CAI Classification Problem
Report 3: Implementation II

Nil Crespo-Peiró, Paco Rahn, Amin Ranem

26. January 2021

1 Introduction

Once the downsampled dataset has been loaded and the boilerplate code has been set up, the source code needed to be completed to be able to train different models in the future. Further, a first prototype for the Graphical User Interface (GUI) has been designed. In the following sections, the implementation details regarding the changed boilercode will be described and the structure of the user interface will be discussed.

2 Implementation 2

In this section, the final changes to the boilercode will be described. The second part of the report deals with the first prototype of the Graphical User Interface that has been divided into four main windows that will be explained in the corresponding section.

2.1 Final Code Structure

One of the main issues using the whole Cholec80 dataset was the size of each video. Although they have been downsampled to 1 fps, each frame still needed to be resized to 224x224 pixels, in order to be used for transfer learning with PyTorch. With this approach, the Cholec80 dataset that will be used for training has been downsampled to 1 fps and each frame resized to 224x224 pixels. However, the whole transformed dataset is still too big to train with, given the time restriction of this project. Based on this problematic, from each video 2000 slices were randomly selected, if the video has more than 2000 frames, otherwise the whole video will be used for training. This has been realised using the `random` library from Python. The labels have been selected correspondingly. With these adjustments, the code base has been finished successfully and different models can be trained. In the next step, the first GUI prototype will be described, which will be connected to the source code in the next (future) steps.

2.2 User Interface

Since the GUI is an important thing to create that serves as a "bridge" between the source code and the users that will benefit from our work, a first design has been implemented using the `PySimpleGUI` library for Python.
In order to make the life easier for the users of our Surgical Tool Recognition System, the GUI has to be designed in a way that it is firstly, appealing to the user, secondly, easy to understand/use and finally to ensure that the users give the inputs in the desired format in order to be able to make predictions with the trained model. In the next sections, a first prototype of the GUI will be presented, because it represents the foundation of the source code to be used in real case scenarios.

2.2.1 First Window: Browse window

In Figure 1 one can clearly see an introduction message, followed by a brief description on how to proceed: Either by writing a path to the video for which the user wants the surgical tools to be predicted or by browsing through his/her computer.
Once this decision has been made the user can continue by hitting `Submit` or to `Cancel` and choose another video file.

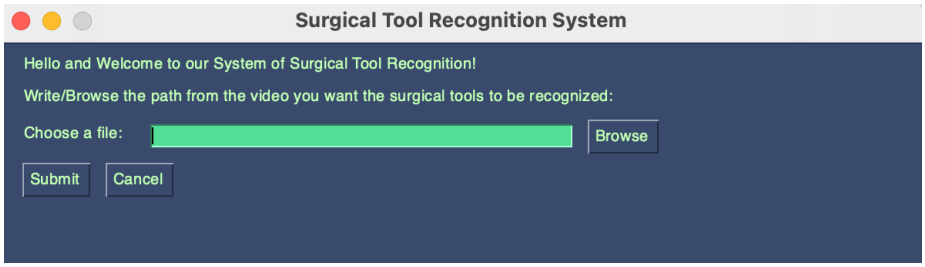

Figure 1: Browse window allowing users to select a video file to be tested

Once the user has pressed `Submit`, a "pop-up" appears in order to make sure this is the intended path to the video that should be further analysed.

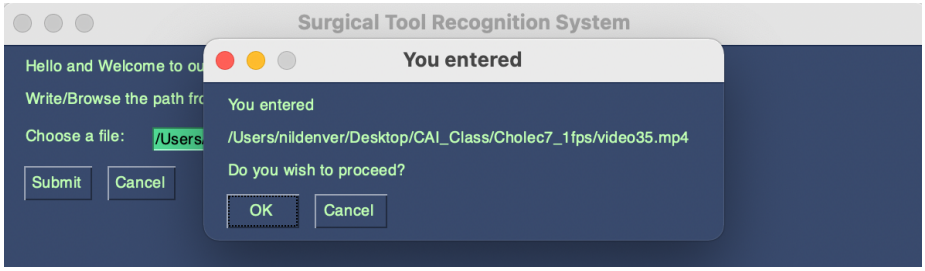

Figure 2: Pop up window that ensures that the choice of the user is correct

As one can see in Figure 2, the popup shows the path that has been chosen along with a message reassuring that the decision was correct.

2.2.2 Second Window - Model Flags, Losses and Specifications

Once the user has pressed `OK` in the last popup window, the following loading window appears:

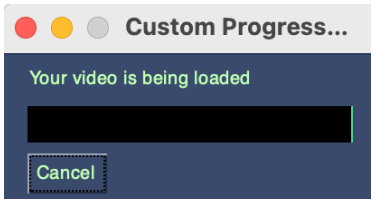

Figure 3: Loading window before choosing the parameters of the model

Once the loading bar has been filled, a Menu where the user can specify all the flags, parameters and options that could be chosen for the training of the model appears. Even though this window is already implemented, it will be redesigned and used to show the user the parameters of the pre-trained model which will be used for the actual prediction.

