CLASSIFICATION OF HOME OBJECT IMAGES

MACHINE LEARNING HOMEWORK 2 REPORT

Gioele Migno - 1795826

12/22/2020

ABSTRACT

The project consists of the development of an image classifier that is able to classify eight different classes of home objects. In order to perform this task have been compared three different convolutional neural networks by using transfer learning method in some of them.

PREPROCESSING

Analysis of the dataset

The dataset contains images of eight different categories of home objects, the typologies considered are the following:

- *cereal_bowl*
- *chocolate_drink_bottle*
- *dinnerware*
- *Glass_Cleaners*
- *Mixed:* (Mix of fruits)
- *Party_Mix_snack*
- *pasta_sides*
- *plastic_container*

The images used are derived from several sources, then their resolutions and ratios are different. In order to train a neural network it is required to choose a fixed dimension as the input layer, so all pictures must be adapted to this dimension, to define it, at first the characteristics of the images have been analyzed.

Overall the dataset is composed by 7945 images, the most common resolutions are:

- 300x300 pixels (1045 samples)
- 225x225 pixels (348 samples)
- 191x191 pixels (316 samples)
- 300x225 pixels (187 samples)

Moreover, considering the ratio (width/height) with one decimal digit, the most common ratio is 1.0 (3730 samples), in Figure 1 is shown the distribution of the ratios.

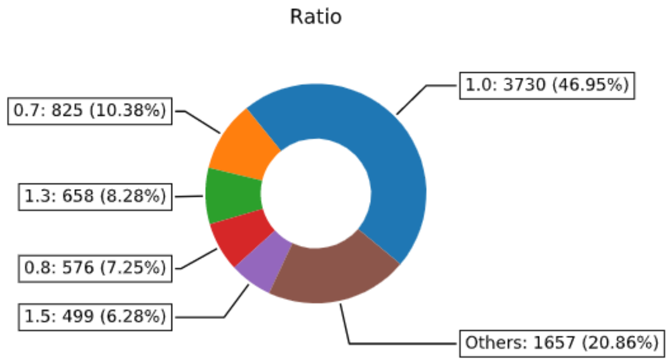
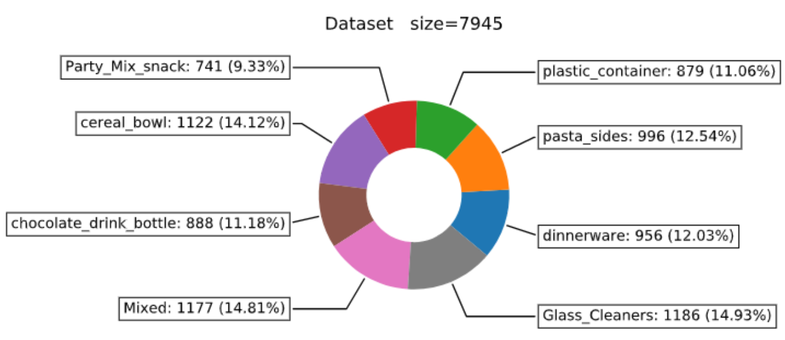


Figure 1: Distribution ratios of images of the overall dataset

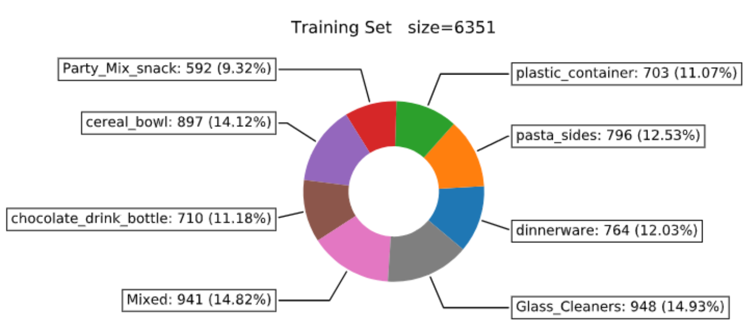
As a result of these considerations, the dimension of the input layer has been set to 256x256, so using an unitary ratio.

