UDACITY

PROJECT

Facial Keypoint Detection and Real-time Filtering

A part of the Artificial Intelligence Program

PROJECT REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Great work on this project. You have met all the requirements of the project, and your implementation produces very good results.
Good luck with the nanodegree!

Files Submitted

`CV_project.ipynb` --> all required python functions are completed in the main notebook. `CV_project.ipynb` TODO items should all be completed.

Step 1: Add eye detections to the face detection setup

The submission returns proper code detecting and marking eyes in the given test image.

Good, your implementation correctly identifies and marks the eyes

Step 2: De-noise an image for better face detection

The submission completes de-noising of the given noisy test image with perfect face detections then performed on the cleaned image.

Great, you have correctly used cv2.fastNlMeansDenoisingColored to denoise the photo, and now all the faces are being detected

Step 3: Blur and edge detect an image

The submission returns an edge-detected image that has first been blurred, then edge-detected, using the specified parameters.

You have successfully blurred the image and performed the edge detection to identify the main edges in the image

Step 4: Automatically hide the identity of a person

The submission should provide code to automatically detect the face of a person in a test image, then blur their face to mask their identity.

Very good job. I just wanted to point out that you could have used cv2.blur function to do the blurring too.

Step 5: Specify the network architecture

The submission successfully provides code to build an appropriate convolutional network.

Good architecture. I liked how effectively you used Convolution2D and MaxPooling2D. Consider adding Dropout to it.

Step 6: Compile and train the model

The submission successfully compiles and trains the CNN.

Please note that you can show accuracy stats by adding `metrics=['accuracy']` to `model.compile(` .

Step 7: Answer a few questions and visualize the loss

The submission successfully discusses any potential issues with their training, and answers all of the provided questions.

Great, I enjoyed reading your answers. I liked how you described different approaches you took.

Step 8: Complete a facial keypoints detector

The submission successfully combines OpenCV's pre-processing techniques and face detection with a trained CNN keypoint detector.

Very good. Your implementation correctly identifies face keypoints.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