Università degli Studi di Milano

MSc. in Data Science and Economics

Statistical Methods for Machine Learning

Binary classification of cats and dogs images

Angela Roberti
angela.roberti@studenti.unimi.it

May 19, 2023

Introduction

This project concerns the implementation of a Convolutional Neural Network (CNN) for cats and dogs image classification. In particular, after creating a baseline model I've explored the performance of different CNN applying changes in the hyperparameters as well as the impact of regularization methods. Finally, I've computed a more accurate risk estimate of my best model with a 5-fold cross validation.

1 Dataset

The dataset used for this project initially contained 25000 images total, 12500 images labelled as cat images and 12500 images as dog images. In order to improve the performance of the models built during the project, the dataset has been cleaned by removing the images that were found as corrupted. After the cleaning, 23410 images were selected as non-corrupted, 11741 for cats images and 11669 as dog images. The number of images (only 6.36% of images were removed) made it possible to still use them to build a model that will result in high accuracy.

Besides, all the images were then converted from JPEG to RGB format and rescaled in a standard size, having the shape of (128x128x3).

2 Theoretical framework : why CNN?

Neural Networks are a useful tool to solve different types of problems related to images one of those is image classification, as for this project. A Neural Network is able to recognize and analyse images by learning and identifying complex patterns and features. Neural Networks that have at least one hidden layer which is not either the input nor the output layer are called deep neural networks. In particular Convolutional Neural Network (CNN) is a class of deep NN and is the most diffused type of neural network used when the goal is to assign a label or a category to a large amount of data.

A CNN consists of 3 main layers:

- **CONVOLUTIONAL LAYER**: it is the core of the structure; it applies a series of filters (for this project with fixed size 3x3) to the image each capturing a specific pattern or a feature. The filter, or kernel, moves across the input image checking if a feature is present in it, as the iterations goes the kernel sweeps the entire image. Convolution operations are nothing but element-wise matrix multiplication between the filter values and the pixels in the image and the resultant values are summed.
- **POOLING LAYER**: as the name suggests conducts the mathematical operation of pooling so the extraction from a given matrix of a new

matrix with lower dimensionality and which contains only one element for each cluster of the original matrix. They are used to downsample the image helping in reducing the numbers of parameters and consequently the computation required.

- **FULLY CONNECTED LAYER (FC)**: or Dense layer, is the last layer, before the output layer, where image classification happens according to the features extracted in the previous layers. It's called fully connected because here all the nodes from one layer are connected to every activation unit, or node, of the next layer.

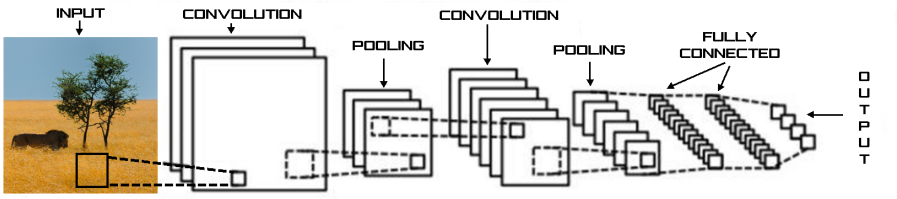


Figure 1: Example of CNN structure

The **flatten layer**, that comes before the FC, simply convert the output of the previous layer, which is multi-dimensional, into a one-dimensional vector. It does not introduce any additional parameters or computations, and its purpose is solely to make the input compatible with the subsequent fully connected layer.

The **activation functions** used in this projects are the *Rectified Linear Unit (ReLU)* and the *Sigmoid function*. The latter is typically used for binary classification, and it has been used in this case only for the output layer which is composed of 1 node, it is the layer that outputs the probability that the input is a cat of dog's image. The Rectified Linear Unit, used in the other occurrences, is a non-negative activation function. It has over the Sigmoid function the advantage that it is simple to calculate as it involves only the comparison between its input and the value 0, and it has a derivative which is either 0 or 1 depending on the input given. Besides, ReLU does not saturates close to its extreme producing very small gradients that can cause the signal to be slow to propagate especially for larger networks, as the Sigmoid does instead. This helps the backpropagation of the error during training.

