

Soheil Shirvani

Retail Product Classification and detection

Snap Company Test Case Project



In this project we are going to first develop a model to classify the retail products which are given in 18 classes. We are assumed to be given an image of a shelf consisting multiple retail products. We then want to detect each of the objects in the image and apply a classification model to determine its class.

We first are going to determine some techniques used in this project, then we will demonstrate some models to be fitted on the dataset. We separate our test and train dataset using a list given by the provider. 0.2 data are used as validation and 0.8 others are used as the training dataset.

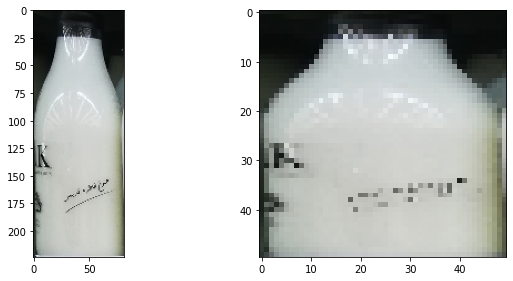
Transfer learning is used to develop a model to fit on the dataset. For preprocessing, the in-built preprocess of the model is used. Multiple models have been tried and the best result is going to be introduced.

Results are plotted and analyzed at the end of this report.

There are 3445 product images in size of 224\*224\*3, a sample of this dataset is shown below:



The black part in these images may be a problem for our model as model can overfit to the black parts. Thus, it is possible to crop the black part and resize the image again. A sample to this technique is done below:

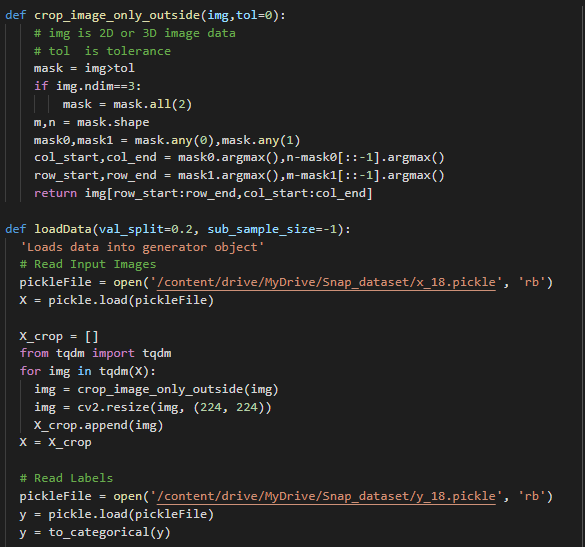


Since there are not much data in our dataset, we use augmentation technique to create a better dataset. This technique applies some deformations on the origin images and create an artificial dataset. The result dataset helps the model to face less overfitting. A sample of this technique is shown below:

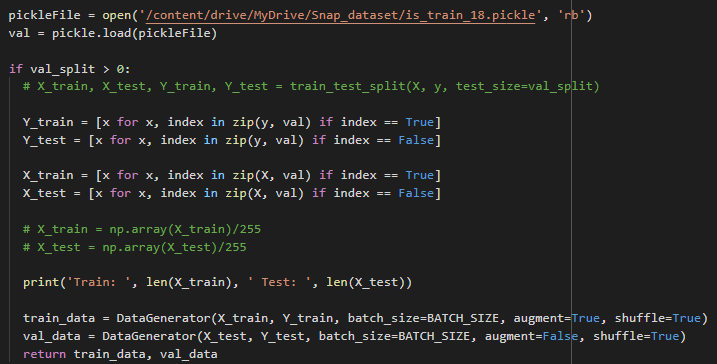


Result of applying augmentation can be any of the above images.

The data is first loaded using the pickle files provided by the company, some samples were given in the techniques section. Then, the back parts of each image are cropped and images are resized back to the origin size 224\*224\*3. The labels are loaded using the pickle file and converted to One Hot Format using keras categorical function.

