Criterion E: Evaluation

Evaluation

The program satisfies the user's requirements and basic needs for training models for different parameters and evaluating their performances. It portrays very well the graphical representation of different models' learning curves across epochs for both training and validation accuracies and losses. Moreover, it does so both as charts saved on the device as well as an interactive, online representation using TensorBoard, which enables the user to analyze the graphs in detail. In addition, it determines the best model and graphs its learning curves separately to enable the user to analyze and compare them clearly.

Evaluation against success criteria (including user feedback)

i.	All inputs are presented with a user-friendly GUI	✓ - the client agrees that the input and output boxes are user-friendly, but they are not prepared to handle differently formatted user inputs
ii.	An input for the user to define the path to the directory with the dataset and where all output data will be stored	✓ - the changing between directories works well when the user reads the instructions with understanding
iii.	A user input to define the dataset on which the network will be trained	✓/✗ - there is no issue for the user to define their training dataset; however, the dataset itself has to be carefully chosen by the user or otherwise the program is not prepared to handle it – the only format it accepts is where the images are square and in separate folders named by their category. Hence, the user is not always able to use the product, even when it is meant for categorizing images.
iv.	An input for the user to define the target image size after re-sizing	✓/✗ - the input itself allows the user to input only one number, which then becomes the measure of size of both sides of the image after re-sizing, making it a square. That is correct in the case of this program; however, the program could be developed to handle also rectangular images, not only square ones.

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	✓ - the input works well when the user inputs the parameters correctly, according to the given instructions; only the upper limits of the possible integer/tuple inputs were not tested.
vi.	A set of different models is trained and saved in a "models" folder	✓ - when provided the correct inputs and dataset, the models are trained as required for each set of input parameters. However, the user has to clear all the previous models from the chosen working directory, if there were any, or choose a new directory. However, the program is prepared to handle this situation by outputting an error box issuing an appropriate warning to the user.
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	✓ - the plots are graphed without issues and saved in the correct folder. The user is satisfied with the ease of reviewing the different models' performances.
viii.	Find the best model - with the highest final validation accuracy – and create two additional graphs comparing its training and validation losses and accuracies	✓ - the assumed best model is found, according to the user's desired criteria. The user is also satisfied with the separate graphs, providing them with the option to review the best model's performance closely, in more detail.
ix.	Save the models' logs to a "logs" folder	✓ - the logs are saved correctly
x.	Use the models' logs to create an interactive visual representation of the models' performances on a local host website using the Tensorboard library	✓ - the user is happy with the ability to review the learning curves online and interact with them to analyze them in more detail. The user also works on a similar device with Windows so the program works for them, but the TensorBoard commands were not tested for other systems or devices.

Recommendations for Future Improvements

Firstly, the program could be improved by enabling it to handle different types of neural networks, not only CNNs, such as for example Recurrent Neural Networks, which are commonly used for statistical data analysis and prediction.

Secondly, there are certain shortfalls of the program's design and structure due to my inexperience in the subject – there is nearly none error handling, in this case, especially needed for wrongly formatted user inputs, the program was neither designed nor tested for different operational systems than Windows, and the program is asked to simply wait and sleep for over minute near the end.

Thirdly, the program is only able to group images into categories if the dataset is specifically in such a format where the images are square and in separate folders named by their category. This should be improved by changing the program so that it can handle more types of labeling methods.

Lastly, even for the targeted image recognition, the program can only be used to group single objects. If an image consisted of numerous elements that need to be categorized, the network would have to become much more complicated. Hence, there is room for improvement.

Final conclusion

In conclusion, I would consider this project an overall success. All initial success criteria were satisfied, and the user is able to use the program for the desired purpose of analyzing different models' performances. Even though it has a few shortcomings and a lot of room for improvement, both my client and I are satisfied with the result.

Word count: 360