**Dog Prediction Report**

|  |  |
| --- | --- |
| **Question** | **Answer** |
| Question 1: Use the code cell below to test the performance of the face\_detector function.   * What percentage of the first 100 images in human\_files have a detected human face? * What percentage of the first 100 images in dog\_files have a detected human face?   Ideally, we would like 100% of human images with a detected face and 0% of dog images with a detected face. You will see that our algorithm falls short of this goal, but still gives acceptable performance. We extract the file paths for the first 100 images from each of the datasets and store them in the numpy arrays human\_files\_short and dog\_files\_short. | Percentage of images with human face in human\_files: 100.0% Percentage of images with human face in (dog)train\_files: 11.0% |
| **Question 2:** This algorithmic choice necessitates that we communicate to the user that we accept human images only when they provide a clear view of a face (otherwise, we risk having unneccessarily frustrated users!). In your opinion, is this a reasonable expectation to pose on the user? If not, can you think of a way to detect humans in images that does not necessitate an image with a clearly presented face? | It's possible to train an algorithm on a set of pictures that have humans not facing the camera... though this would take a lot more effort, potentially. If the user does not provide a human face, it's possible that the program could mention that they see a human but no face. This isn't too problematic as the use case relates to seeing which dog breed the human resembles, and without a face that would be very troublesome indeed. Asking the user to have a clear view of the face is probably a good design choice especially to avoid training another CNN.  We suggest the face detector from OpenCV as a potential way to detect human images in your algorithm, but you are free to explore other approaches, especially approaches that make use of deep learning :). Please use the code cell below to design and test your own face detection algorithm. If you decide to pursue this *optional* task, report performance on each of the datasets. |
| **Question 3:** Use the code cell below to test the performance of your dog\_detector function.   * What percentage of the images in human\_files\_short have a detected dog? * What percentage of the images in dog\_files\_short have a detected dog? | 0%  100% |
| **Question 4:** Outline the steps you took to get to your final CNN architecture and your reasoning at each step. If you chose to use the hinted architecture above, describe why you think that CNN architecture should work well for the image classification task. | Overview of CNN architecture [ 6 x Conv2D\_maxpooling ] -> [ GAP ] -> [ FC ] -> [ Dropout ] -> [ Output - FC ] Conv2D\_maxpooling = [ conv layer -> pooling layer ] pattern architecutre - one convolutional layer followed by a pooling layer, repeated 3 times receptive field size(kernel size) 5x5 since we have a larger image stride width of 1 and padding of zeros Use small kernel to gather a lot of information The beginning layers will be used to extract low level features Several layers will help identify complex structures on the dog Increase depth layer after layer to match increased complexity of model Use pooling to decrease dimensionality of feature map Connect to FC layers |
| **Question 5:** Outline the steps you took to get to your final CNN architecture and your reasoning at each step. Describe why you think the architecture is suitable for the current problem. | * Classify dog breed problem requires more filters to discover more pattern in the image. CNN should be deep enough to discover these patterns.To get deep, we should add more layers, but at some point the performance declines due to vanishing gradient problem. Resnet architecture/InceptionV3 is the best to overcome this problem unlike VGG. * Thinks to learning transfer, using the weights in pre-tranined Resnet, images are passed through the Resnet network to the last max pooling layer to get a new data set. * Finally, creating our network by two layers only. The input layer that uses the images passed through the Resnet and an output layer with fully connected layer with softmax to get object in the image. * Compare the performance between Resnet50 and InceptionV3 |