2022 Report for Homework 1 Knowledge Discovery Course - UNICT

10 Species Monkeys Classification

Berlotti Mariaelena *mariaelenaberlotti.mb@gmail.com*
Di Grande Sarah *digrandesarah@gmail.com*
Licciardello Cristina *cris.licciardello@gmail.com*

1 Dataset Description

The object of our report is the analysis of the 10 Species Monkeys dataset available on Kaggle: https://www.kaggle.com/datasets/slothkong/10-monkey-species
The dataset contains 1369 images referring to 10 different monkey species:

- **n0:** alouattapalliata;
- **n1:** erythrocebuspatas;
- **n2:** cacajaocalvus;
- **n3:** macacafuscata;
- **n4:** cebuellapygmea;
- **n5:** cebuscapucinus;
- **n6:** micoargentatus;
- **n7:** saimiriscIureus;
- **n8:** aotusnigriceps;
- **n9:** trachypithecusjohnii.

The problem we are dealing with is an image classification problem: image classification refers to the labelling of images into one of a number of predefined classes. In this case, our aim is to build a neural network that allows us to recognize a monkey image and to classify it into one of the ten categories. The number of images per class are:

- **n0:** 131 images
- **n1:** 139 images
- **n2:** 137 images
- **n3:** 152 images
- **n4:** 131 images
- **n5:** 141 images

- **n6:** 132 images
- **n7:** 142 images
- **n8:** 133 images
- **n9:** 132 images

The images we are analyzing have all different sizes; almost all of them are 400x300 pixels (or larger) and in JPG format. Since we are already referring to RGB images, there was not need to convert them. Since we have complex images, the data augmentation can help our model in improving the learning. So, We decided first to reduce the image size to 50x50 pixels each, with the aim of uniforming data shape and reducing memory requirements. We employed the images rotation of 20 degrees and finally, we normalized our data.
We divided the dataset into three parts:

- **training:** 969 images
- **validation:** 200 images
- **test:** 200 images

Figure 1: Dataset splits

2 Model Description

We created two different networks and for each one of them we started from deeper networks till we arrived to simpler ones.

2.1 First Network

The first network is composed by 4 convolutional layers, 1 fully connected and a classifier. The first convolutional layer takes as input an image of size 3x50x50, it applies a kernel of size 5 and a stride of 1. In this part, we did not apply the padding (padding = 0). This first layers gave us an output of shape 64x46x46, which is passed first, to a ReLU activation function, next to a Batch Normalization layer and finally, to a Max Pooling layer which reduces the size of the image to 64x23x23. The second convolutional layer is structured in the same way of the first one and it gave us an output shaped 128x21x21 that is reduced by the Max Pooling to an image of size 128x10x10. The third convolutional layer passes its output to a ReLU activation function that gave us an image of size 256x9x9. Finally, the fourth convolutional layer is structure in the same way of the third layer and produced an output image shaped 512x8x8. This last layer is connected to a fully connected layer that takes 32768 input features and gives us Finally, the final classifier takes those output features and returns back the classification of the monkeys in the ten species.

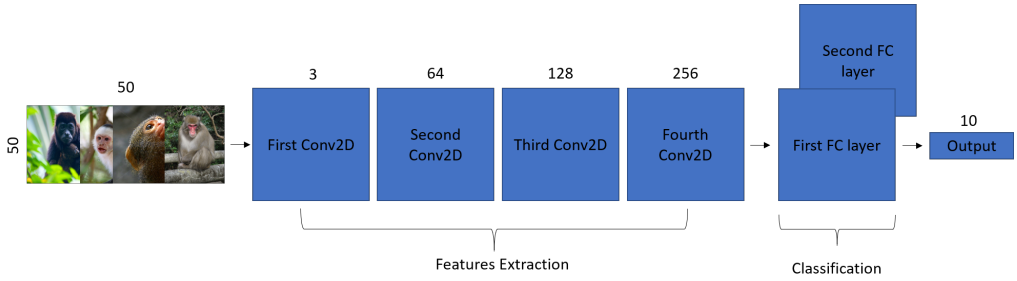

Figure 2: Network architecture

3 Training procedure

With a batch size of 10 and the Cross Entropy loss, we decided to apply the SGD optimizer with a learning rate of 0.01. At the beginning we run the code for 25 epochs, but then we decided to apply the early stopping in order to avoid overfitting and we stopped at epoch 13.