Split Dataset

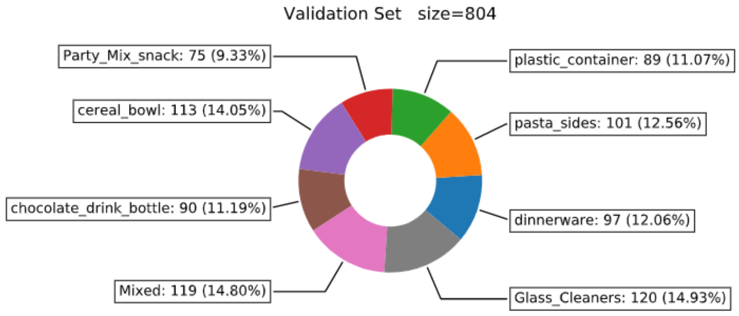
The dataset has been splitted into training (80%), validation (10%) and test (10%) set, the operation has been performed preserving the distribution of the classes in the set. In Figure 2 are shown the number of samples, the distribution in the overall dataset and in each subset.



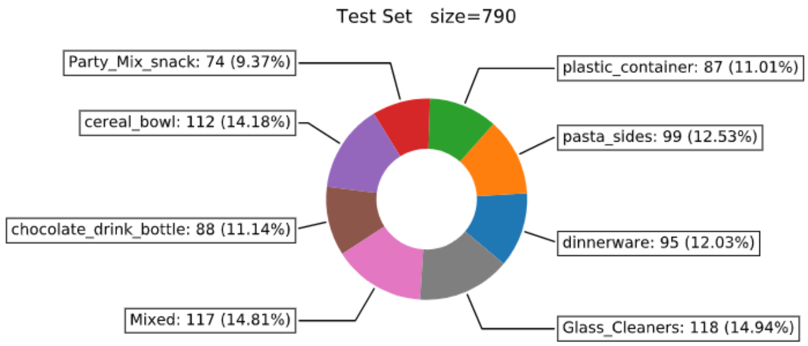
(a)



(b)



(c)



(d)

Figure 2: Distribution of classes in the overall dataset and in each subset

During the training, data augmentation technique has been used, it has been performed through `ImageDataGenerator` Keras class by using the following parameters:

```
rescale = 1./255
zoom_range=0.1
rotation_range=20
width_shift_range=0.1
height_shift_range=0.1
horizontal_flip=True
vertical_flip=False
```

EXPERIMENTS

The project has been performed testing three different convolutional neural network models, the first is ResNet50, then AlexNet and VGG16.

During the training of all networks, the environment used is Google Colab with GPU, the batch size is 64, the optimizer is Adam with default initial learning rate `lr=0.001` and the loss function is `categorical_crossentropy`. In order to reduce the randomness during the several trainings, both random seed of tensorflow and numpy have been set.

ResNet50

The ResNet50 network has been implemented by using the model defined in Keras Applications and importing only the convolutional layers hence, without the final fully connected layers (include_top=False). After that layers, the following ones have been added:

```
Flatten
BatchNormalization
Dense (Nodes=4096, Dropout=0.2, Activation_Func=ReLu)
BatchNormalization
Dense (Nodes=4096, Dropout=0.2, Activation_Func=ReLu)
BatchNormalization

BatchNormalization
Dense (Nodes=8, Activation_Func=Softmax)
```

During the training, only the added layers have been trained, the other ones have been initiated with the weights pre-trained on ImageNet dataset and set as not trainable.

The initial learning rate of Adam optimizer has been left at the default value (lr=0.001), the model has been trained for 100 epochs that have required 14954 seconds (about 4 hours). The highest validation accuracy is 68.16% reached at epoch 91 with val_loss=12.4924

In Figure 3 the plot of the accuracy during the training is shown.

