Homework 2 Machine Learning
Image classification

Santilli Sofia
matricola 1813509

December 2020

1 Description of the problem

The goal of this work is to classify a series of images from a dataset of 8745 files. These images belong to eight different classes:

- 1007 images belong to *dust mops and pads*;
- 1051 images belong to *figs*;
- 1101 images belong to *food*;
- 1206 images belong to *fruit plate*;
- 1059 images belong to *instant soup*;
- 1170 images belong to *nuts snacks*;
- 1064 images belong to *plastic knife*;
- 1087 images belong to *Seltzer water*.

Between them, there are some random images that don't belong to the respective class: these samples have been introduced in order to make the prediction a bit more difficult.
These images have dimensions of 118x224x3, where 118 is the width and 224 the height; 3 represents the fact that we're working with RGB images.

In order to do the classification, we rely on the Neural Networks.

2 Split the dataset

The first thing to do is to split the dataset's folders of each class of images in three groups:

- the *training set*, used in order to fit the parameters of the model;
- the *validation set*, used during the training. The fitted model is used to predict the responses for the observations in this second set of images. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters;
- the *test set*, which is not used during the training; the final fitted model is applied to the test set in order to provide an evaluation of this model (its accuracy and other relevant data).

In order to do this split, we use the function 'ratio' of the library '*split-folders tqdm*'. The percentages in which the dataset is divided are:

- 80% in the training set (6992 images);
- 10% in the validation set (871 images);
- 10% in the test set (882 images).

This function copies the images from the initial folder of the dataset to a new folder, in the percentages indicated above. So now we have three new folders, one for each set of images; every folder is divided in turn in the eight folders of the different classes.

3 Loading of the data

The data are loaded from the new folders in which we have the partitioned dataset. We apply to the three sets of data the Keras's class *ImageData-Generator*, which generates batches of tensor image data with real-time data augmentation.
The batch size used is 32: it defines the number of samples that will be propagated, at the same time, through the network.
Moreover I also replace, in the case of the training data, the original batch

with the new randomly transformed batch: I rescale, apply a zoom, randomly rotate and translate (on both the horizontal and vertical position), flip the image horizontally. In fact, applying image augmentation allows us to do a random image transformation to reduce model overfit.

4 Convolutional Neural Networks

In deep learning, convolutional neural network is a class of deep neural networks. It's a type of feed-forward artificial neural network. They have several applications in analyzing visual imagery.
CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is when each neuron in one layer is connected to all neurons in the next layer. They take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.
The CNNs applied here have a structure divided in two parts:

- the first layers are a stack of convolutional and pooling layers, in order to generate features from the images;
- fully connected layers, in order to classify the images based on the detected features.

In this work we will apply to the splitted dataset two types of convolutional neural networks:

- AlexNet;
- TransferNet.

In the case of the TransferNet, we exploit a pretrained model from *ImageNet*, the *VGG16 model*.

In the following sections the predictions of these two models are reported.
In particular we're interested in loss and accuracy.
A loss function is used in order to optimize the algorithm. It's calculated on training and validation and its interpretation is based on how well the model is doing in these two sets.

4.1 AlexNet

AlexNet provides several convolutional layers in succession, instead of alternating them at the pooling level.
The layers are eight: the first five are convolutional layers (some of them followed by max-pooling layers) and the last three are fully connected layers. It uses the non-saturating ReLU activation function, which shows improved training performance over tanh and sigmoid.
Moreover AlexNet allows us to treat 3 channels and so to manage color images (the RGB scheme).

In the sequel we're reporting the predictions obtained applying the AlexNet model with different numbers of epochs and we're comparing the results obtained.
The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset.[1]

4.1.1 Ten epochs

After the first epoch, we have obtained on the training set an accuracy of 0.25 and a loss of 3.03, while on the validation set there's an accuracy of 0.14 and a loss of 2.9. After ten epochs, as recognizable in Figure 1, we reach on the training set an accuracy of 0.47 and a loss of 2.12; on the validation set an accuracy of 0.38 and a loss of 2.52.
So, how expected, as the number of epochs increases, accuracy raises while loss decreases. But we still get inefficient results: applying the model trained for ten epochs to the test set, we have an accuracy of 0.38 and a loss of 2.48. Let's increase the number of epochs.