4 Performance analysis

In terms of performance by using this first CNN we obtained the best results and for this reason we decided to choose it as final model. In particular we got a train accuracy of 0.57, a validation accuracy of 0.51 and a test accuracy of 0.56, while the loss functions reach the following values: TrL=1.8981, VL=1.9412, TeL=1.9163.

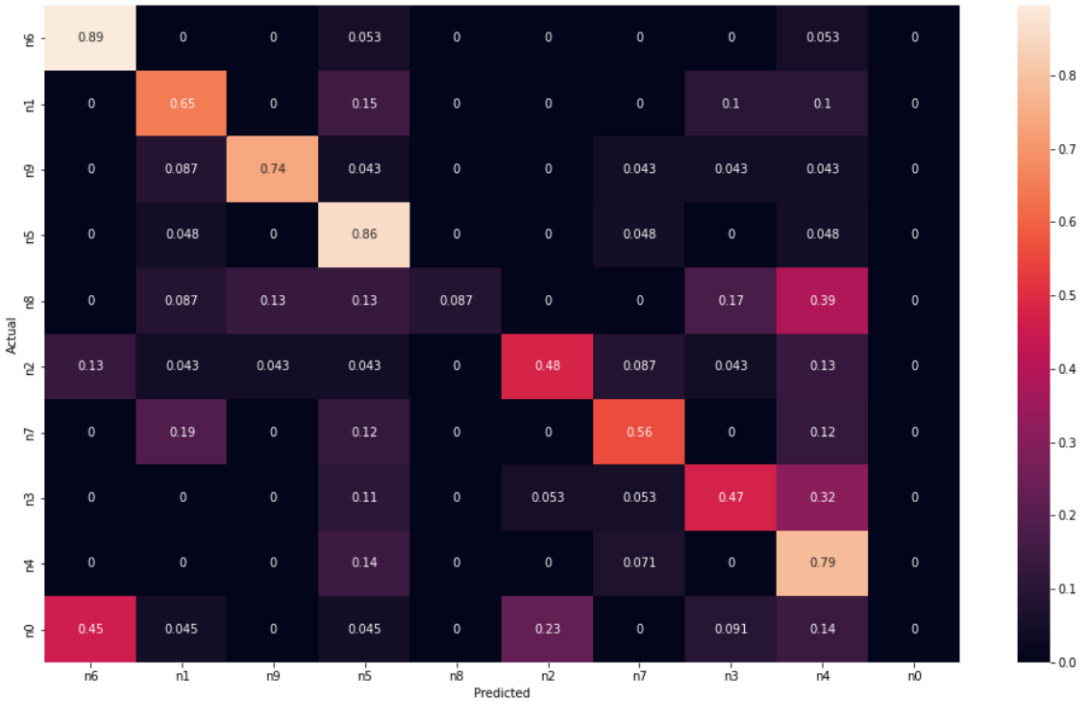

Figure 3: Network architecture

5 Training procedure

According to what is shown by the confusion matrix we can see that:

- For the category **n6** the model predicts in the correct way 0.89 of the true labels.
- For the category **n1** the model predicts in the correct way 0.65 of the true labels.

- For the category **n9** the model predicts in the correct way 0.74 of the true labels.
- For the category **n5** the model predicts in the correct way 0.86 of the true labels.
- For the category **n8** the model predicts in the correct way 0.09 of the true labels.
- For the category **n2** the model predicts in the correct way 0.48 of the true labels.
- For the category **n7** the model predicts in the correct way 0.56 of the true labels.
- For the category **n3** the model predicts in the correct way 0.47 of the true labels.
- For the category **n4** the model predicts in the correct way 0.79 of the true labels.
- For the category **n** the model does not predict any correct label.

6 Ablation studies

We tested the performances of this first network also in the following cases:

- **1st + 2nd + 3rd + 4th + 5th Conv layers + FC + Classifier:** in this case we obtained the following accuracy: TrA=0.48, VA=0.43, TeA=0.44.
- **1st + 2nd + 3rd Conv layers + FC + Classifier:** in this case we obtained the following accuracy: TrA=0.39, VA=0.42, TeA=0.41.
- **1st + 2nd Conv layers + FC + Classifier:** in this case we obtained the following accuracy: TrA=0.48 VA=0.49, TeA=0.54.
- **1st Conv layer + FC + Classifier:** in this case we obtained the following accuracy: TrA=0.49, VA=0.52, TeA=0.52.

