Michela Proietti 1739846

ABSTRACT

In this report, we propose the results obtained while solving an image classification problem using the RoboCup@Home-Objects dataset, that contains images of objects we typically find in a home environment. In particular, we considered only 8 of the available classes in order to train and test different neural networks. One of them was built from scratch, while other experiments have been done using transfer learning with MobileNet, and all the models have been tested both with and without data augmentation.

PRELIMINARIES

In all the algorithms presented above, we have used the same loss function in order to make fair comparisons. Specifically, we used categorical cross-entropy, that measures the performance of a classification method whose output is a probability. The cross-entropy loss is defined as follows:

$$CE = -\sum_{i=0}^{m} y_i \log(\hat{y}_i)$$

where y_i is the ground truth and m is the number of classes. As usual, before giving the result of the model as input to the loss function, we need to apply an activation function and the one we used was the softmax activation function. In this case, we talk about categorical cross-entropy loss. This combination was chosen because we are trying to solve a multiclass classification problem and therefore just one element in the vector y , the one that corresponds to the predicted class, must be different from 0. By using the softmax activation function just that particular class will be considered in the loss function. Consequently, starting from the softmax formula, we can write the categorical cross-entropy loss as follows:

$$f(\hat{y}_i) = \frac{e^{y_i}}{\sum_{j=0}^{m} e^{\hat{y}_i}} \quad \Rightarrow \quad CE = -\log \frac{e^{y_i}}{\sum_{j=0}^{m} e^{\hat{y}_i}}$$

For all the other layers, the activation function that has been used is the ReLU, whose advantage is that of having a very simple derivative, thus allowing fast computations, and we have also used same padding, so that the output images of each convolutional layer have the same dimensions of the input images. As optimization method we have used the Adam optimizer, that exploits the advantages of both momentum and RMSProp, thus allowing to get quickly close to the optimum and to reduce the oscillations simultaneously. In some of the experiments, we have also used max pooling in order to reduce the dimensions of the feature space, thus decreasing the number of parameters to learn and the number of computations. As regularization methods, we have chosen to use dropout, in order to consider a different subset of nodes in the network at each training iteration, and in some cases, we have also used early stopping, to avoid continuing the training even when the performance on the validation set decreases while the performance on the training set is still increasing, thus facing the problem of overfitting. In the experiments with the pretrained network, we also used batch normalization, which means there are layers that transform the input so that it will have a mean zero and a standard deviation of 1, in order to speed up the training and adding also another regularization effect.

DATASET

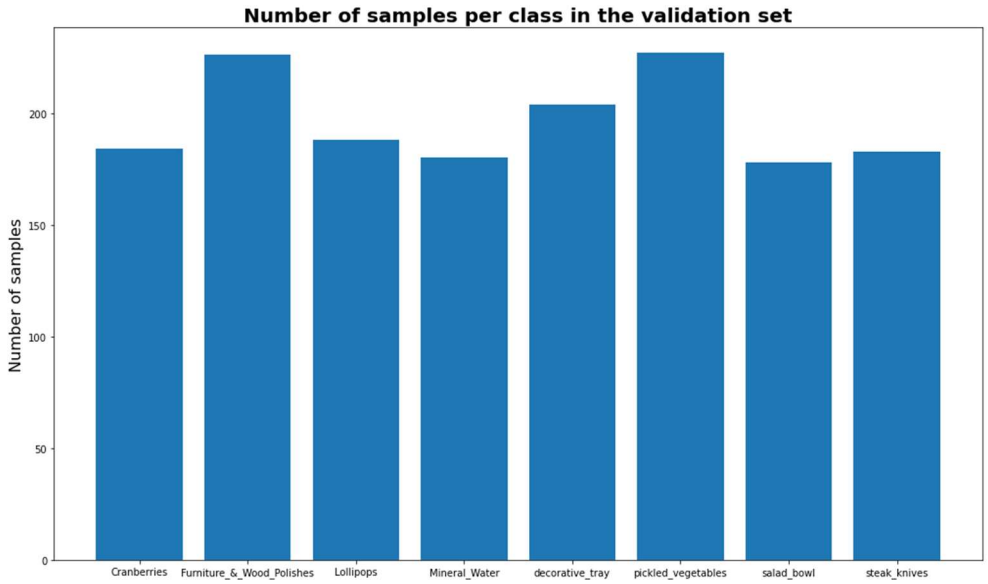
The dataset is made up of 9236 images belonging to 8 classes. The following table shows how data is split among the different classes and the image below shows one sample for each class.