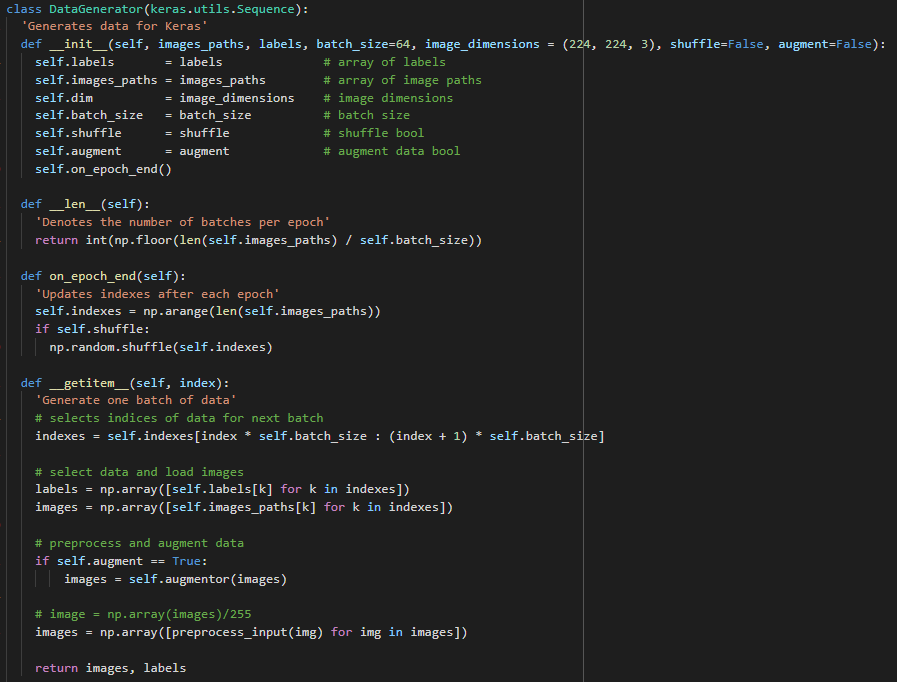


A validation pickle file is loaded. This file contains indexes of the validation set from the origin input dataset. This file is provided by the company and in this project the same number of validation set is used.



2402 data are in train set and 1043 data are in validation set. The data the converted to a Data Generator introduced in the following section.

Dataset Generator which is inherited form keras dataset generator was customized for this project as below:



This data generator loads each batch of image, find its indexes (random indexes if shuffle is True) and then for each image in the batch, image and its relative label is loaded. For each image in the batch (if augmentation is True) an augmentation is applied and each image preprocessed using VGG16 in-built preprocess. The images and their relative labels are then loaded to the model and update the weights.

This generator is used to create batches of training and validation data after loading them. Batch size and image dimension are two important factors in dataset generator.

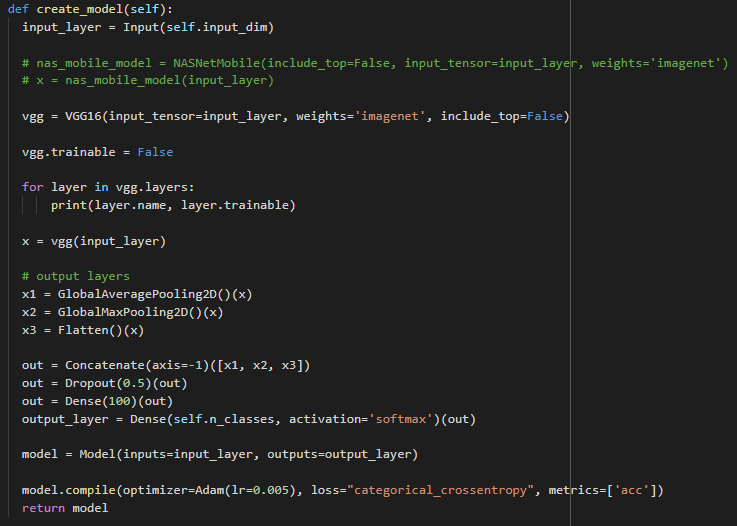
Augmentation used in this project is shown below:



This augmentation seems a lot. This is the reason why did not get a good training accuracy but our validation accuracy was good.