4.1.2 Thirty epochs

The model, after a train of 30 epochs, manages in reaching an accuracy of 0.65 and a loss of 1.5 in the training set; an accuracy of 0.52 and a loss of 2 in the validation set. Applying it on the test set, the accuracy is 0.58, the loss is 1.83.
The predictions made through thirty epochs are better than the ones of ten

[1] One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters.

	precision	recall	f1-score	support
Dust_Mops_&_Pads	0.745	0.716	0.730	102
Figs	0.843	0.708	0.769	106
Nuts_snacks	0.686	0.598	0.639	117
Seltzer_Water	0.755	0.645	0.696	110
food_tray	0.511	0.649	0.571	111
fruit_plate	0.439	0.623	0.515	122
instant_Soup	0.825	0.617	0.706	107
plastic_knife	0.800	0.785	0.792	107
accuracy			0.666	882
macro avg	0.701	0.668	0.677	882
weighted avg	0.695	0.666	0.673	882

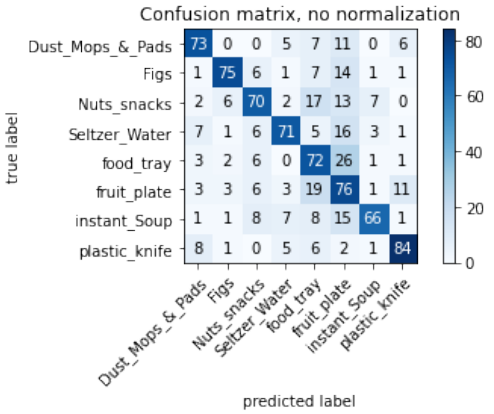

Figure 1 AlexNet, 100 epochs. Left: classification report. Right: confusion matrix.

epochs. Now we want to examine the case of one hundred epochs, in order to understand what changes increasing so much the number of epochs.

4.1.3 One hundred epochs

With one hundred epochs the training reaches an accuracy of 0.94 and a loss of 0.37. Regarding the validation set, the accuracy is 0.65 and the loss 1.77. Applied to the test set, accuracy is 0.66, loss is 1.75. Looking at the confusion matrix in Figure 1, we see the darker squares on the diagonal, which is understandable because an accuracy of about 70% is quite a good result, also if we could hope for better predictions.
From this confusion matrix we can see that there are some wrong predictions for all the classes of images. The dataset in fact is quite balanced: more or less there is the same number of samples for each class. If we want to be more accurate, the confusion matrix tells us that the most common error is due to the predicted labels for 'fruit plate' and 'food tray', which in reality belong to other classes.
This situation is recognizable also in the classification report (Figure 1), in which the values, for every class, are around 0.7 and are a bit lower for 'food tray' and 'fruit plate' classes.

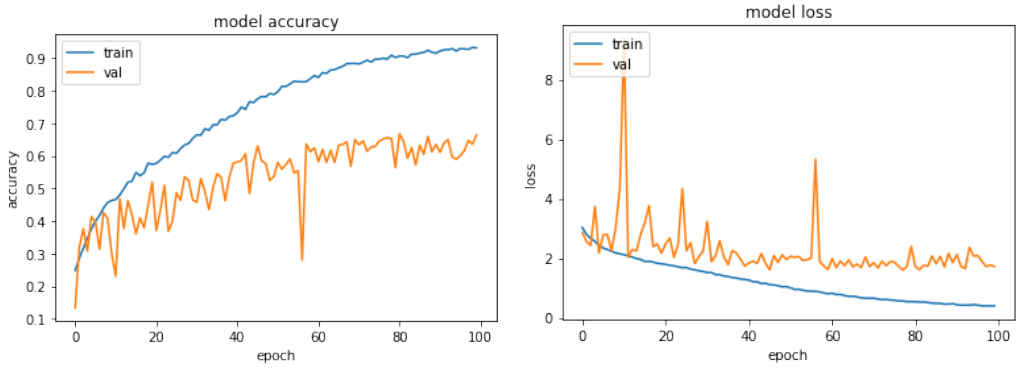

Figure 2 AlexNet, 100 epochs. *Accuracy* (left) and *Loss* (right). Blue is for the training set, orange for the validation set.

Now we focus on the results obtained from the predictions done on the train and the test set, in particular comparing these results with what we get when we used thirty epochs.
Regarding the training set, there are big improvements:

- accuracy passed from 0.65 to 0.94;
- loss passed from 1.5 to 0.37.

It may seem a very good outcome, since the general goal is to have accuracy tending to 1 and loss tending to 0. However the increase in the number of epochs doesn't bring the same benefits to the predictions done on the test set:

- accuracy passed from 0.58 to 0.66;
- loss passed from 1.83 to 1.75.

