***README FILE***

*PDE-3433 Advanced Robotics - UNO Cards Recognition*

*Elena Zoretich, M00725319, 3rd year Robotics student (2022/2023).*

***Task Description***

For this second assessment we have been asked to create a computer vision software able to detect and recognize uno cards from both image files and camera stream. For both options, the software must be able to recognize the position of a uno card, identify its color, detect the contours of the numbers/symbols and extract selected silhouette features to estimate the figure of the card. The results must then be display in the card image or camera frame for the user to be visible.

For more accurate estimations, the software implemented a Random Forest Classifier Machine Learning model trained on a set of 280 cards total.

***Libraries & Installing***

The version of Python used to create the software is Python 3.8.6, but latest version should be able to work well too.

The libraries used for this project are the following:

***Numpy***  
The library includes linear algebra operations, exhaustive mathematical functions, random number generator, array manipulations and much more.

To install Numpy library, please follow the guidelines in the following link:  
<https://numpy.org/install/> (pip install, recommended)

***Matplotlib***  
The library includes tools for mathematical representations, customizable visual styles and layouts, interactive plotting with zooming, pan and updates, and much more.

To install the Matplotlib comprehensive library, please follow the guidelines in the following link:  
<https://www.tutorialspoint.com/how-to-install-matplotlib-in-python> (pip install)

***Cv2***  
Cv2 in the module import name for python Opencv. Opencv is an open-source computer vision software library with more than 2500 algorithms for state-of-the-art computer vision.

To install the Opencv library, please follow the guidelines in the following link:  
<https://pypi.org/project/opencv-python/> (pip install)

***Glob***  
Glob is a python module used to find all pathnames matching a specified pattern. In python versions 3.10 and above, the glob module is included in the standard module list, therefore there will be no need to manually install it.

For prior versions, please follow the guidelines in the link below to install the glob module:  
<https://pypi.org/project/glob2/> (pip install)

***PIL***  
Python Imaging Library (PIL) provides file format support, efficient internal representations and solid, powerful image processing tools.

To install the PIL library, please follow the guidelines in the following link:  
<https://pypi.org/project/Pillow/> (pip install)

***Sklearn***  
This library is a useful tool for Machine Learning in Python. It contains efficient tools statistic modelling, including classifications, regression, clustering and dimensionality reduction.

To install the Sklearn library, please follow the guidelines in the following link:  
<https://scikit-learn.org/stable/install.html> (pip install)

***Joblib***  
The library provides a set of tools for lightweigh pipelining in Python, such as transparent disk-caching and simple parallel computing. Optimize to be fast and robust while handling large data and Numpy arrays.

To install the Joblib library, please follow the guidelines in the following link:  
<https://joblib.readthedocs.io/en/latest/installing.html> (pip install)

***Itertools***  
This module implements a great number of iterator blocks, which algebra make possible to efficiently construct specialized tools.

This should be a build-in module, therefore no installation should be required.

***Tkinter***  
This package is a standard Python interface toolkit. It is used to create nice Graphical User Interfaces (GUIs).

Note that Tkinter should be included in all standard Python distributions, nevertheless, to install the Tkinter library, please follow the guidelines in the following link:  
<https://www.tutorialspoint.com/how-to-install-tkinter-in-python> (pip install)

***Keras Processing Image***  
This library is a data augmentation and processing module for the keras library of deep learning. The module procides the utility to ork with text, image, and sequence data.

To install the Keras Processing library, please follow the guidelines in the following link:  
<https://pypi.org/project/Keras-Preprocessing/> (pip install)

***Tensorflow Keras Processing***  
With this library, it is possible to build and eexport models that accept raw images or raw structured data as input. It includes layers for standardization of input images, random augmentation transforms to a batch of images and much more.

To install the Tensorflow Keras library, please follow the guidelines in the following link:  
<https://www.activestate.com/resources/quick-reads/how-to-install-keras-and-tensorflow/> (pip install)

***Repository Content***

The ‘Computer\_Vision\_UNO\_Cards\_Recognition’ repository includes a total of 9 files:

***Dataset folder***  
This folder contains all the 280 cards used to train the Machine Learning model and recognize card color and figure from images.

***Camera Recognition Screenshots (Setup visualization) folder***  
This folder contains 8 screenshots captured from trials of Camera\_Recognition. Those screenshots have been added in order to provide a better understanding of the camera setup and estimate the lighting necessary to obtain best results. Please note that due to poor lighting and low camera resolution, as well as the presence of lighting changes, to obtain adequate results, at time it is necessary to reposition the card around the camera frame.

