Criterion A: Planning

Defining the problem

My client Mr xx is a friend and student of Aerospace Engineering at the Delft University of Technology. He has courses in neural network programming and, among them, networks for image recognition – so-called Convolutional Neural Networks (CNNs). Moreover, he uses such networks for his personal projects to primarily group datasets of images into categories; for example, to categorize different types of craters on moons and planets from photos provided by NASA.

However, during our discussion (transcripted in Appendix A), I have found he has been struggling with finding the right parameters for his CNNs. Finding the most suitable parameters for a specific purpose is often a long and arduous process of testing and checking the results. Hence, I have decided to help Mr xx by creating a simple neural network program that would help him optimize CNNs for his projects. The program would first ask the user to input a desired range of parameters in a user-friendly GUI format, train different models of CNNs, and provide a graphical representation of the models' performances. This way, Mr xx can choose the model that fits his project best. I asked my Computer Science teacher about this idea and he agreed.

Rationale for the proposed solution

I decided to use OOP Python because I am familiar with it and the needed Tensorflow Keras libraries for CNN programming are suitable for this language. Additionally, OOP will be useful for grouping all models' training and optimization functions into one class. Moreover, such python libraries as Pickle (for packaging and saving the models), Tkinker (for the user input GUI handling), and matplotlib (for creating graphs) will be extremely useful throughout the project.

Word count: 280

Success criteria

i.	All inputs are presented with a user-friendly GUI
ii.	An input for the user to define the path to the directory with the dataset and where all output data will be stored
iii.	A user input to define the dataset on which the network will be trained
iv.	An input for the user to define the target image size after re-sizing
v.	An input that allows the user to define the rangers for their desired CNN parameters: number of dense layers, number of neurons per layer, number of convolutional layers, kernel sizes
vi.	A set of different models is trained and saved in a "models" folder
vii.	The models' performances are represented visually and saved in a "plots" folder as graphs of training accuracy, training loss, validation accuracy, and validation loss
viii.	Find the best model - with the highest final validation accuracy – and create two additional graphs comparing its training and validation losses and accuracies
ix.	Save the models' logs to a "logs" folder
x.	Use the models' logs to create an interactive visual representation of the models' performances on a local host website using the Tensorboard library