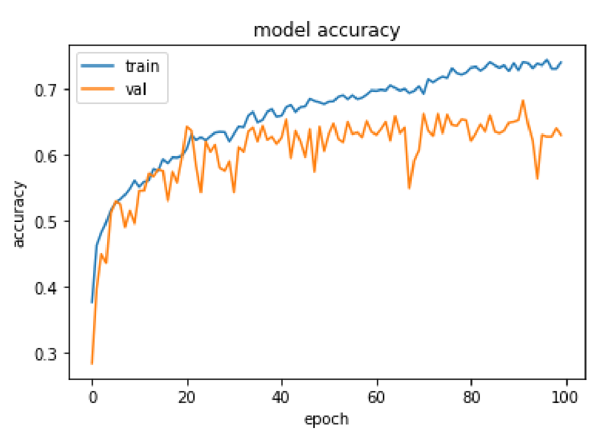


Figure 3: Accuracy in training and validation set during the training

AlexNet

AlexNet has been trained from scratch, in this case the model has been trained using two different initial learning rates, at first lr=0.0001, then the default one (lr=0.001). The best result has been obtained using the first one, the highest accuracy is 74.38% reached at epoch 76. (Training time = 11394s (about 3 hours))

Figure 4 shows accuracy and loss during the training of the best model (lr=0.0001)

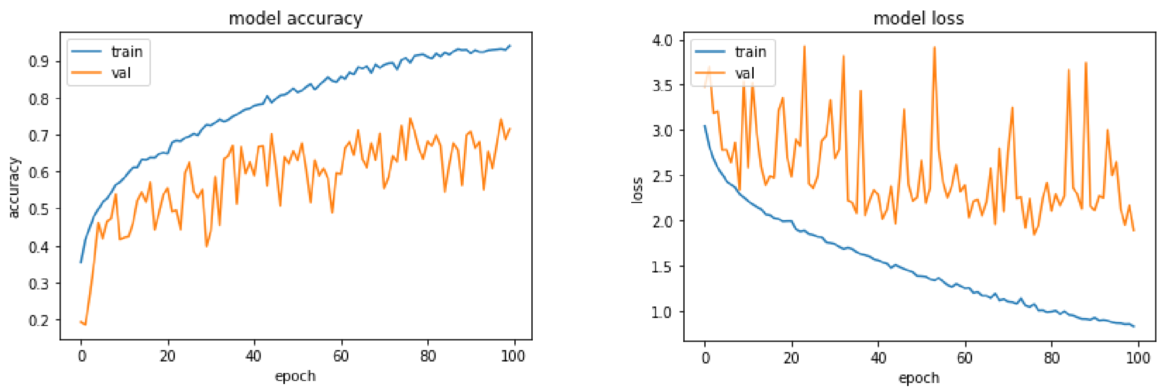


Figure 4: Accuracy and Loss in training and validation set during the training

VGG 16

The last convolutional neural network tested is VGG_16, the model pre-trained with ImageNet dataset, has been imported using keras applications. In Figure 5 is shown the architecture of the network without the last dense layers.

