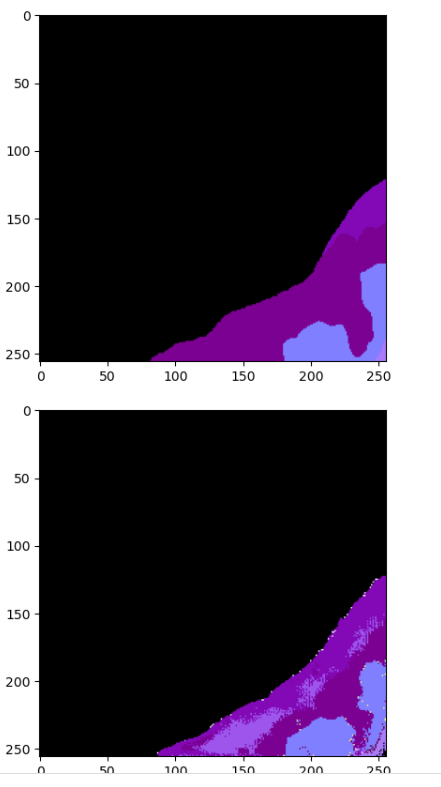
**Classification model:**

|  |  |
| --- | --- |
| Total prams | 495,523 |
| epochs | 200 |
| accuracy | 0.68 |
| Val loss | 1.4551 |
| Val accuracy | 0.7167 |

**Outputs:**



**Segmented outputs original(top) vs predicted(bottom)**



A picture containing map, text, screenshot

Description automatically generated

A screenshot of a graph

Description automatically generated with low confidence