6.1 Second Network

The second network is composed by 2 convolutional layers, two fully connected layers and a classifier. Also in this case we decided to apply the ReLU and the Max Pooling operation. By considering all the trials done for this CNN, in which we started always with a number of 5 convolutional layers and we finished with 1 convolutional layer, we obtained worst results with respect to the previous described network.

7 AlexNet

To deal with our problem we decided to implement also the AlexNet.

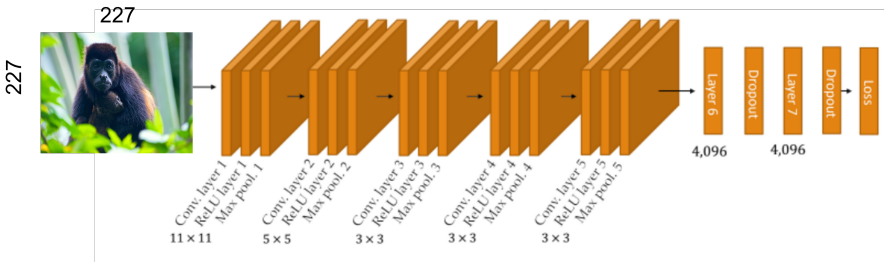

Figure 4: AlexNet architecture

The AlexNet is a CNN is made by 8 layers: we have 5 are convolutional layers and 3 fully connected layers. The first convolutional layer takes input image of size 227x227x3, it applies a kernel of size 11 and a stride of 4. The Padding value is fixed at to which means that we are adding two rows of padding to the top and bottom, and two columns of padding to the left and right of the input image. The Padding value we are using is zero (Zero-Padding). The AlexNet applies the Relu activation function followed by a MaxPooling layer which let us reduces the size of the image from 64x56x56 to 64x27x27. The second convolutional layer applies instead a kernel of size 5, a stride of 2 and a padding = 2. The output image is then passed to a Relu function and to a MaxPooling layer that reduces its size from 192x27x27 to 192x13x13. The third convolutional layer applies a kernel of size 3 and a stride of 1. This time the padding = 1 meaning that we are adding one row of zero-padding to the top and bottom, and one column of padding to the left and right of the input image. This convolutional layer

gives us an output shape 384x13x13. Also in this case, the layer is followed by a Relu activation function. Next convolutional layer applies the same values of before; it produces an output image of size 256x13x13. The fourth convolutional layer is followed by a ReLU activation function. Also for the final convolutional layer the values of the kernel, the stride and the padding are the same of before. The output image is passed first to a Relu activation function and then to a Max Pooling layer which reduces the size of the image to 256x6x6. The AlexNet applies the Drop out technique; in particular the value is fixed to 0.5 which means that a half of the neurons are disabled. As said before the network is made by 3 fully connected layers which combine the features extracted before. We trained the AlexNet on our data and we obtained worst results; indeed, the running of the AlexNet model for 10 epochs in our data, gave us a 0.07 test accuracy. For this reason, we decided to apply the fine tuning technique, as we will see in the next section.

7.1 Fine Tuning

The Fine Tuning is a technique consisting on the application of a pre-trained model on a specific dataset in order to have small changes in the parameters and to adapt them to the particular case. For our data we decided to take the weights of the AlexNet as pre-trained network and we obtained a good result in terms of test accuracy (0.89) even if we had over-fitting (training accuracy = 0.95).

7.2 AlexNet as features extractor

We decided to implement the AlexNet as features extractor. In this case, we freeze the pre-trained model and we don't change the parameters. This pre-trained model is used to extract features, then we process those features with a standard machine learning algorithm. Using this technique on our data, we obtained a 0.86 test accuracy in 10 epochs.

8 Conclusions

Once trained the two different networks, we decided to choose the first one because it returns the best performance: 0.56 of accuracy in 13 epochs. However, by using the pretrained AlexNet model for fine tuning and as feature

extractor, we have obtained best results. In particular, in the first case we obtained an accuracy of 0.89 (with overfitting) and in the second one an accuracy of 0.86, both in 10 epochs.