This technique helps the model to touch a better accuracy. A model without augmentation was tested and performed poor.

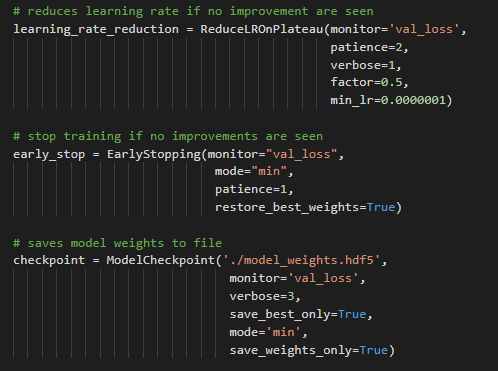
The base model used in this project is VGG16. Other models like InceptionV3, NasNetMobile, and VGG19 were tested but the best result was achieved by VGG16. Adding a GlobalMaxPooling, GlobalAveragePooling, and Flattening to the last Conv layer in VGG16 and Concatenating them, then Adding a 100-layer dense and 18-layer dense (18 is the number of classes) our final model was created.



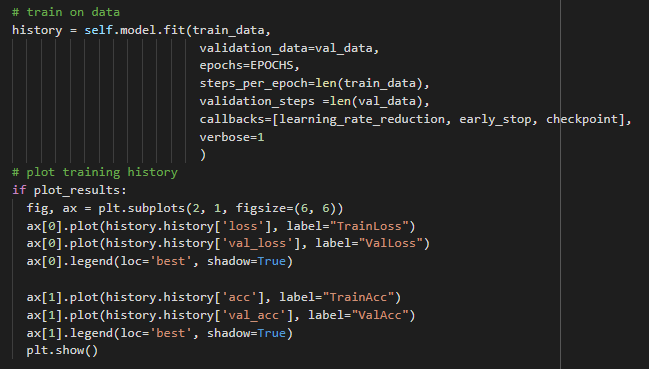
The model is then complied using Adam optimizer with initial learning rate of 0.005 and “categorical cross entropy” loss function.



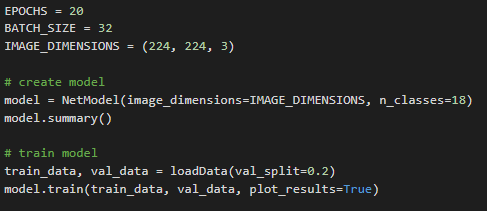
Three Callbacks including Learning rate reduction, Early Stopping, and Model Check Points were created and the model is fitted using these callbacks.



The model trained using the above techniques and after completion, the results are plotted.



We run our model with below conditions:

