(Group 4) - Code Explanation for Binary Image Classification using CNN

The code starts with importing several python libraries and frameworks that perform different functions like NumPy for arrays, OpenCV for image processing, Tensorflow & Keras for layers, seaborn for confusion matrix, matplotlib for plotting graphs, and sklearn for machine learning models. These libraries are essential in performing various tasks such as data preprocessing, model building, model training, evaluation, and visualization.

Next, the code defines a directory for inputting images into the model with variable names melt_pool_folder and no_melt_pool_folder. It then loads each image using a for loop, resizes them to a fixed size using the cv2 library, and converts them to NumPy arrays. The images are then labeled as either melt pool (1) or no melt pool (0) and saved into an array. The code links the melt pool images with their respective labels, shuffles the data images and labels with the same shuffle indices so that the model sees a random mix of images during each epoch of training. This helps prevent overfitting and can improve the overall accuracy of the model.

The data is then split into training and testing sets using the train_test_split function from the sklearn library. The code uses the TensorFlow.keras framework to build a Convolutional Neural Network (CNN) model for binary image classification, which consists of several layers, including convolutional layers, pooling layers, and dense layers. The code evaluates the performance of the trained model on the testing data using the evaluate method of the tensorflow.keras framework and uses the predict method to make predictions on test images.

Later, the code generates a confusion matrix using the sklearn and seaborn libraries, which summarizes the performance of a classifier by comparing the actual labels of a dataset with the predicted labels generated by the model. The confusion matrix is used to calculate True Positive, True Negative, False positive, and False negative which ultimately used to calculate accuracy, precision, F1 score, recall, as these are function of this terms.

The values of these metrics are then written into an excel sheet using the openpyxl library to analyze and compare the trend of the data. Moreover, to understand the prediction of images various plots are created like scatter plot and histogram using python code. Finally, based on analysis, we fine-tune the model by changing the parameters such as filter size, number of filters, number of images, activation function, size of images and number of conv2D layers to maximize the accuracy of the model.

In the end, we also used various well known machine learning models for Binary Classification like Naive Bayes, Random Forest, Logistic Regression, Decision Tree, K-Nearest Neighbor, Support Vector Machine to compare various model with our model and analysis the performance metrics like F1 score , accuracy , precision and recall and saved in excel file to create graphs.