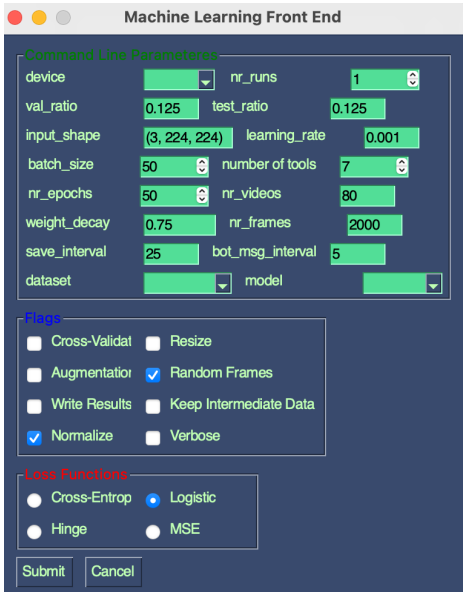

Figure 4: Parameter window where all the flags and configurations can be chosen

However, as we have already mentioned this will be redesigned, since the GUI will only be used for making predictions with a pre-trained model.

2.2.3 Third Window - Output window

In this third window, the user will face a similar window like the first one, but in this case he/she will have to write/browse to a folder where he/she wants the results of the Surgical Tool Recognition System to be saved. The window looks like the following:

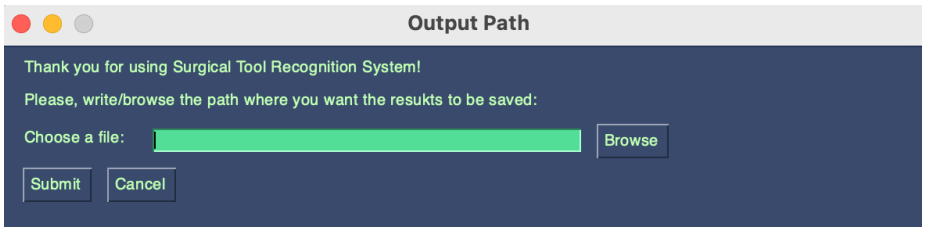

Figure 5: Output window where a folder where the results will be stored has to be decided

As well as in the first window, a "pop-up" will appear to ensure the right location for storing the predictions.

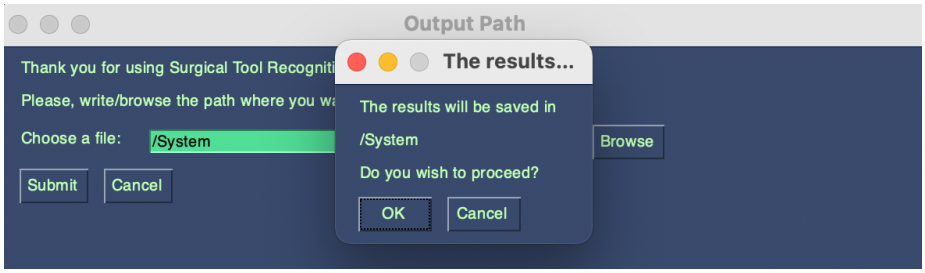

Figure 6: Confirmation window in order to make sure that is the folder where the results have to be stored

Once again, the user will have a choice between the buttons `OK`, in case he/she wants to proceed, or `Cancel`, if the target folder is not the desired one.

2.2.4 Fourth Window - Results window

Right after the user has decided to proceed with that target path, another loading window will appear to indicate the computation of the predictions.

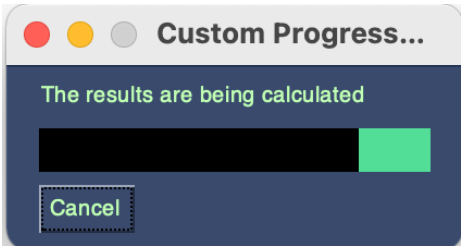

Figure 7: Loading window before seeing the results

Once the computation is finished, the final window will appear, showing the results. This window will show the results of the Surgical Tool Recognition System applied to the specified video.

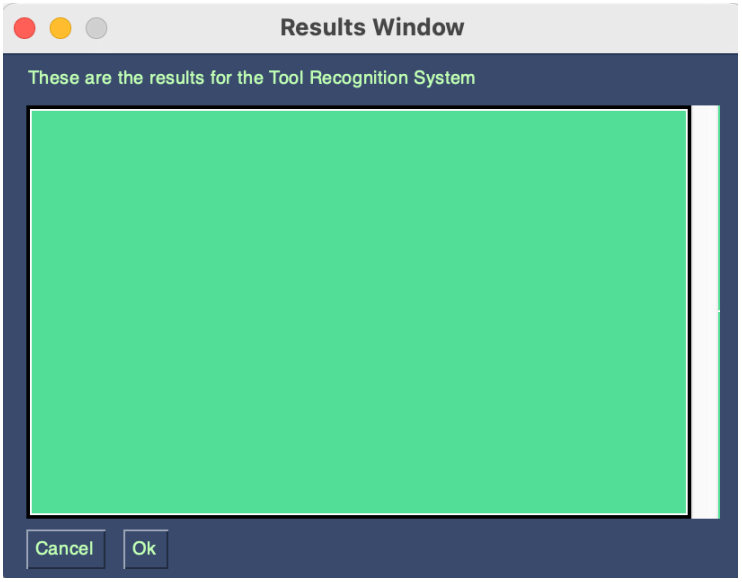

Figure 8: Results window of our system

In the text field, the results will be printed and the user will be able to analyse them aside from having them stored in the path they provided earlier. As we have already mentioned, this GUI is our first design and will be certainly modified/extended to improve it to the user and the source code's needs. The GUI needs still to be connected to the implemented source code.