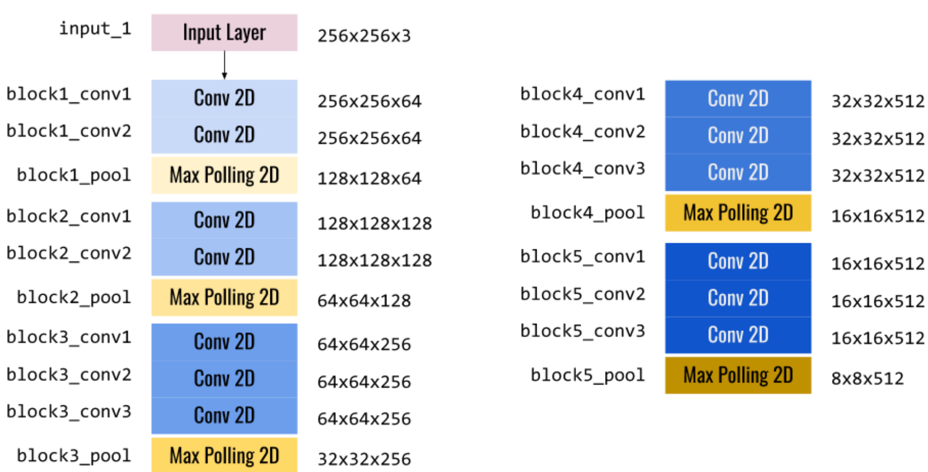


Figure 5: Convolutional Layers of VGG16 network

In this case several tests have been performed by changing the layers that can be trainable of VGG16 network and the different numbers of nodes in the last fully connected layers; however, the structure of the last layers is the following for every test.

```
Flatten
BatchNormalization
Dense (Dropout=0.4, Activation_Func=ReLu)
BatchNormalization
Dense (Dropout=0.4, Activation_Func=ReLu)
BatchNormalization

BatchNormalization
Dense (Nodes=8, Activation_Func=Softmax)
```

TEST 1: One trainable convolutional layer, dense 200,100

The only layer of VGG 16 network set to trainable is the last one (`block5_conv3`). The first dense layer has 200 nodes and the second instead 100.

The model has been trained for 100 epochs, it required 12013 seconds (about 3h 20 min), the best value of accuracy in the validation set has been reached at epoch 55 (val_accuracy=84.33% loss=0.7482). In Figure 6 is shown the training progress.

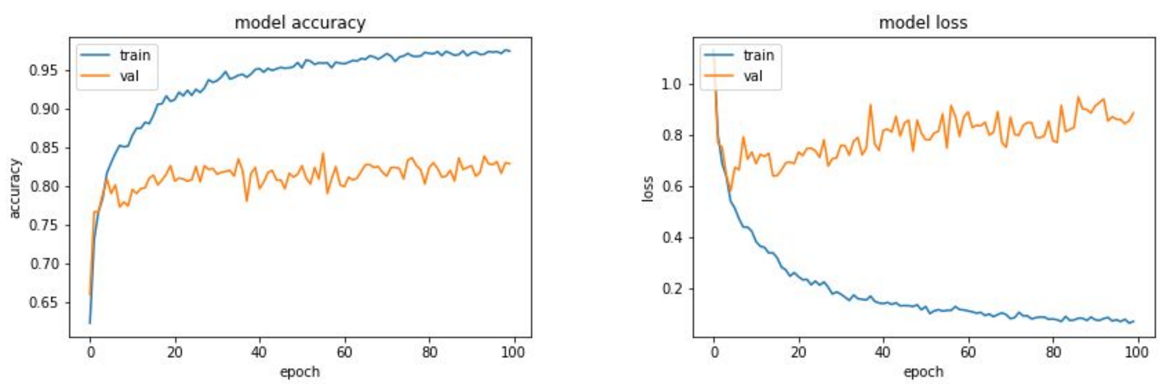


Figure 6: Accuracy and Loss in training and validation set during the training

Looking at the previous plots we can notice that the model stops to improve performance in the validation set very early, so it is necessary to add several trainable parameters. In the next two tests it was tried to add nodes in the dense layers and set to trainable also an another convolutional layer.

TEST 2A: One trainable convolutional layer, dense 300,200

In the second test, the dimension of the two dense layers is increased to 300 in the first, and 200 in the second one.

The model has been trained for 20 epochs, the highest accuracy obtained is 82.09% at epoch 16, Figure 7 shows the training.

