Cats and dogs images classification

Alessia Poli, 989824

May 2023

Abstract

The following analysis aims at developing Convolutional Neural Neutroks for the classification of Cats and Dogs images. The projects is structured as follows: I used Tensorflow to build 4 models, each having a different architecture, to inspect the impact of varying parameters in the networks. I, then, evaluated the performance of the model using k-fold cross validation.

1 Introduction

Convolutional Neural Networks (CNNs) allow to make explicit assumptions on image inputs, and to study the impact of different architectures and techinques on the efficiency and performances of a model. Model 1, 2, and 3 represent different level of depth of the Neural Networks. Model 4 introduces data augmentation as a regularization technique. in order to evaluate the learning algorithms implemented, I finally structured k-fold cross validation, with $k = 5$.

This report contsins the analysis of the models implemented, and of ther results as follows: in Section 2 I describe the preliminary operations performed on the dataset, to prepare it for the further analysis; Section 3 examines the theoretical framework behind the Convolutional Neural Network, which is necessary to understand the models implemented; Section 4 analyses the different models and compares the results obtained; finally, in Section 5, I study the methodology behind k-fold cross-validation and how it performs in the best performing of the previous models.

2 Data and Preprocessing

Figure 1: Examples of images in the dataset

The Dataset is very well known in the computer vision field. It contains 12500 images of Cats, and 12500 images of Dogs; so, its total dimensionality is of 25000. Each image is associated to a label, "Cat" or "Dog", depending on whether the image represent a cat or a dog. In Figure 1 are displayed some of the examples of the images in the dataset.

The dataset requires preprocessing in order to perform binary classification.

The steps involved are:

- Removing corrupted images: I deleted 1588 corrupted images, since they had size equal to 0,
- Formatting the original images from .Jpg to RGB,
- Scaling images to a standard size (180, 108),
- Images normalization: RGB images have been rescaled from [0, 255] to [0, 1].

Images after been processed should look like images in Figure 2.

Figure 2: Examples of images after preprocessing

3 CNN theoretical framework

Convolutional Neural Network (also called ConvNet) is a depp learning algorithm used to solve the typical problems of the computer vision field. Given images as inputs, the algorithm assigns importance to each object in the image, and it is capable of differentiating one from the other. The output of the learning algorithm are the probabilities of an image belonging to each class.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. Images are processed as a binary representation of visual data: they contain pixels arranged in a grid-like fashion having values corresponding to the intensity of the color of the pixel.

CNN architecture generally has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

3.1 Convolutional layer

The convolutional layer is the core building block of CNN. The Convlution Operation contains a set of kernel filters, which are parameters used to learn patterns throughout the training. The size of filters is smaller than the size of the actual images. So, if the images is composed of three dimensions, as in the RGB format, kernel's height and width will be spatially smaller, while depth extends to all three channels.

The convolution operation is defined by five parameters:

- Number of filters: adding multiple convolutional layers allows to identify higher level features.
- Size of filters: the dimension of each kernal.
- Padding: it is the number of pixels added to an image. It assists the kernel in the processing of an image, as it is added to the frame of the image to allow for more space for the kernel to cover the image. Padding allows higher accurcay.
- Stride: it determines how big the steps are for the convolutions when sliding the convolutional kernel. The higher the stride, the smaller the output.
- Activation function: it adds non-linearity to the CNN. It defines whether a neuron is activated or not.

3.2 Max Pooling layer

Pooling summarizes the presence of features in patches of the feature map. In particular, Max Pooling summarizes by considering the most activated presence of a feature.

The pooling layer downsamples: it creates a lower resolution version of the original inputs, that still contains the large or important structural elements, without the details that would not be useful in the task.

3.3 Fully Connected layer

The Fully Connected layer is the final layer of a CNN. The Flatten layer performs the task of classification based on the features extracted through the previous layers and their different filters. The Flatten layer can also pass through Dense layers, learning complex relationships between the Convolutional and Max Pooling layers.

It is defined by the following parameters:

- Number of node,
- Activation function.

4 Model architectures and results

The mdoelss which are objet of the following analysis, have the following hyperparameters:

- Number of filters: the number of filters varies among the models implemented. This parameter distinguishes each model I built.
- Size of filters: it was fixed at 3×3.
- Padding: I left the dimensions of the output to be the same as the input's, so that they have the same width and height.
- Stride: I used value of 1. This value results in the maximum overlap of receptive fields.
- Activation function: I implemented the ReLu function: $f(x) = max(0, x)$. ReLu returns 0 if the inputit receives is negative, and returns the input itself if it is positive.

4.1 Model 1-3

The first model I implemented has the following architecture: it has three blocks of Convolutional and Max Pooling layers, with 32, 64, and 128 filters computed.

The second model has exactly the same structure, but it also adds one Dense layer with 128 filters. And the third model also adds a convolutional layer with 256 filters.