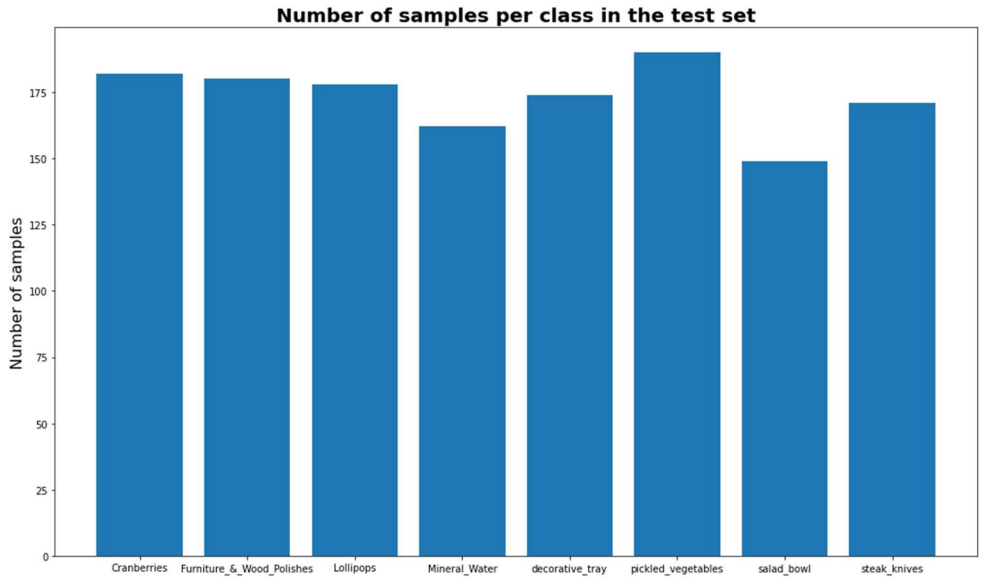
CLASS NAME	NUMBER OF SAMPLES
Cranberries	1146
Decorative tray	1155
Furniture & wood polishes	1321
Lollipops	1143
Mineral water	1076
Pickled vegetables	1164
Salad bowl	1113
Steak knives	1118

As we can see from the table, the dataset is well balanced, however each class contains images of objects that do not actually belong to that particular class, for example images of bags among lollipops images, and this represents a problem because those images can confuse our algorithm during training. For this reason, we did not expect high performances, independently from the algorithm that was being used.

All the images have been rescaled by performing a division by 255. Then, we split the dataset in training and test set and then we divided again the training set in training and validation set. The validation set has been used during the training in order to monitor the performance of the model, while the test set has been used for testing the trained model on new data. The following table shows how data have been divided among the three sets, while the images below show how samples in each of those sets are distributed among the different classes.

CLASS NAME	TRAINING SET	VALIDATION SET	TEST SET
Cranberries	780	184	182
Decorative tray	915	226	180
Furniture & wood polishes	777	188	178
Lollipops	734	180	162
Mineral water	777	204	174
Pickled vegetables	747	227	190
Salad bowl	786	178	149
Steak knives	764	183	171
TOTAL	**6280**	**1570**	**1386**





In the experiments that have been done on both the models that have been used, we also tried to perform data augmentation on the dataset. In particular, we used height and width shifts by specifying a percentage of the height /width of the image to shift equal to 10%. We also performed a horizontal flip and a rotation of the images by 20 degrees. Finally, we specified a zoom range of 0.1, which means that the range will be between 90% zoom in and 110% zoom out and the zoom amount will be uniformly randomly sampled from this interval. We have done some experiments in which we trained the models directly on the augmented training set, however in this case the results did not improve at all. So, we tried to add the augmented images to the training images we already had, thus getting a bigger training set, made up of 15516, that let us reach better results on the test set.

EXPERIMENTS

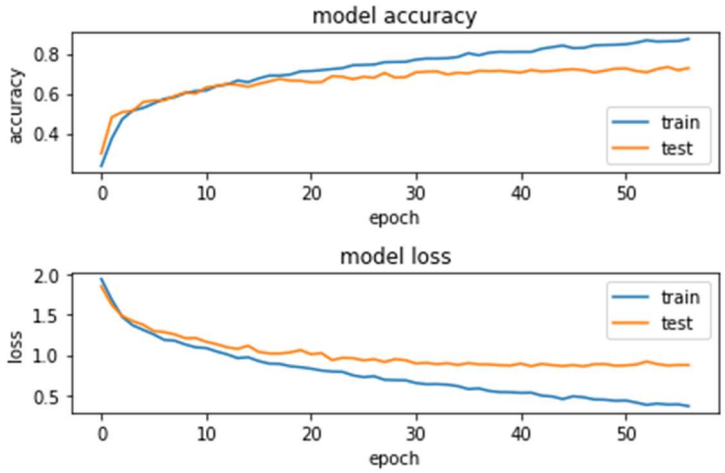
First of all, we are going to analyse the network built from scratch that has the following structure:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
```