***Uno\_cards1.jpg file***  
The image is used as decorative display for the GUI.

***Dataset\_Generation\_Augment.py***  
This file has been used to create a bigger dataset from the initially given images. Thanks to the use of ImageDataGenerator(), each numbered card in the UNO deck has been augmented in 6 different, semi-random ways. As the new images for the dataset have been already created (and added in the Dataset folder present in this Github Repository), the code has been uploaded for displaying purposes, therefore it is not recommended to run the code once downloaded.

***ML\_Train\_Save\_Model.py***  
This file has been used to pre-process the 280 cards available in the Dataset folder, extract selected features for machine learning, randomly split the dataset in training (86%) and testing (14%) batches, then save the train model in a joblib file. The file will later be loaded in an apposite python file and estimation will be carried on.

***ML\_samples.joblib***  
This .joblib files stores the trained model previously save in the ML\_Train\_Save\_Model.py. It will be loaded and used in both Image and Camera Recognition files (as well as Full\_GUI code implementing both image and camera methods of recognition).

***Camera\_Recognition.py***  
This file includes the software recognizing uno cards ONLY from camera stream. No GUI has been implemented, as the code has been uploaded for visualization/documentation purpose only.

***Image\_Recognition.py***  
This file includes the software recognizing uno cards ONLY from images located in Dataset folder. No GUI has been implemented, as the code has been uploaded for visualization/documentation purpose only.

***FULL\_GUI\_Image\_Camera\_Recognition.py***  
This is the main code, which users should run to access the Image and Camera recognition. GUI has been added, and it includes button option to select recognition modes (from images, or from camera stream), as well as a menu bar with File 🡪 Exit option, necessary to terminate the execution of the code.

***Execution***

Once the user confirmed that the necessary libraries have been installed, the first step consists in downloading the Github files available in the repository.

The main, final code the should be run is the ‘FULL\_GUI\_Image\_Camera\_Recognition.py’ file. The code has a GUI which enables the user to select either image recognition (from the button label ‘Image Recognition’) or camera stream recognition (from the button labelled ‘Camera Recognition’). Before running the code, some changes must be made:

* Determine the filepath in which it is possible to find the Dataset folder including all 280 cards, the filepath where the ML\_samples.joblib model is stored, and filepath in which the uno\_cards1.jpg image can be found.
* Open the ‘FULL\_GUI\_Image\_Camera\_Recognition.py’ file, go to the following lines and change with the appropriate filepath:
* **Line 145** 🡪 substitute “'C:/Users/elena/Desktop/UNO\_Cards/Dataset/\*.jpg'” with the path in which the Dataset folder can be found on your pc.  
  Please be aware that the slashes should always be inserted as ‘/’, and that no specific card name should be added, but the extension ‘/\*.jpg’ should be attached at the end of the folder path.
* **Line 163 & Line 244** 🡪 substitute “'C:\\Users\\elena\\Desktop\\University Stuff\\3rd Year\\PDE-3433 Advanced Robotics\\Uno\_Cards\\ML\_samples.joblib'” with the path in which the ‘ML\_samples.joblib’ file can be found on your pc.
* **Line 277** 🡪 substitute “"C:/Users/elena/Desktop/uno\_cards1.jpg"” with the path in which the ‘uno\_cards1.jpg’ file can be found on your pc.

Now the code should be able to run without any issues.

Once the GUI appears, if the ‘Image Processing’ button is clicked, the program will automatically select a random set of cards in which the ML trained model will display its predictions. Once all test cards have been displayed, the program will automatically send you back to the GUI main menu, and you will be able to reselect the same recognition modality, tested on a set of different cards, or select recognition from camera stream. If you do not wish to visualize all cards during image recognition, instead of pressing the ‘Esc’ button to iterate through images, please click the window ‘X’ button to interrupt the program and go back to the GUI menu.

If you wish to test the Camera recognition software, please select the ‘Camera Recognition’ button, and wait for the camera frame window to appear (this might take few seconds).  
As the software has been tested with the use of an External camera, please try to follow the subsequent instruction, to create a setup as similar as possible to the one used by the author.