Examining the results of these three models, in conjunction, allows to show whether a deeper and more complex architecture performs better o worse when classifying images. In fact, augmenting the number of layers should allow the algorithm to learn more features of the images.

The measures implemented to evaluate how the algorithms perform are Accuracy and Loss. Accuracy represents the number of correct predictions over the total number of predictions. In binary classification it may also be measured as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP = true positive, TN = true negative, FP = false positive, and FN = false negative. An immediate representation of such measure is given by the confusion matrix. In Figure 3, I compare the confusion matrices of the three

models, computed on a sample from the test of the predictions for each one of them. All the three models seem to perform well in capturing the right predictions, as true positives' and true negatives' number dominates false positives' and false negatives'.

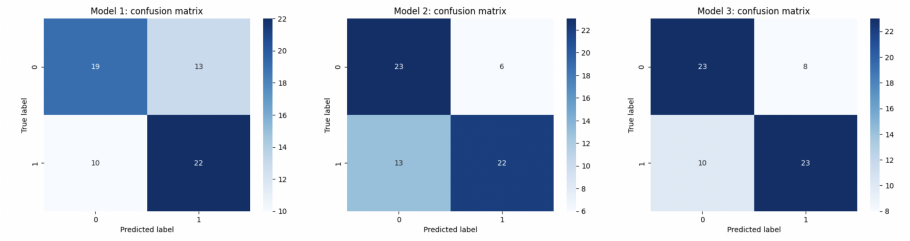

Figure 3: Confusion matrices of Model 1, Model 2, and Model 3

But performances should also be evaluated in terms of loss. Given that the problem is a binary classification, the loss function used is the binary cross-entropy. It measures the dissimilarity between the predicted probability distribution and the true binary labels of the dataset. Binary cross-entropy is the negative average of the log of corrected predicted probabilities:

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) \tag{2}$$

where p_i is the probability of the label associated to the prediction is 1, and $(1 - p_i)$ the probability of the label being 0.

We distinguish between three loss measures: training loss, validation loss, and test loss. Train and Validation metrics are used in the implementation of the model. Accuracy and Loss for train and test are used to evaluate model performances. Loss and accuracy values over the epochs of each of the implemented model are dispalyed respectively in Figure 4, Figure 5, and Figure 6.

As we can see from accuracy and loss values for the three structures (Table 1), the model seems to be improving, given that its architecture becomes more complex by adding layers.

	Accuracy	Loss
Model 1	0.711	1.244
Model 2	0.752	1.107
Model 3	0.754	0.761

Table 1: Model 1, Model 2, Model 3: accuracy and loss

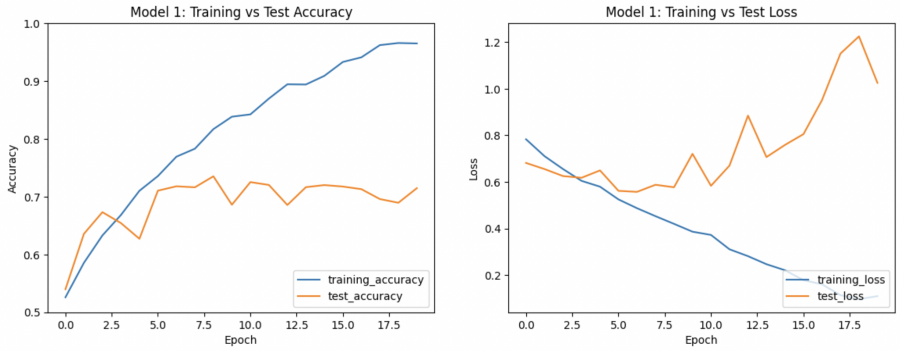

Figure 4: Model 1: accuracy and Loss

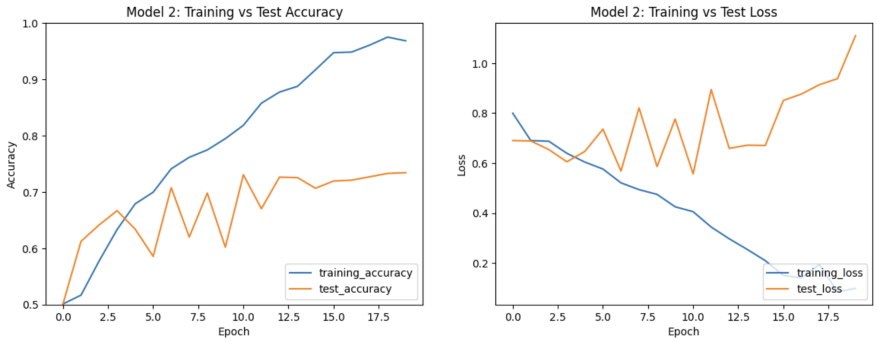

Figure 5: Model 2: accuracy and Loss

Figure 6: Model 3: accuracy and Loss

Accuracy and loss improve across the three model, so adding filters is useful for the algorithm to learn patterns among the images. But it does not seem to be enough to solve the overfitting issue. In fact test loss increases throughout the epochs, while train loss goes down. A reason behind overfitting maybe due to the fact that the model was trained for a too long period. It could be helpful to introduce early stopping.

