Computer Vision Project 1
Automatic Grading of Multiple Choice Tests

Matei Bejan, group 407

April 2020

The application makes use of 9 main functions, 3 neural networks and some additional data. The functions are:

- **crop_upper_checkbox** The function crops a region around the upper checkbox and removes all the white edges until it find a black edge (the checkbox itself) and then all the black edges until it finds a white edge (the inside of the checkbox) from all edges. The resulting image will be the inside of the checkbox.
- **crop_lower_checkbox** Same as **crop_upper_checkbox**, but applied to the lower checkbox.
- **crop_left_table** The function crops the left table out of the picture, making sure to leave enough space in all direction so that the main lines of the table - those that contain the ticks - are visible.
- **crop_right_table** Same as **crop_left_table**, but for the right table.
- **get_vertical_lines** The function detects the vertical lines using the Hough line transform and then filters the lines by not allowing diagonal-type lines and by thresholding the distance between the detected lines. The resulted lines are sorted with respect to their x coordinate and the first 5 lines are returned.
- **get_horizontal_lines** Same as **get_vertical_lines**, but applied to the horizontal lines. The only differences would be that the sorting is done with respect to the y coordinates and that it returns the first 16 lines.
- **find_x_from_images** This function is adapted from what we did in the lab. The main differences would be that:
  a. The thresholds for the answer box crop are no longer hardcoded, but make use of a percentage of the distance between the x and y coordinates for the horizontal and vertical lines, respectively.
  b. The patches are parsed twice. The first time for calculating a patch threshold and the second time for the actual detection, using the threshold in order to differentiate between a ticked box and a blank one.

- **align_images** This function detects ORB features between a template and a rotated picture, matches them, sorts them, removes the bad matches and retains only the top features in order to find and use homoegraphy so it can warp the pictures. Both the number or ORB features and the top % that are used for warping come as function parameters.
- **get_option** This function calls the **crop_upper_checkbox** and the **crop_upper_checkbox** functions and, depending on which checkbox is darker (contains a digit) it applies a CNN for digit recognition on it and returns the predicted number, alongside with the subject (”Fizică” or ”Informatică”), as a tuple.

The networks are:

- **14_recognizer.h5** A CNN trained on the following data: those MNIST images that contain 1, 2, 3 or 4 enhanced with the cropped checkboxes from the scanned train images.
- **digit1recognizer.h5** A CNN trained on MNIST enhanced with hard-coded crops of the first digit in the handwritten grade.
- **digit2recognizer.h5** A CNN trained on MNIST enhanced with hard-coded crops of the second digit in the handwritten grade.

The architecture used to train all the CNNs can be found in this Google Colab notebook.

The additional data consist of 3 template images: 1_reference_scanned.png, 2_reference_rotated.png and 3_reference_perspective.png. Those are three cherry-picked images from the train data with respect to how well the warp features can be seen. The three are manually cleaned - all the non-printed data is removed- and are used when calling the **align_images** function as templates for warping.

Tasks 1-3 are solved as follows:

1. Align image to template, when the need arises.
2. Extract tables and answers.
3. Extract exam choice either via the **get_option** and applying **14_recognizer.h5** on it to infer the option (for data with no annotations) or by parsing the file name and extracting the data from the respective solution .txt file.
4. Calculate the grade based on the extracted answers and option.

The solution for task 4 relies on cropping the first and second digits and applying the adequate CNN in order to infer the number. Then, if the prediction is not part of all possible scores, the grade gets reassigned the most frequent score that has the same first digit from our test data as our predicted score. We chose this approach as the first digit predictor is much more accurate than the second one (90% vs. 50% accuracy, respectively). Also, if the score is less than or equal to 1, we use the most frequent grade we found in the test set as our prediction.

Note: The project archive comes with an *output* folder, where the results are stored.