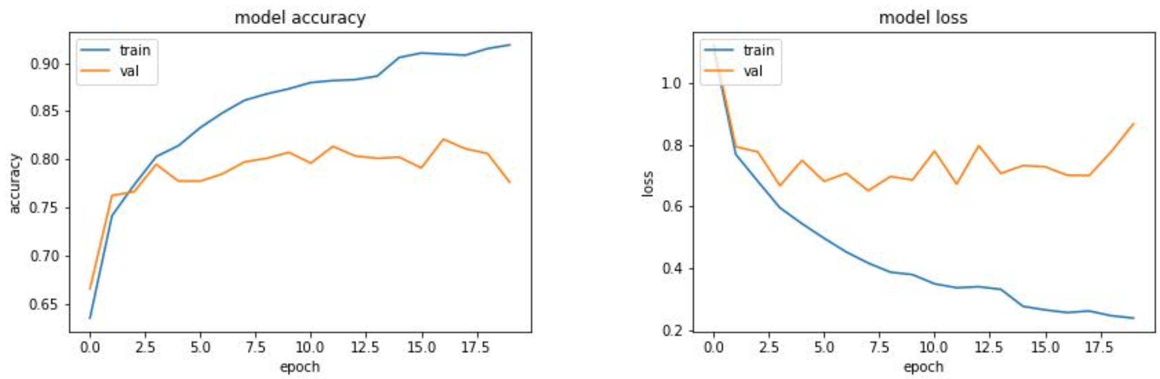


Figure 7: Accuracy and Loss in training and validation set during the training

TEST 2B: Two trainable convolutional layers, dense 200,100

In the third test instead, the dimensions of the dense layers are the same of TEST 1, but now two convolutional layers are trainable (`block5_conv3` and `block5_conv2`). Best accuracy is 82.21% reached at epoch 11)

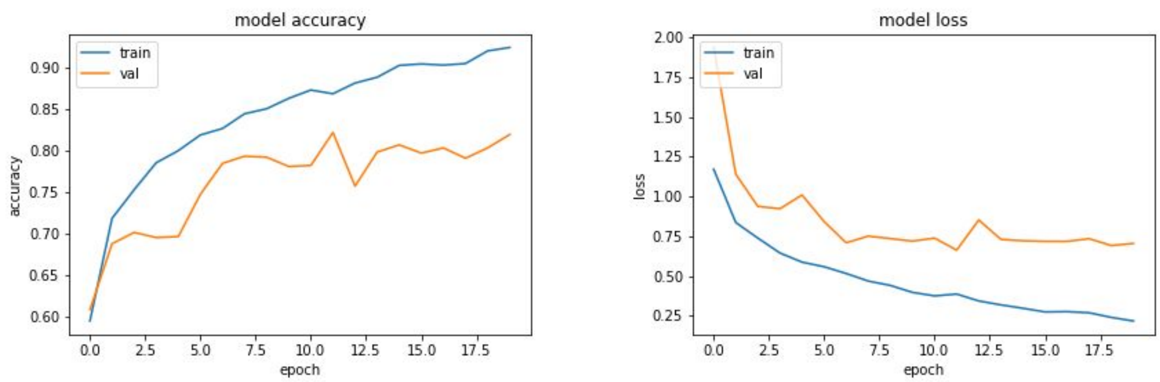


Figure 8: Accuracy and Loss in training and validation set during the training

TEST 3: Three trainable convolutional layers, dense 200,100 (100 Epochs)

In TEST 2A the training plots have a trend more flat than TEST 2B, so making trainable more convolutional layers is more effective than adding nodes in the last dense layers. For this reason, in the last test (TEST 3) the number of final nodes is the same of TEST 1 but instead of only one, the three final convolutional layers are trainable (block5_conv3, block5_conv2 and block5_conv1).

The model has been trained for 100 epochs, and it required 13458 seconds (about 3h 45min). The highest accuracy is 85.57% at epoch 85 with 0.8281 loss.