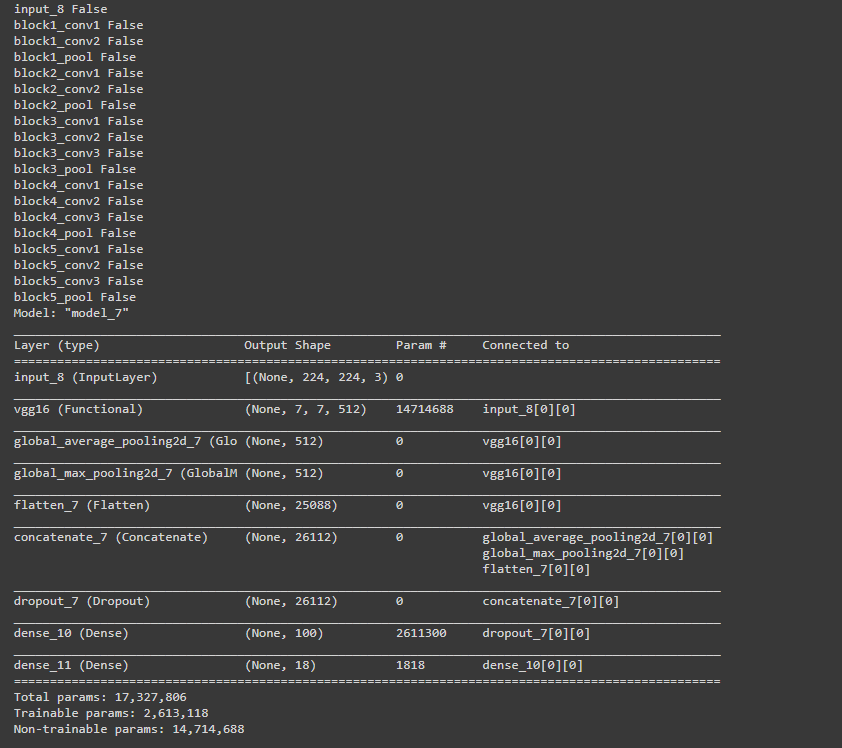


Here our batch size is 32, our input size is the origin size of the images. We fit our model for 20 epochs but because of early stopping callbacks our model converged earlier.

Turning of VGG16 layers trainable (meaning we did not train VGG model weights) we managed to reach the best accuracy. Augmentation and choosing VGG16 as our base model, also were the techniques made our accuracy higher.

After completion of training our model plot the accuracy and losses of each epoch. Training was done in 4 epochs before model stops and results were gathered. Model weights were saved and they are ready for further usage.

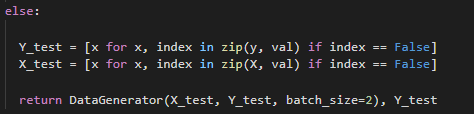
Best Results are shown below:

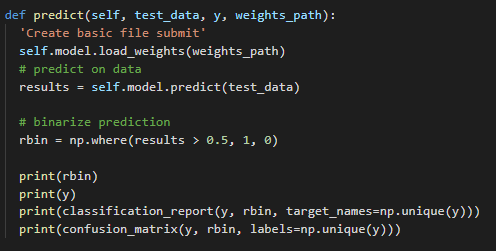




As it is shown above, our best accuracy on validation data reached 94% and for training data it was 85%. Our accuracy on training data was a little lower because of over use of augmentation. Due to deadline of this project, I could not tune the augmentation, so results may not be the best.

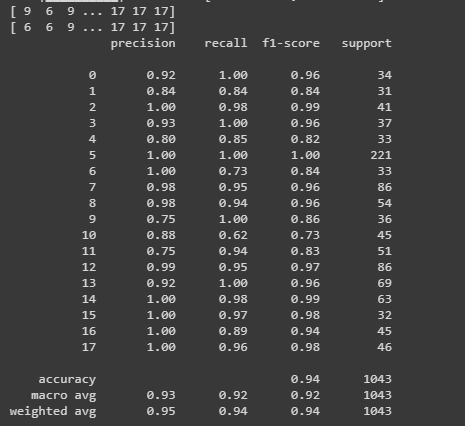
We can now evaluate model by just giving Validation Generator to the model and predict the results:



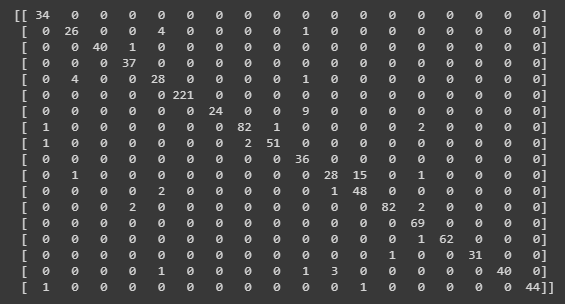




By predicting the validation data generator as shown above, results are like below:



And Confusion matrix is like below:



As it can be seen only small amount of data is misclassified. Most of them were in class 1.

Code, Report, and best weights are attached to the same file.

Good Luck,

Soheil Shirvani