* Place the external camera in a downward facing position, approximately 18-19 cm away from where the card will be placed.
* The camera should be facing a black surface, to minimize brightness noise and to optimize the contours detection.
* The lighting in the room you are should not be excessive. As the software has been mainly tested in evening times, it is recommended to run the program in a dark room, illuminated by a desk lamp positioned roughly 1m away.
* The ‘FULL\_GUI\_Image\_Camera\_Recognition.py’ file includes, in line 196, a Opencv function which select the appropriate device for video capture. Generically, if the code is run on a laptop with camera, ‘cv2.VideoCapture(0)’ will open the aforementioned. Although in my experience, VideoCapture(-1) correspond to the laptop camera, while VideoCapture(0) correspond to my external webcam.  
  Test with a heuristic technique which VideoCapture value correspond to the appropriate camera device.

Once you wish to terminate this recognition method, please press the ‘Esc’ button, and you will be redirected to the GUI menu.

If you with to terminate the software, place your cursor over the ‘File’ option available on the GUI menu bar and click on ‘Exit’. This will terminate the program.

***Other Files Execution***

As ‘FULL\_GUI\_Image\_Camera\_Recognition.py’ includes all components necessary to carry on the assessment’s tasks, it is not necessary to fun the other files available in the repository. Although they have been added for visualization/documentation only, if you wish to run them anyway, some changes must be made. Similarly, to the main code, the filepaths must be changed accordingly:

* **Camera\_Recognition.py:**  
  **Line 26**, substitute path with location where saved ML.samples.joblib model is stored.  
  (Also, in **Line 28**, change VideoCapture() argument with number corresponding to your external camera)
* **Dataset\_Generation\_Augment.py:**  
  **Line 30**, substitute with path in which initial set of cards you wish to augment are found.  
  **Line 34 & 44**, substitute with path in which you wish to save the new set of augmented cards.
* **Image\_Recognition.py:**  
  **Line 125**, substitute with path in which your dataset card images are).  
  **Line 126**, substitute with path in which ML\_samples.joblib model is stored.
* **ML\_Train\_Save\_Model.py:**  
  **Line 82**, substitute with path in which dataset of card images can be found. The trained model .joblib file can be find in the same folder in which the code is stored.

***Common Issues and Help***

If the README file has been properly followed, no issues should be encountered. Nevertheless, a short list of possible issues has been drafted below:

* **SyntaxError: (unicode error) ‘unicodeescape. Codec can’t decode bytes in position 2-3: truncated \UXXXXXXXX escape**:  
  This error might appear if the wrong type of slashes has been added to a filepath. By replacing the backward slashed with forward slashes, the error should disappear.
* **No errors, but the code terminate without executing**:  
  If there are any syntax error related to folder names, the software will not be able to execute properly. Make sure folder and file names are appropriate before running the code again.
* **FileNotFoundError: [Errno 2] No such file or directory:**This error might occur if you are trying to load the Machine learning trained model file (ML\_samples.joblib) from the wrong folder. Make sure the path corresponds to the place in which the files has been stored.
* **ModuleNotFoundError: No module named '\_\_\_' :**This error will occur if any of the used libraries is not installed. Please read the ‘Libraries & Installing’paragraph at the beginning of this README file and make sure all necessary libraries are installed.

If any other issue is encountered while trying to execute any of the files, please do not hesitate and contact the author. Contacts will be left in the paragraph below.

***Authors***

Elena Zoretich, Middlesex University, Student ID M00725319.

Email = [ez111@live.mdx.ac.uk](mailto:ez111@live.mdx.ac.uk)

Github IS & URL: EZoretich, <https://github.com/EZoretich>

***Acknowledgements***

A large variety of resources, including videos, books, and websites, has been used for the development of the program. Some of the main references are listed below.

* Documentation for better understanding of Image Processing and Opencv  
  <https://www.analyticsvidhya.com/blog/2021/05/image-processing-using-opencv-with-practical-examples/#:~:text=OpenCV%20%E2%80%93%20Open%20Source%20Computer%20Vision.&text=It%20is%20used%20in%20various,social%20distancing%2C%20and%20many%20more>
* ImageDataGenerator() for card augmentation - creation of bigger dataset  
  <https://medium.com/swlh/image-classification-for-playing-cards-26d660f3149e>
* Joblib documentation  
  <https://joblib.readthedocs.io/en/latest/>
* Saving and loading Machine Learning trained models  
  <https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/>
* Understanding color recognition for camera stream and cv2.getTrackbar()  
  <https://www.youtube.com/watch?v=t71sQ6WY7L4&t=1172s>
* Machine Learning – train\_test\_split  
  <https://builtin.com/data-science/train-test-split>