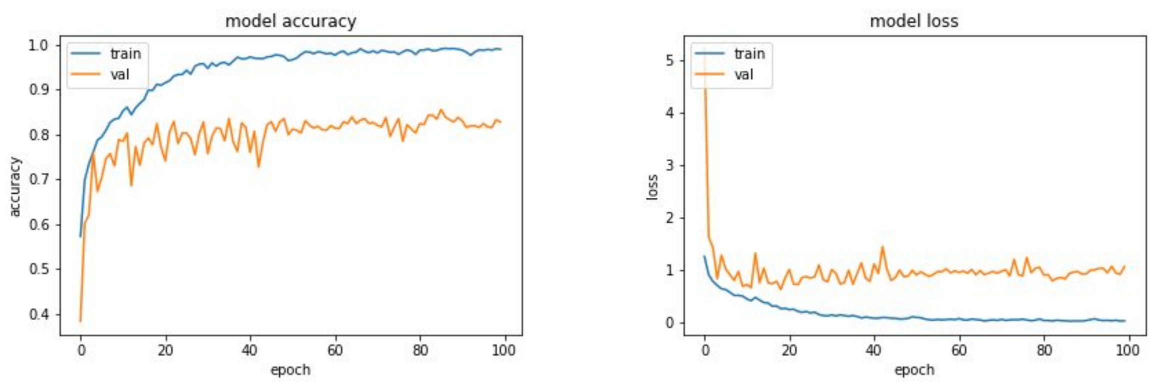


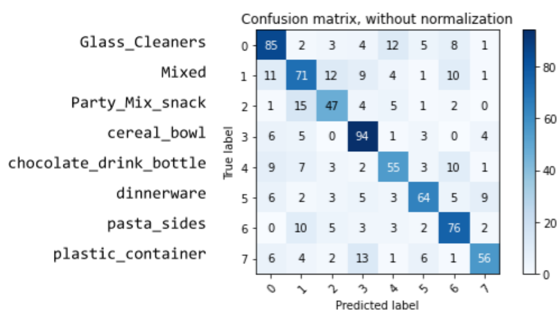
Figure 9: Accuracy and Loss in training and validation set during the training

RESULTS

In the following are shown the results obtained with the best model of each network in the validation set, and for the best of all, namely the one of the last experiment (TEST 3), also the results in the test set.

ResNet50

Results in validation set

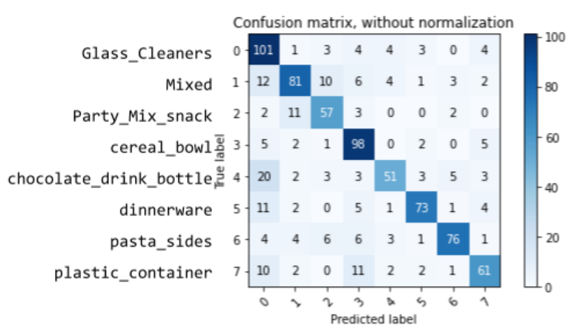


	precision	recall	f1-score	support
0	0.6855	0.7083	0.6967	120
1	0.6121	0.5966	0.6043	119
2	0.6267	0.6267	0.6267	75
3	0.7015	0.8319	0.7611	113
4	0.6548	0.6111	0.6322	90
5	0.7529	0.6598	0.7033	97
6	0.6786	0.7525	0.7136	101
7	0.7568	0.6292	0.6871	89
accuracy			0.6816	804
macro avg	0.6836	0.6770	0.6781	804
weighted avg	0.6831	0.6816	0.6802	804

Figure 10: ResNet50: Results in the validation set

AlexNet (lr=0.0001)