```
conv2d_3 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 256)         0
_________________________________________________________________
global_max_pooling2d (Global (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 520
=================================================================
Total params: 405,384
Trainable params: 405,384
Non-trainable params: 0
_________________________________________________________________
```

It is made up of 4 convolutional layer of 32, 64, 128 and 256 nodes, that apply to the images 3x3 kernels and that use as activation function the ReLU function. We use same padding in order to keep the size of the output images equal to the size of the input ones, but then we apply max pooling with a 2x2 kernel. Moreover, at each layer we have added dropout with 0.3 probability for each node to be discarded at each training step. Then, we have a 2D global max pooling block, that performs the same operation of max pooling, but the pool size is equal to the size of the entire input of the block, so it computes a max value for each of the input channels, thus obtaining a 2D tensor with shape (batch_size, channels). Finally, there are two dense layers: the first one has 64 nodes and the ReLU function as activation function, while the last one has 8 nodes and the softmax function as activation function. This model has been obtained by starting with models that were simpler, and by adding time after time other convolutional layers, changing the number of nodes in each layer, changing the value for dropout and adding additional tools, like max pooling and early stopping. Other tries were done relatively to the batch size and the number of epochs and we ended up using a batch size of 64 and 100 epochs. The patience in early stopping was set to 10, in fact without early stopping the training continued but there was no improvement in the performances on the validation set, while if we used a lower patience we did not manage to get to the highest reachable accuracy. The following table and plots show the results obtained using this network.

Test accuracy	Test loss	Correctly classified	Incorrectly classified	Predict time (s)	Epoch time (s)
0.74	0.83	1026	360	0.23	1

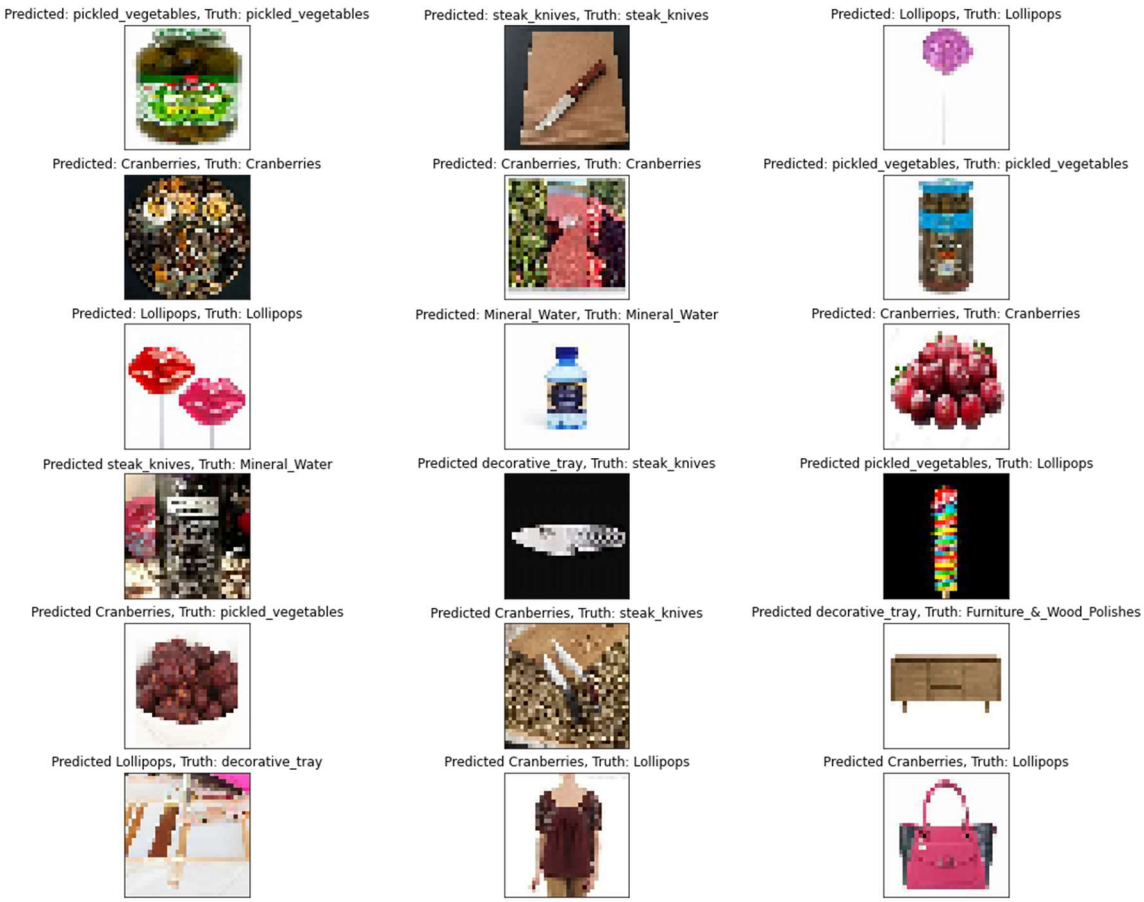