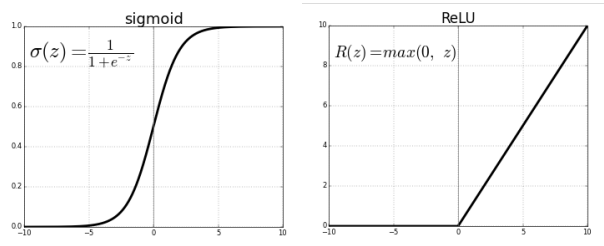
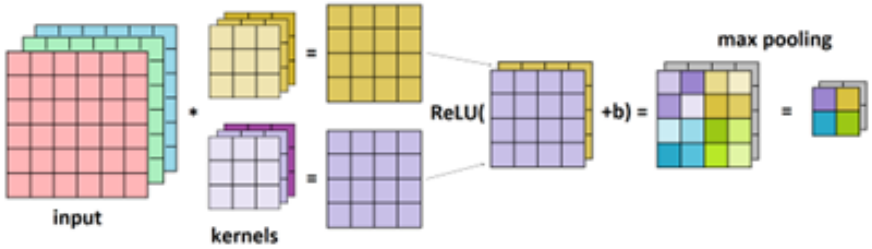


Figure 2: Sigmoid and ReLu functions

For the **pooling layer** It's used *Max Pooling*, so the final matrix will be made by only the greatest values among those in the original matrix. This helps to decrease the dimensionality of the input data to make if more manageable at the cost of some negligible loss of information content.



3 Model compile

The **learning rate** is a hyperparameter that controls the amount of change of the weights during training in response to the estimated error. Usually deep learning neural networks are trained using the stochastic gradient descent optimization algorithm.

For this project I chose to use the *Adaptive learning rate optimization algorithm (Adam)* that combines the advantages of two other optimization methods: AdaGrad and RMSprop. Adam computes individual learning rates for different parameters based on their historical gradients. Step size of Adam update rule is invariant to the magnitude of the gradient.

For the **loss function**, since this is a project related to binary classification, I've selected the *Binary Cross Entropy loss function*, where the function compares each predicted probability with the actual class of output, that can be either 0 or 1.

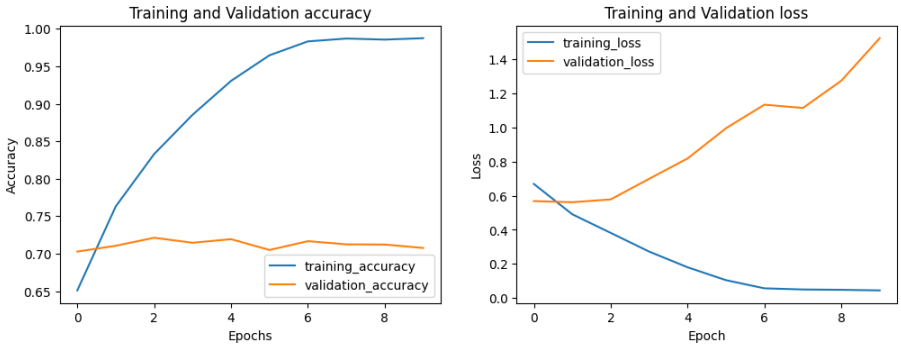
Finally, the **metric** chosen is *accuracy*. Accuracy metric computes the accuracy rate so the number of correctly predicted images as a fraction of the total number of predicted images.