4.2 Model 3 with early stopping

Since overfitting may be caused by a too long learning phase, I introduce Early Stopping in fitting the Model 3. This allows to stop the implementation when validation accuracy does not improve in more than 3 epoch consecutively.

	Accuracy	Loss
Model 3	0.847	0.390

Table 2: Model 4: accuracy and loss

Table 2, shows that here is significant improvement in both loss and accuracy. Moreover, introducing early stopping drastically reduces overfitting, as train and test loss converge, decreasing throughout the epochs, as we can see from Figure 7.

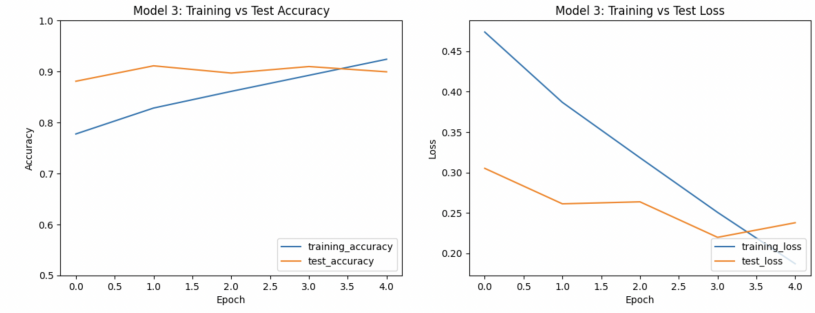

Figure 7: Model 3 with early stopping: accuracy and loss

4.3 Model 4

An alternative source of overfitting may be due to too high variance of the model. So, it can also be addressed introducing data augmentation. Data Augmentation introduces random noise in the data, hence reducing variance.

Model 4 is implemented on the structure of Model 3 with early stopping, and Data Augmentation. To augment data, I performed some geometric transformations:

- Random zoom of 15%,
- Random width shift of 10%,
- Random height shift of 10%,
- Random horizontal flip.

In the model with augmented data the values for accuracy and loss are displayed in Table 3.

	Accuracy	Loss
Model 4	0.851	0.336

Table 3: Model 4: accuracy and loss

Test loss and accuracy both have a slight improvement with respect to Model 3. But, from Figure 8, test loss seems to increase after few epochs, which may lead to overfitting, even though extremely reduced with respect to Model 1, 2, and 3.

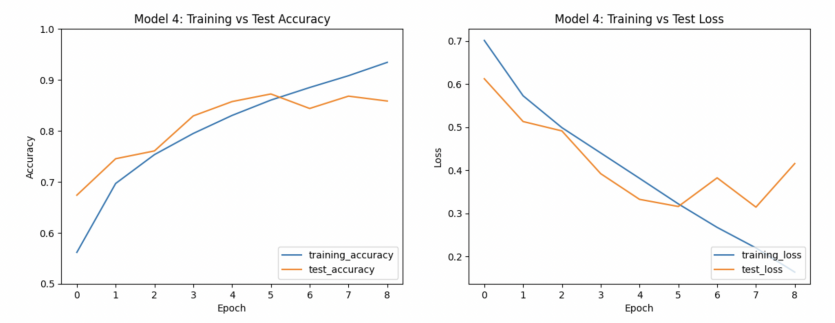

Figure 8: Model 4: accuracy and loss

Overall, data augmentation is capable of reducing overfitting.

5 Cross validation

Cross Validation allows to evaluate Machine Learning model performances based on error metrics. K-fold Cross Validation divides the dataset in folds, and each fold is used once as a testing set. In the speicifc case, I implemented 5-folds cross validation: in the first iteration, the first fold is used as the test set and the remaining ones to train the model; in the second iteration, the second fold is used as test and the other as train; and so on.

Cross validation was computed on Model 3, as it has the best performances in terms of accuracy and loss, and in terms of accuracy. In this implementation I reduced the number of epochs, for computational reasons, but I had the chance to verify that the results I obtained were also valid for higher number of epochs and without early stopping.

	Accuracy	Loss
Model 4	0.847	0.362

Table 4: CV Model: accuracy and loss

Figure 9: CV Model: accuracy and loss

Cross validation is able to maximize accuracy and minimize error (Table 4). Overfitting disappears too. In my analysis, cross validation yields to the best results, in reducing overfitting, and, consequently, at allowing for generalization of the model.

6 Conclusion

The analysis yields to the conclusion that, generally, a more complex model is better able at capturing features and patterns of images. Although, overfitting is recurrent issue of this models, due to high variance, or too long implementation phases, etc. It can always be addressed with techniques, such as early stopping, data augmentation, and cross-validation. And not having overfitting is necessary to allow the model to be better at generalization.