As we can see, most of the samples are correctly classified, and both the training and the prediction on the test set are pretty fast. Moreover, it seems like there is not overfitting, since the accuracy on the validation set increases accordingly to the one on the training set. In fact, even if we continued the training, both accuracies did not increase. This means that the network is not powerful enough to learn other features, but even changing the structure of the

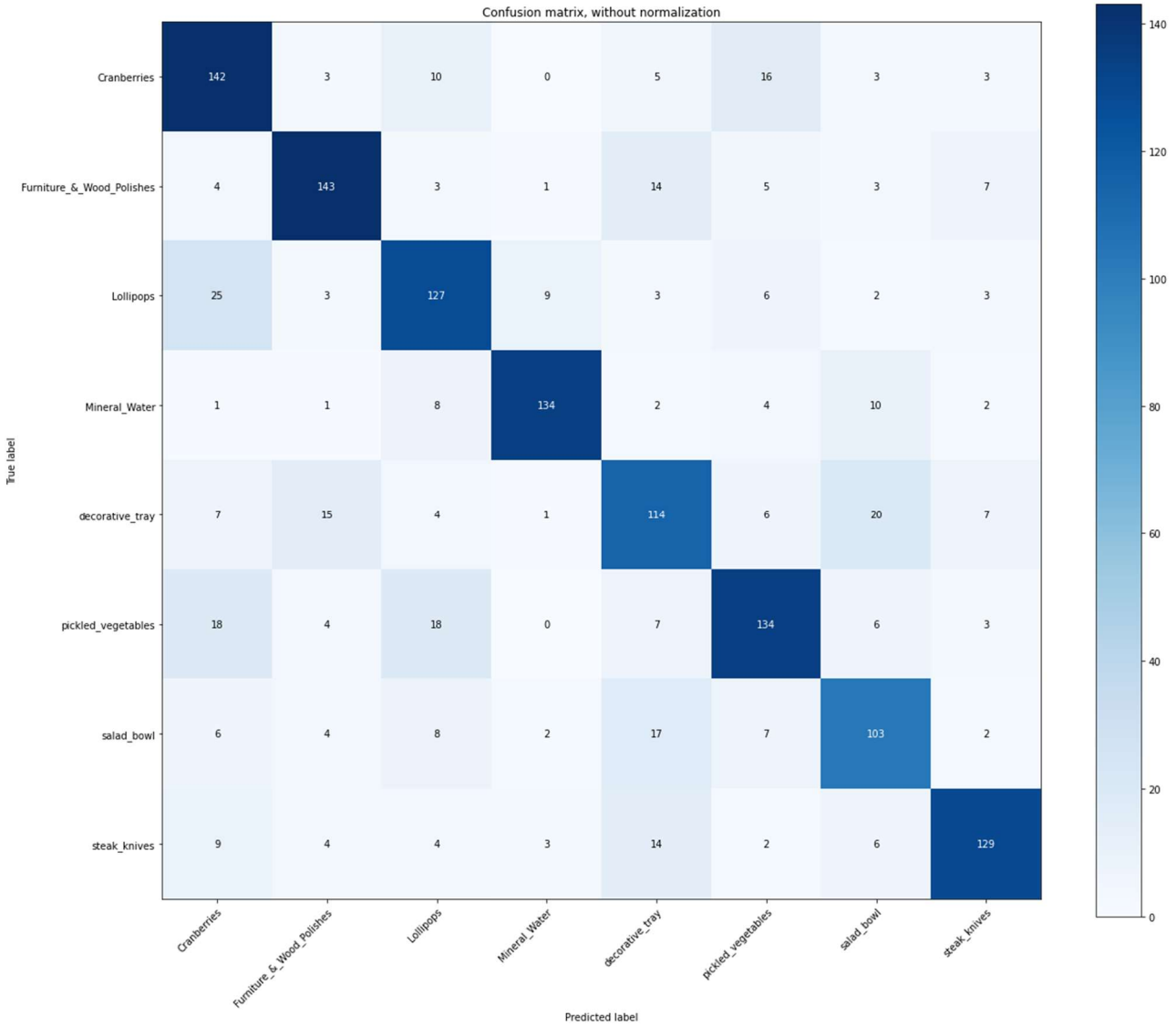
network we did not manage to get better results. In particular, by adding other convolutional layers the performances did not improve, and in most of the cases they worsened. Therefore, we chose to focus on the model that gave the best results, and we considered other metrics, namely precision, recall and F1-score. The following table shows the values obtained.

	Precision	Recall	F1-score	Support
Cranberries	0.67	0.78	0.72	182
Furniture & wood polishes	0.81	0.79	0.80	180
Lollipops	0.70	0.71	0.71	178
Mineral water	0.89	0.83	0.86	162
Decorative tray	0.65	0.66	0.65	174
Pickled vegetables	0.74	0.71	0.72	190
Salad bowl	0.67	0.69	0.68	149
Steak knives	0.83	0.75	0.79	171
Accuracy			0.74	1386
Macro avg	0.75	0.74	0.74	1386
Weighted avg	0.74	0.74	0.74	1386

As we can see, the class on which the model makes more mistakes is decorative tray, while the one in which it has the best performance is mineral water, but it is also the one for which we have the lowest number of test samples, so this might be the cause of this slight difference in performance. This is also confirmed by the confusion matrix that is reported in the next page, and that clearly shows on which classes the network does more mistakes. Here, we reported some random examples of the classifications done by the network.



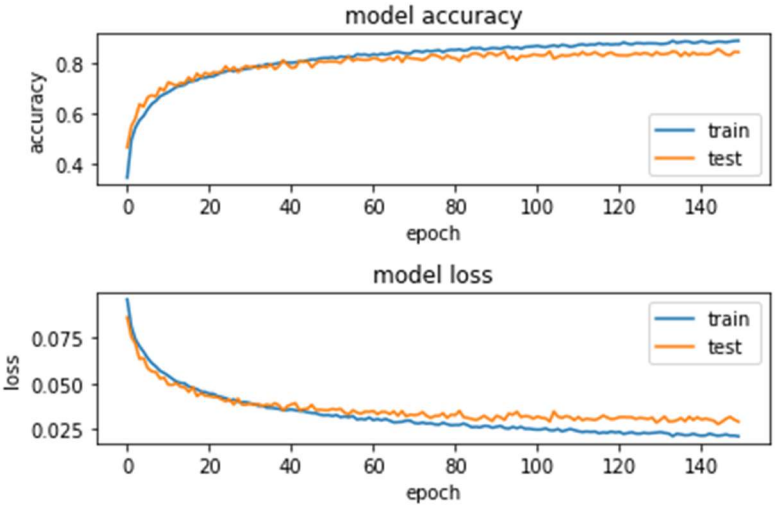
The last two examples of incorrectly classified images are two samples belonging to the class lollipops that are classified as belonging to cranberries. In fact, from the confusion matrix we can see that 25 out of 51 errors done on the class lollipop are caused by the fact that the network predicted the class cranberries. In the same way, the model makes a lot of mistakes on the class cranberries by predicting pickled