4 Model 1 : Baseline model

I've used $Model1$ as my baseline model, from which I've started to investigate the structure of my convolutional neural network. Its architecture it's simple as it requires only:

- Convolutional layer, with filter size fixed 3x3, followed by
- Activation layer (ReLU),
- Pooling layer (MaxPool)
- Flatten layer
- Fully connected layer (64)
- Activation layer (ReLU),
- Fully connected layer (1)
- Activation layer (Sigm)

Figure shows the accuracy and loss curves over 10 epochs. After the 6th epoch there is a sign of plateau of the training accuracy that reached over 95% accuracy indeed the validation accuracy doesn't seem to perform well, stucking around 70%. Regarding the loss graph, the validation loss is highly increasing. The final test accuracy is 0.7142.



5 Models 2: Second convolutional layer

In CNN setting the input image goes through a set of filters, each of these activates certain features from the image, does its work and afterwards it passes the output to the filter in the next layer. Each layer of the CNN learns to identify different features/characteristics and this operation is repeated for each layer. In the first layer the kernels start as simple features, they then increase in complexity to check and identify more detailed features that uniquely represent the input object.

Consequentially, the next step to improve the performance of $Model1$ is to increase the complexity of the model, this will allow the CNN to identify larger portions and more complex features of the images.

For $Model2$ a second convolutional layer it's added with number of filters 32 and its performance it's compared with $Model2.2$ where instead the second convolutional layer has double filter number, 64.

After training both models for 10 epochs, the test accuracy were respectively 0.7644 and 0.7824 for $Model2$ and $Model2.2$. So the higher number of parameters allowed to have a more precise accuracy overall.

	Model 2	Model 2.2
filter size	32,32	32,64
n of parameters	1853473	3705921
test accuracy	0.7644	0.7824

In both cases, again after the 6th epoch the training accuracy reached its peak. The validation accuracy increased compared to model 1 but still remains stuck at a too low level of accuracy. This and the large gap between the training and validation accuracy are evident signs of overfitting that will be solved later.

Models 3 : Adding 3rd convolutional layer

I've further investigated the architecture of the models by adding another layer to the previous models with kernel seize 64 and 128 for the third convolutional layer. This choice was inspired by the VGG structure where the number of filters increases as we go deeper into the network, starting from 64 and doubling after each pooling layer, the convolutional layers have a fixed filter size of 3x3 and ReLU activation is used after each convolutional layer.

After 25 epochs, the test accuracy was a little bit smaller for $Model3.2$ (0.7886) compared to $Model3$ (0.7950) . Looking at validation accuracy

curves it's possible to notice that $Model3.2$, even if it has a slightly lower test accuracy, has overall a higher trend in validation accuracy.

	Model 3	Model 3.2
filter size	32,64,64	32,64,128
n of parameters	859265	1699009
test accuracy	0.7950	0.7886