So, increasing OF about three times the numbers of epochs and computations, the results increase only in training and not so much in test set (in which we are interested in). So, with one hundred epochs, there is a situation of overfitting and a waste of resources.

For this reason now let's move on to making predictions with another Convolutional Neural Network.

	precision	recall	f1-score	support
Dust_Mops_&_Pads	0.888	0.853	0.870	102
Figs	0.852	0.868	0.860	106
Nuts_snacks	0.811	0.624	0.705	117
Seltzer_Water	0.854	0.800	0.826	110
food_tray	0.713	0.694	0.703	111
fruit_plate	0.661	0.689	0.675	122
instant_Soup	0.723	0.879	0.793	107
plastic_knife	0.856	0.944	0.898	107
accuracy			0.789	882
macro avg	0.795	0.794	0.791	882
weighted avg	0.792	0.789	0.788	882

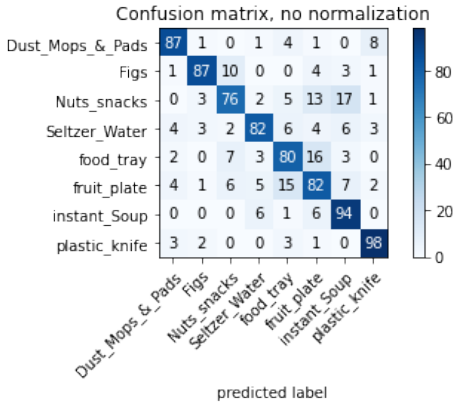

Figure 3 TransferNet, 20 epochs. Left: classification report. Right: confusion matrix.

4.2 TransferNet

In this section we rely on transfer learning: instead of starting the learning process from scratch, we apply a pre-trained model. This type of model was previously trained on a large benchmark dataset to solve a problem similar to the one that we want to solve. So in this process we retrain the TransferNet model but allow only the last few layers to change.
Both the TransferNet model and the pre-trained model used in this section are based on Convolutional Neural Networks.
The pre-trained model used is the VGG-16. It is an improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG-16 seems to be a good starting point as it resulted in having a good accuracy on the Imagenet database.

4.2.1 Twenty epochs

In the training, TransferNet reaches an accuracy of 0.86 and a loss of 0.39; in the validation set, an accuracy of 0.79 and a loss of 0.76.
Evaluating the model on the test set, we obtain an accuracy of 0.79 and a loss of 0.73.

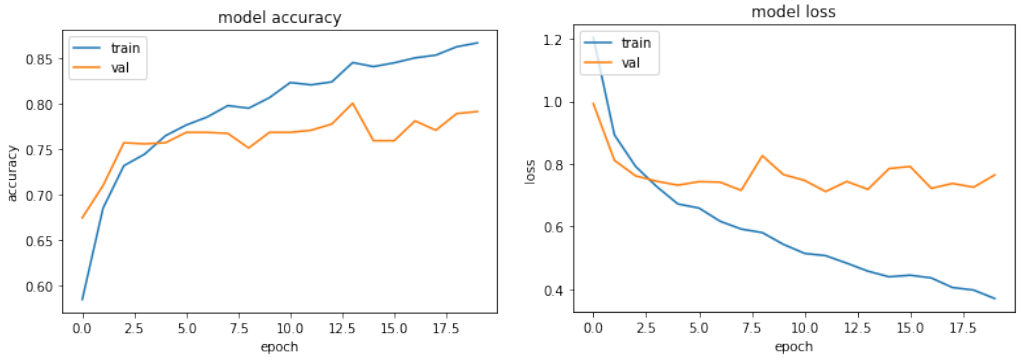

Figure 4 TransferNet, 20 epochs. *Accuracy* (left) and *Loss* (right). Blue is for the training set, orange for the validation set.

Comparing these results with those obtained from the AlexNet with 100 epochs, we see that:

- in the predictions on the test set, accuracy and loss are better in the case of TransferNet;
- in the predictions on the training set, in TransferNet, loss is better while accuracy is a bit lower.

So, since what we are really interested in is having the best possible results in the predictions on the test set, we can say that applying TransferNet (with only 20 epochs) is a better choise than AlexNet (with one hundred, or also fifty, epochs).
The most relevant observations that we can do by looking at the confusion matrix (Figure 3) are:

- 'food tray' images and 'fruit plate' tend to be exchanged;
- 'nuts snacks' images tend to be misclassified (in particular they are predicted by the model as 'fruit plate' or 'instant soup' images).