vegetables and the fourth example of incorrectly classified image shows exactly this. The same also happens for the class pickled vegetables, often confused with the class lollipops, and for steak knife, often confused with the class decorative tray. However, the results of the network are not bad and at the same time this model is very fast.



Then, we trained this model on the augmented dataset and we got much better results, that are shown in the following table.

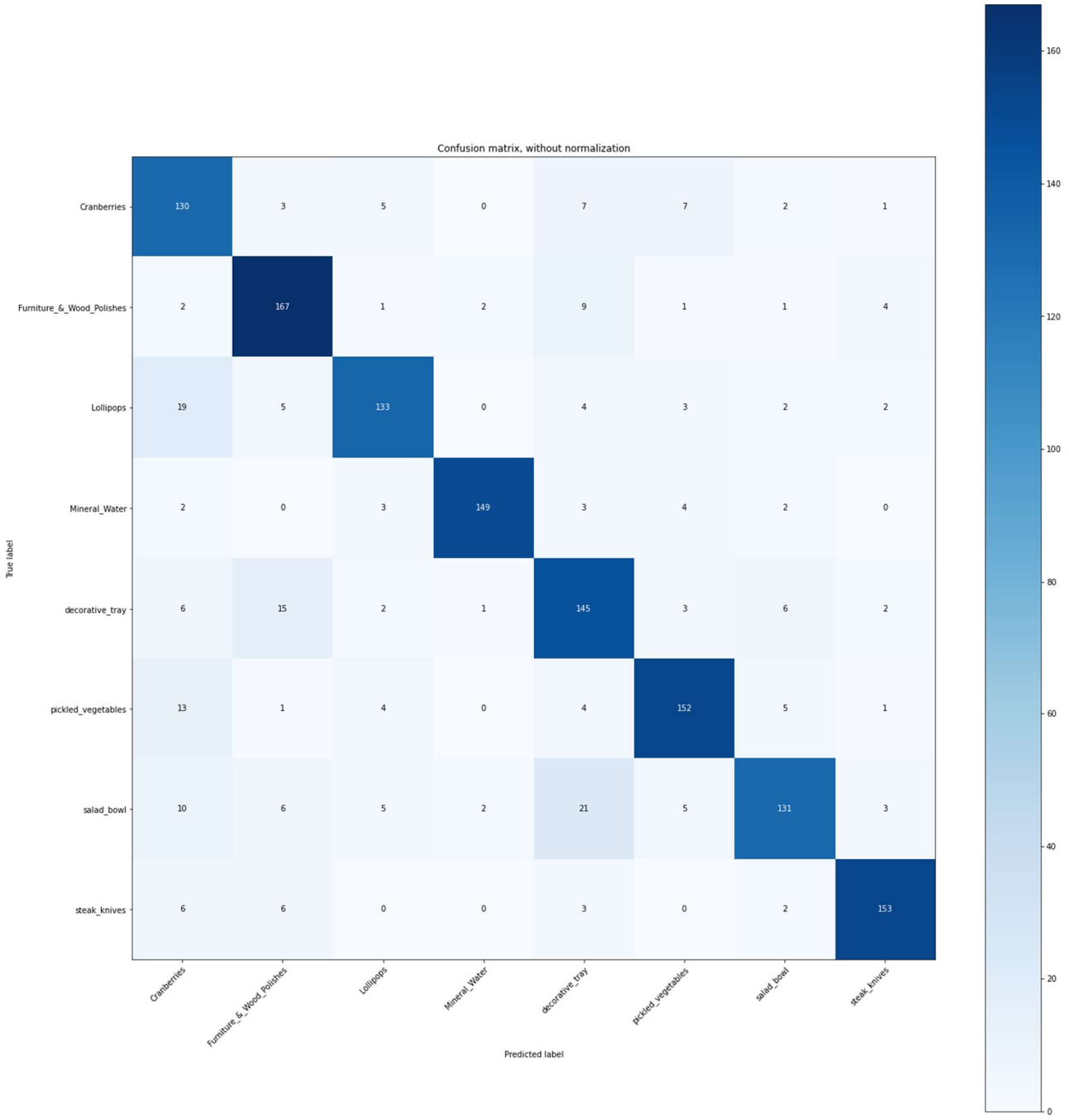
Test accuracy	Test loss	Correctly classified	Incorrectly classified	Predict time (s)	Epoch time (s)
0.8369	0.0297	1160	226	0.14	3.5

In this case, the accuracy on the validation set gets closer to the one on the training set. Even in this experiment, we tried to increase the number of epochs to try to get to higher accuracy, and we found out that by using 150 epochs we manage to get to the highest accuracy, while when we tried greater values just



the accuracy on the training set increased, so the model was overfitting the training data.

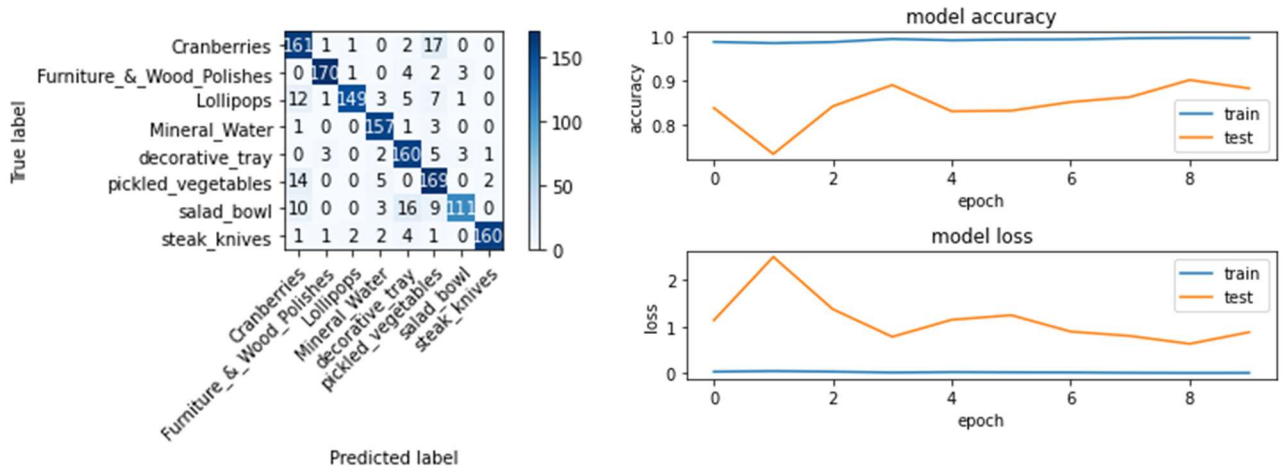
Looking at the confusion matrix we can notice that in this case the class on which the network has the best performance is still mineral water, while the one for which it gives more incorrect predictions is salad bowl.



Finally, we tried to use transfer learning by using MobileNet. This model has 27 convolutional layers and a total number of 4.2 million parameters. It is formed by 2 units: the first one is characterized by a 3x3 convolution, followed by batch normalization and ReLU activation function, while the second one is a 3x3 depth wise convolution, followed by batch normalization and ReLU activation function and by a 1x1 convolution that is as well followed by batch normalization and ReLU activation function. This block is repeated 13 times and then we have an average pooling layer, a fully connected layer and a final softmax layer. The total number of layers is 88, but we considered all the layers up to the last global average pooling layer, thus ignoring the last 5 layers, and then we added a dense layer with 8 nodes and a softmax activation function, that will give us the output of our new model. However, we need to train also some layers from MobileNet and we found out that training the last 30 layers of the network gives the best results, that are those showed in the following table.