Results in validation set

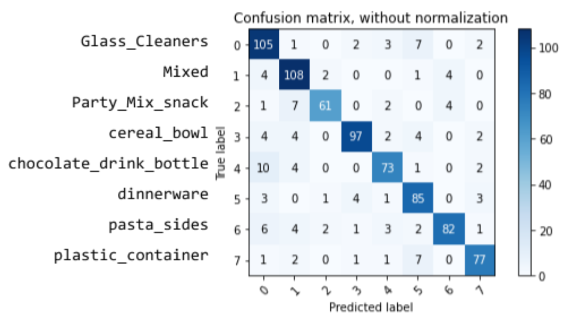


	precision	recall	f1-score	support
0	0.6121	0.8417	0.7088	120
1	0.7714	0.6807	0.7232	119
2	0.7125	0.7600	0.7355	75
3	0.7206	0.8673	0.7871	113
4	0.7846	0.5667	0.6581	90
5	0.8588	0.7526	0.8022	97
6	0.8636	0.7525	0.8042	101
7	0.7625	0.6854	0.7219	89
accuracy			0.7438	804
macro avg	0.7608	0.7383	0.7426	804
weighted avg	0.7576	0.7438	0.7435	804

Figure 11: AlexNet50: Results in the validation set

VGG 16

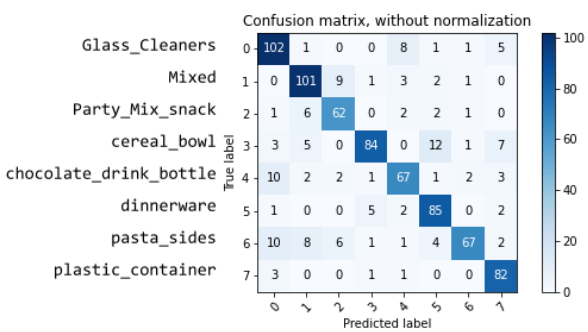
Results in validation set



	precision	recall	f1-score	support
0	0.7836	0.8750	0.8268	120
1	0.8308	0.9076	0.8675	119
2	0.9242	0.8133	0.8652	75
3	0.9238	0.8584	0.8899	113
4	0.8588	0.8111	0.8343	90
5	0.7944	0.8763	0.8333	97
6	0.9111	0.8119	0.8586	101
7	0.8851	0.8652	0.8750	89
accuracy			0.8557	804
macro avg	0.8640	0.8523	0.8563	804
weighted avg	0.8604	0.8557	0.8562	804

Figure 12: Results in the validation set

Results in test set



	precision	recall	f1-score	support
0	0.7846	0.8644	0.8226	118
1	0.8211	0.8632	0.8417	117
2	0.7848	0.8378	0.8105	74
3	0.9032	0.7500	0.8195	112
4	0.7976	0.7614	0.7791	88
5	0.7944	0.8947	0.8416	95
6	0.9178	0.6768	0.7791	99
7	0.8119	0.9425	0.8723	87
accuracy			0.8228	790
macro avg	0.8269	0.8239	0.8208	790
weighted avg	0.8292	0.8228	0.8213	790

Figure 13: Results in the test set

In the test set, the performance is lower than the validation set (about -3% of accuracy), the most common mistakes in the test set are the predictions of `cereal_bowl` as `dinnerware`, `chocolate_drink_bottle` and `pasta_sides` as `Glass_Cleaners`. However, it is important to note that in the dataset there are several wrong images that don't correspond to their label, the presence of these noises penalizes the performance in testing and limits the ability of the model to learn during the training.

CONCLUSIONS

In order to develop a classifier of home object images three different CNNs have been tested, the best performance has been obtained using transfer learning technique with VGG 16 network pre-trained in ImageNet dataset, the fine-tuning has been performed by training the last three convolutional layers and the fully connected ones. In this way good performance has been obtained also in few epochs, the others networks instead, (ResNet50 and AlexNet) require more iterations since they are composed by more layers, moreover, AlexNet has been trained from scratch.

In order to improve the performance obtained with VGG 16, it is necessary to remove the wrong images of the dataset to improve the training and make more truthful the testing, moreover the reduction of images size could be useful and can also speed up the training.