These observations are recognizable also from the histograms in Figure 5. In fact the images predicted to be 'nuts snacks' (number 2) are fewer in number than they really are. On the contrary, the images predicted as 'food tray' (number 5) and 'fruit plate' (number 6) are more than we expect. This is confirmed in the classification report: we can see that the lowest values (which consist in a percentage of 60%-70%) correspond to the rows related

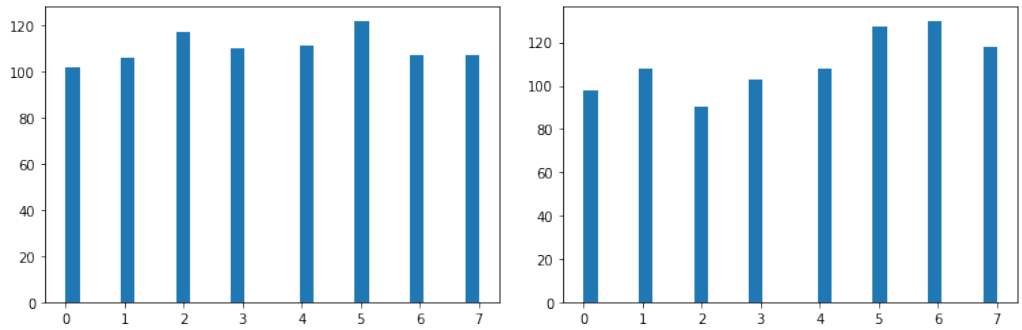

Figure 5 TransferNet, 20 epochs. Number of images for each class, in the test set (left). Prediction done on the test set (right).

to the three classes mentioned above.
Now we're going to see if the results for the test set get better if we use fifty epochs in the training.

4.2.2 Fifty epochs

With fifty epochs we see the same situation already encountered in the case of AlexNet with 100 epochs. In fact, with respect to the paragraph 4.2.1 , we have better results in training (accuracy=0.96, loss=0.2), but almost equal results in validation and testing.
Therefore, computing the training with this elevate number of epochs, is only cause of overfitting and doesn't bring improvements for the predictions obtained.

4.2.3 Twenty epochs, with some changes in the parameters of the model

Until now we have been working with the adam optimizer and the data augmentation presented in the paragraph 3 '*Loading of the data*'.
First we consider the *sgd optimizer* instead of the *adam*. In this case we obtain the following results:

- training set: accuracy of 0.84, loss of 0.46;
- validation set: accuracy of 0.78, loss of 0.68;

- test set: accuracy of 0.78, loss of 0.70.

So, with respect to the model with the adam optimizer, we obtain almost the same values. So the type of optimizer applied doesn't affect significantly the predictions in this case.

Now we consider a training set which hasn't been transformed randomly through the *ImageDataGenerator* and we see how the resulting predictions change:

- training set: accuracy of 0.96, loss of 0.10;
- validation set: accuracy of 0.77, loss of 0.97;
- test set: accuracy of 0.80, loss of 0.95.

Therefore, not transforming randomly the samples in the training dataset, the values in the case of the training set are a little better, while, in the predictions for the test set, loss is a bit higher.
Thus we can infer that, if the samples in the training set are slightly modified, our model will train better. This results in a higher number of correct predictions on the images in the test set.

5 Conclusions

In this work it is given a dataset of images belonging to eight different classes. The approach used in order to do the classification of these images applies the Convolutional Neural Networks, that have many applications in recognizing images.
Two models have been used to do the predictions: AlexNet and TransferNet. From the first we reach, recurring to 100 epochs in the training, about 65% of accuracy in the evaluation of the model on the test set, but we don't manage in obtain a loss lower than 1.7. Moreover these results are quite similar using about a third of the epochs; so a training with 100 epochs is only an overuse of the resources, which causes overfitting.
Instead of using AlexNet, we found a better model in the Transfernet CNN. It exploits the VGG16 Imagenet pretrained model.
We have found out that, through a training of 20 epochs, the previsions on the test set obtained consist in an 80% of accuracy and one of the lowest loss

obtained among all the tests carried out. It's not a perfect prediction, but it's still good. Moreover, these results regarding the test set, are better than those achieved from the AlexNet in the case of 100 epochs.
Found that the pretrained TransferNet in the case of 20 epochs gives good predictions, other tests on this model have been done, changing other parameters (the optimizer and the data augmentation of the training set), but without making significant improvements.