Test accuracy	Test loss	Correctly classified	Incorrectly classified	Predict time (s)	Epoch time (s)
0.8925	0.7262	1237	149	5.5469	29

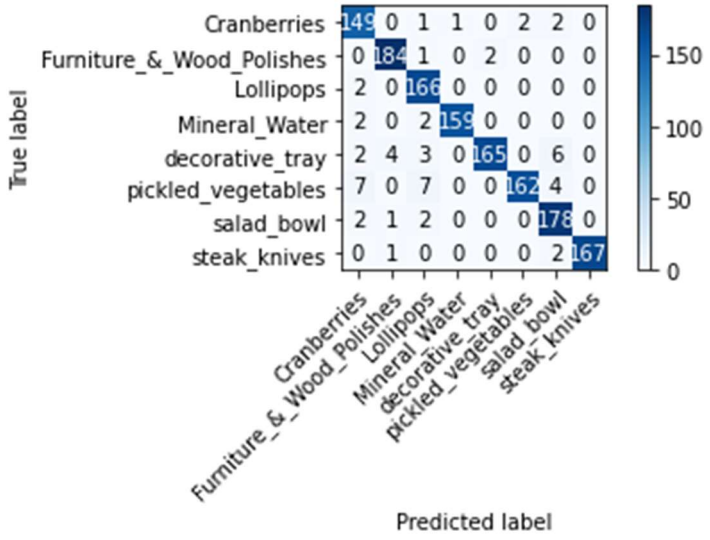
This experiment has been carried out without performing data augmentation, but as we can easily see, the results we get are much better than those we obtained with the previous model trained on the augmented dataset. However, the time required for both training and giving the predictions for the test dataset is much longer. From the plot below, we can see there are much more oscillations in the accuracy and the loss on the validation data.



After this, we trained the same model on the augmented dataset, by using a batch size equal to 100 and 10 learning epochs. In this case, the results we get are even higher and they are reported in the following table.

Test accuracy	Test loss	Correctly classified	Incorrectly classified	Predict time (s)	Epoch time (s)
0.9596	0.1388	1330	56	5.5625	66.6

As we can see, the training time is more than two times greater than before, while the predict time did not increase. However, now our model manages to correctly classify almost all the images in the test set, as we can easily see from the following confusion matrix.

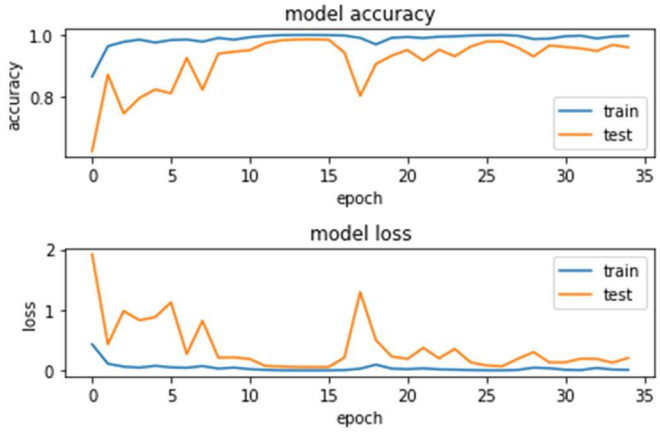


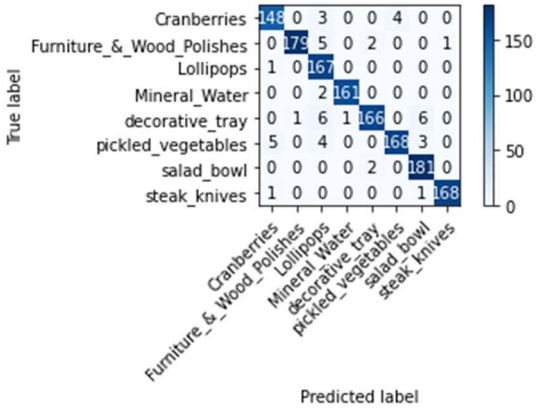
If we look at the classification report, we notice that for the class steak knife, we get precision 1, in fact from the confusion matrix we can see that no sample from the other classes is predicted as a steak knife. The class with the lowest recall is pickled vegetables and in fact in the confusion matrix we see that 7 images have been predicted as cranberries, 7 images as lollipops and 4 images as salad bowl. Instead, the one with the lowest precision is the cranberries class, because except for furniture & wood polishes and steak knife, for all the other classes we have that some samples are incorrectly predicted as belonging to cranberries.

	Precision	Recall	F1-score	Support
Cranberries	0.91	0.96	0.93	155
Furniture & wood polishes	0.97	0.98	0.98	187
Lollipops	0.91	0.99	0.95	168
