Project 2020

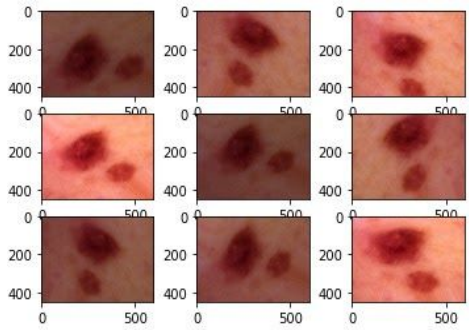
SU_23575_23637_23661_23873

CS412 - Machine Learning

Report

Problem Approaches

The initial steps taken were understanding the problem we have. In order to fully understand the issue here, the given data was investigated comprehensively. Since we only had 10000 image data to handle and predict results, we need to do data augmentation. So that, we aimed to have higher accuracy scores. After exploring our image data, it has been figured out that the dimensions of the images are not equal. Turning images into grayscale was also one of the methods we considered but in the end we used colored images. Another approach we tried to do was resizing images to the same scale. We uploaded every image to google drive with the same sizes (320x320). After a couple of days, we figured out that we would not need these operations beforehand. In the final version of the project, we resize them before giving as input to the model. We also used several methods to upload images to our kernels. Our project has different versions where we tried our best to increase efficiency each time. The details of methodologies are in the next section.

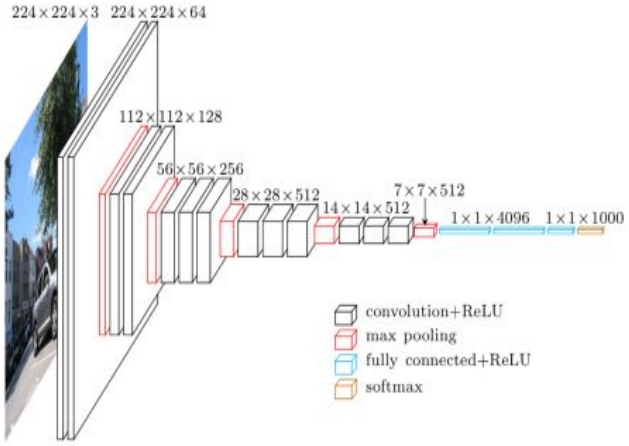


Methodologies and Model

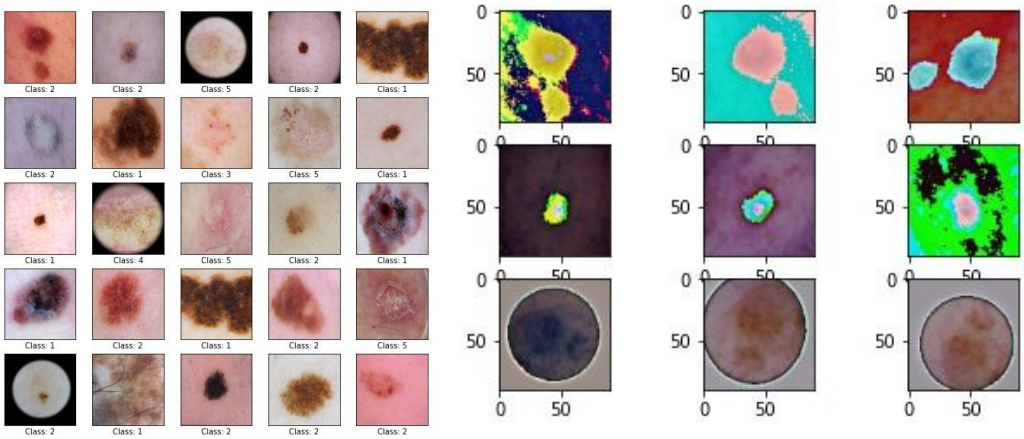
As mentioned above, the first step we took was uploading resized images to google drive. By the way, resizing the all train images took a lot which was inefficient. Another inefficient method was using glob function to reach all images we have uploaded to drive. It also took a lot of time. Once we constructed an image array which is aimed to be used in the neural network model, the rest was constructing label arrays and fitting them into our model. After determining image and label arrays, we splitted trains dataset into training and validation with 0.2 ratio of test size. After we tried several different CNN models with those arrays, we never reached more than 0.45 validation accuracy. In spite of reaching 0.91 training accuracy, our validation accuracy had a high variance. The model had an overfitting. At this point we tried to increase the upload speed for image and label arrays. We constructed two data frames consisting of paths to images and csv files. ImageDataGenerator has increased our upload time which was a very helpful update for us. So that, we trained our model more frequently. Although we changed our generator and CNN model, the accuracy scores were very low considering the model we used. We tried different batch sizes, convolutional layers, activation functions and input sizes. The scores were still lower than our expectations. It was the crucial point in the project. We decided to move on with transfer learning.

Transfer Learning

In order to classify images using Convolutional Neural Networks with Python and the Keras deep learning library, we tried five different pre-trained networks which are ResNet50, Inception v3, Xception, VGG16 and VGG19. In our case no model other than VGG16 worked well hence we decided to build our model on VGG16.



On the left you can see the visualization of the VGG architecture. First, our model read the given datasets (Data_SkinCancer, Train.csv, Test.csv). Then, the model preprocessed and resized the input images into 90x90x3. Up to this point we have prepared our initials. Then we moved to the part where we created our training and validation datasets. To cover this part we resize all the images whereas preprocessed the images by expanding the dimensions and subtract the mean RGB pixel intensity from our dataset. Then we placed these datasets to create a model with our fully connected network..



The first picture matrix shows the training inputs of our model. On the other hand, in the second matrix it can be observed that the model can focus on the appropriate part of the picture with our pre-trainings.

Results

In the final analysis, we deduced that even though we have a higher accuracy score like 0.81, the validation accuracy is not high enough if we do not use transfer learning. Here is the result of the model without transfer learning:

accuracy: 0.8112 - val_loss: 1.7270 - val_accuracy: 0.525

On the other hand, with transfer learning our validation accuracy is considerably high. But the accuracy score is less than the previous model. Here is the result of the pretrained model:

accuracy: 0.6796 - val_loss: 1.0143 - val_accuracy: 0.6213