Deep Learning

Convolutional Neural Network for Image Classification

Academic year 2019/2020

Instructions

The objective of this exercise is to build a Convolutional Neural Network (CNN) for addressing a problem in the field of computer vision. More in detail, the problem is to correctly classify images belonging to two target classes: "cat" and "dog". The considered problem is a toy application, but the concepts you will use can be applied to solve more challenging and interesting tasks, such as distinguish the severity of breast cancer, identify the type of vessells on the sea surface (useful for maritime monitoring and rescue operations), etc.

The choice of this application is mostly due to the fact that collecting images of cats and dogs is a simple an inexpensive task, whereas in the medical field it is a time-consuming and expensive process. Moreover, CNNs produce better and better performance when a vast amount of images are provided in input. While data-augmentation comes in handy for increasing the size of the dataset, collecting images of only 20 or 30 patients is a challenging task.

Differently with respect to the previous exercise, the dataset that you will use does not come packaged with Keras. The dataset can be downloaded from the Moodle platform and consists of images of ten thousands cats and dogs. The two classes are perfectly balanced, thus you have five thousands images with cats and five thousands images with dogs.

The objective of the exercise is to build a CNN that is able to correctly classify an image in one of the two possible classes.

Before starting the exercise, it is important to understand how to store the available images. Considering that images are the only input provided, it is fundamental to find a "smart" way for representing the labels associated to the images. A Naıve approach would be to rename each image containing a dog with the name "dog_x" (where x is a number), and each image containing a cat with the name "cat_x". A more practical approach is to organize the folder with the images in the following way:

```
---Dataset
    ---Training
       ---Cat
       ---Dog
```

```
---Test
       ---Cat
       ---Dog
```

By using this organization, Keras will use the name of the folders ("Cat" and "Dog") as the label of a class. This is a traditional organization of the input data when you are working with CNN and images.

After this preliminary step, you are required to do the following:

- Build the CNN consisting of the following layer: a Convolutional layer followed by a MaxPooling layer; a second Convolutional layer followed by a MaxPooling layer; a Flattening layer; a Dense Layer (this is the first hidden layer for the fully connected part of the CNN); an output (Dense) layer. To summarize the network will have the structure displayed in Figure 1.

  The first Convolution layer will contain 32 neurons, the filter will have a size of 3×3 and the input_shape will be $64 \times 64 \times 3$; the first MaxPooling layer will be have a size of 2×2.

  The same parameters will be used for the second convolutional and MaxPooling layers. The Dense layer will have 128 neurons and in all the layer but not the output one, the activation function to be used is the Relu. For the output layer, the sigmoid function must be adopted (we are dealing with a binary classification problem).

- When calling the compile method, specify the following parameters: optimizer = 'adam', loss = 'binary_crossentropy', metrics = 'accuracy'.

- Use data augmentation to artificially increase the size of the dataset. To do that you should rely on the ImageDataGenerator discussed in the theoretical classes. In particular, try to apply some of the transformation available in Keras. It is important to rescale the images (both training and test sets), but the other transformations you will apply (like rotation, zoom, horizontal flip, etc.) must be applied **only to the training images**. Thus, you must create two different ImageDataGenerator objects: one that will be used on the training set, the second to be used on the test images. After this step, you must call the method flow_from_directory. In case of doubts you are invited to refer to the Keras documentation.

- Call the methods fit_generator and specify that 25 epochs will be executed. The parameter steps_per_epoch will be equal to 8000. This means that, in each epoch, the CNN will see 8000 images (that is the size of the training set). Considering the use of data augmentation, the 8000 images will be different at each epoch.

- Analyze the performance on the test data and compare it (in terms of accuracy) with respect to the training performance.

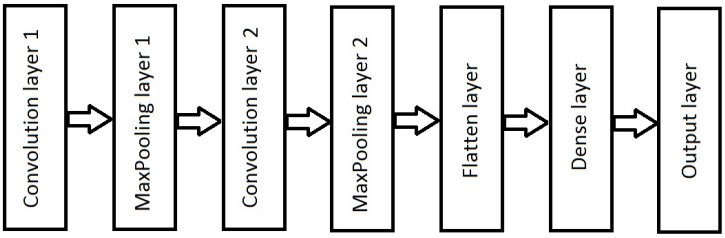

Figure 1: Architecture of the CNN