Mineral water	0.99	0.98	0.98	163
Decorative tray	0.99	0.92	0.95	180
Pickled vegetables	0.99	0.90	0.94	180
Salad bowl	0.93	0.97	0.95	183
Steak knives	1.00	0.98	0.99	170
Accuracy			0.96	1386
Macro avg	0.96	0.96	0.96	1386
Weighted avg	0.96	0.96	0.96	1386

Finally, we tried to increase the number of epochs to see if the accuracy on the validation set continued to grow during the training and we managed to ulteriorly improve the performance of the model by setting the number of epochs to 35. The following table shows the results we got.

Test accuracy	Test loss	Correctly classified	Incorrectly classified	Predict time (s)	Epoch time (s)
0.9654	0.103	1338	48	5.2187	68





	Precision	Recall	F1-score	Support
Cranberries	0.95	0.95	0.95	155
Furniture & wood polishes	0.99	0.96	0.98	187
Lollipops	0.89	0.99	0.94	168
Mineral water	0.99	0.99	0.99	163
Decorative tray	0.98	0.92	0.95	180
Pickled vegetables	0.98	0.93	0.95	180
Salad bowl	0.95	0. 99	0.97	183
Steak knives	0.99	0.99	0.99	170
Accuracy			0.97	1386
Macro avg	0.97	0.97	0.97	1386
Weighted avg	0.97	0.97	0.97	1386

The figure on the top left part of the page shows the accuracy and the loss for both the training and the validation set during the training. As we can see, there are big oscillations in the first 17 epochs for the validation set, but then even if the curves are not smooth, the oscillations are much reduced. In this case, we do not focus on the classification report or the confusion matrix, because they are very similar to the ones we showed previously, but we prefer to show some examples of incorrectly classified images (next page).
Our model made mistakes on images that were very difficult to classify, in which the object to classify is in a very wide environment with many other objects, or for example it can happen that pickled vegetables are found inside a bowl and therefore the model predicts salad bowl instead of pickled vegetables. As said before, there are also many images in the dataset that have nothing to do with the class they belong to. In the examples below, we can see an eggplant, belonging to class salad bowl, that is predicted as a decorative tray. Moreover, as said in the part of the report about the dataset, there are many images of bags inside the class lollipops and therefore