Programming challenge for Machine Learning / Computer Vision Engineer role

1) The three parameters that I could change/alter using the functions in the imaging _interview.py were as follows:

    - The radius of the gaussian blur:
      I chose the kernel size of gaussian blur as two values "**3 and 5**"

    - The number of gaussian kernels:
      I chose only **2** gaussian kernels for the removal of noise from images

    - The minimum contour area:
      I chose the minimum contour area of **450**.

2) I was able to narrow down my problem by thoroughly looking through the dataset and I realized that there were three major types of scenic changes with the dataset.

    - The first scenic change was somewhat related to changes in weather/atmospheric conditions, such as a sunrise, sunset, rain or change of shadows because of different times of the day the images were captured. I realized that such changes wouldn't be containing any crucial information and to be able to get rid of such changes, the dataset could be reduced at large.

    - The second change I observed in the dataset was related to the presence or absence of cars. I realized that this change was definitely crucial and I wanted to cater such changes into my algorithm

    - The third change was related to presence or absence of any person in the image and this was crucial again, therefore it had to be catered as well.

    I resorted to visually debug through the dataset, and I took multiple pairs of images which had the changes as I have discussed above. I used different sizes of gaussian filters and upon visualization I came to the conclusion that the filter size of "**3 and 5**" did the job right as to remove the noise artifacts from the dataset. By using these two filters I was able to get rid of the noise between the two pair of images which was mostly related some minor changes related to climatic conditions

    I decided to stick to only **two** filters because upon introducing a third filter I was losing information from the images that could be of crucial use and be of little benefit to compare pair of images.

    I was able to decide for a minimum contour area of **450** by comparing the image pairs where there was presence of any human or similar and my decision was based on observing the contour areas between such images. The presence of any human was of crucial importance therefore I came to the conclusion that I could set a minimum contour area which coincides the area of a human in the image. Anything below such area would merely be because of noise or climatic reasons. Moreover, the presence of any human between two image pairs could only cause a small change considering the wide field of view that was being captured therefore **450** was the optimal number that I decided to work with.

    I also introduced some other parameters related to **probability** and **no. of contours** on which my algorithm heavily relies on. I looked through the dataset and choose a set of images that contained the changes I have discussed above. I was able to formulate a table of values where I compared the changes in probability and

in no. of contours, in the image pairs and get to some optimal values that would cater the changes. I chose values that were in alignment with most of the observations that I had in my table.

3) The script that I use discards away around 376 out of 484 images, which is roughly around 77 % of the dataset.

4) I would suggest that there can be an introduction of some deep learning approach, where instead of resorting to basic image processing techniques, a network could be designed or some state-of-the-art network could be used. Such a network can take in two image pairs or a single image, and extract the important key features from the image. Alongside, the dataset could be annotated or labelled, which I think should not be a cumbersome task since it's a matter of binary labelling between two pair of images as "True", if they are same or "False" if not. The network could be trained over the dataset and eventually we let the network decide that based on extracted features, the image pairs are same or not. If such an approach is used, I think better unique cases would be collected.

5) The first thing that I think I would suggest that we can change the color space that was used to compare the images. Since the dataset has a lot of lightening artifacts, therefore if we used HSV, instead of Grayscale, I think it would be rather easier to deal with lightening artifacts. Moving on, I was not sure if I could introduce some amendments to the default functions that were provided therefore, I felt like I had to formulate a logic based on what I get returned from those functions. Elsewise if I was given full control of the task, I might have resorted to some other optimal solution as well.

   Lastly, I have introduced **probability** and **no. of contours** variables on the basis of which my logic holds, and I had to make a few observations from the dataset and decide the optimal values. The script, I think is able to discard away a lot of images do not differ much and also alongside some other images that do have a lot of difference between them. I could also get to these optimal values based on some machine learning algorithm rather than visual debugging and I think those optimal values could introduce even better optimization to the dataset. I was not sure if I was also expected to maybe go in that direction but if so, then maybe I needed a little more time to think over the problem and generate some data from the compared images, such as areas and size of contours and then use some machine learning algorithm to decide the optimal values.