the model classifies the sixth image in the right column as belonging to lollipops, even if it actually belongs to pickled vegetables. In the examples on its side, we can see the model predicted cranberries instead of pickled vegetables, but we can see there is red round food in those dishes, so it is likely to make mistakes on them. In the eighth example on the right, pickled vegetables are inside a salad bowl and so it is comprehensible that the model classifies the image as belonging to the class salad bowl. This holds for many other of the images shown below, and therefore the results obtained are even more surprising, considered that the model managed to correctly classify most of the test images and that there might have been some outliers in the training set as well and the model has not been confused by those.

Predicted Lollipops, Truth: Furniture_&_Wood_Polishes

Predicted salad_bowl, Truth: decorative_tray

Predicted Lollipops, Truth: Cranberries

Predicted decorative_tray, Truth: salad_bowl

Predicted salad_bowl, Truth: decorative_tray

Predicted Lollipops, Truth: Cranberries

Predicted Lollipops, Truth: decorative_tray

Predicted Lollipops, Truth: Furniture_&_Wood_Polishes

Predicted salad_bowl, Truth: pickled_vegetables

Predicted Lollipops, Truth: decorative_tray

Predicted pickled_vegetables, Truth: Cranberries

Predicted Cranberries, Truth: pickled_vegetables

Predicted Cranberries, Truth: steak_knives

Predicted decorative_tray, Truth: salad_bowl

Predicted Lollipops, Truth: Cranberries

Predicted Cranberries, Truth: pickled_vegetables

Predicted Cranberries, Truth: pickled_vegetables

Predicted Lollipops, Truth: pickled_vegetables

Predicted Lollipops, Truth: Mineral_Water

Predicted Lollipops, Truth: decorative_tray

Predicted Cranberries, Truth: Lollipops

Predicted Lollipops, Truth: decorative_tray

Predicted decorative_tray, Truth: Furniture_&_Wood_Polishes

Predicted salad_bowl, Truth: pickled_vegetables

Predicted Furniture_&_Wood_Polishes, Truth: decorative_tray

Predicted Lollipops, Truth: Furniture_&_Wood_Polishes

Predicted pickled_vegetables, Truth: Cranberries

Predicted salad_bowl, Truth: decorative_tray

Predicted Lollipops, Truth: Mineral_Water

Predicted steak_knives, Truth: Furniture_&_Wood_Polishes