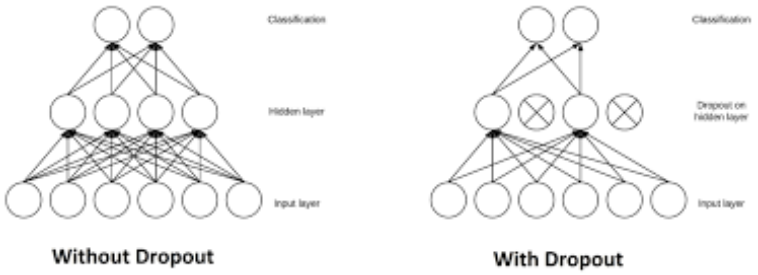
(a) Model 3

(b) Model 3.2

6 Overfitting problem

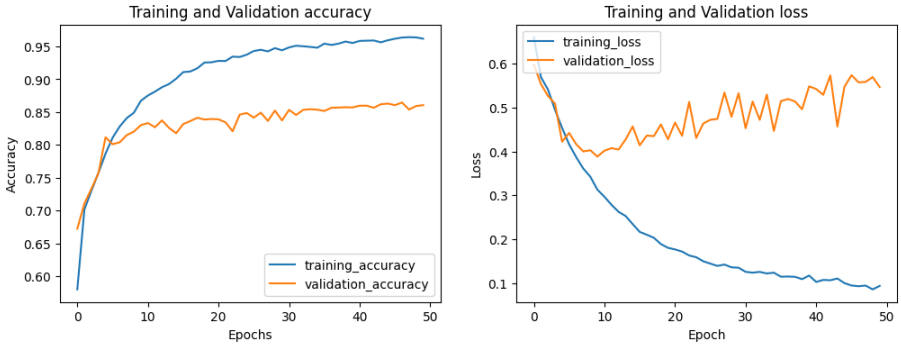
As the accuracy graphs clearly shows, for all the previous models, there is an evident problem of overfitting. The CNN is trying to learn too many details in the training data, and at this point it is also learning the noise in the data which has a direct effect on its performance on the test data set. The amount of prior knowledge is obstructing rather than supporting the acquisition of future information on that same subject.

To overcome this problem, I've implemented the **dropout function**. The dropout layer is a mask that nullifies the contribution of some nodes towards the next layer and leaves unmodified all others. By randomly deactivating some nodes in the layer, it forces the network to learn with less information and thus prevent overfitting on the training data. When the dropout layer is not present, the first batch of training samples influences the learning in a disproportionally high manner.

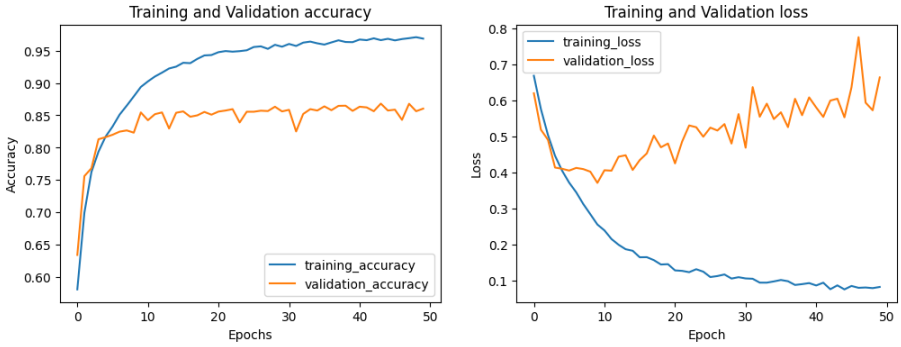


7 Models 4: Models 3 w/ dropout function

To test how far the models can be improved, I've implemented the function of dropout both on the convolutional layers and in the fully connected layers. The models has been tested on 0.2 dropout rate for convolutional layer and for the fully connected layer I let the dropout rate be higher, (0.3,0.5) since it possesses a higher number of trainable parameters. The results show that the models with dropout rate of 0.5 in FC give higher test accuracy (0.8605, 0.8642)



Model 4 with dropout



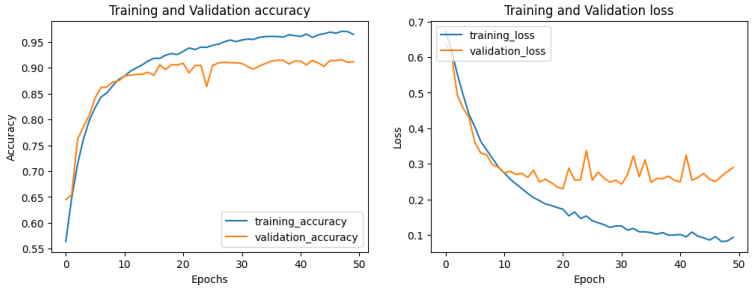
Model 4.2 with dropout

Compared to the models without the dropout function, as the training progressed, there is an observable reduction in the gap between the training and validation accuracy. This reduction signifies that the model's ability to generalize improved, indicating a partial resolution of the overfitting issue. However the validation loss has still a increasing trend and the gap between the accuracy curves can be further improved.

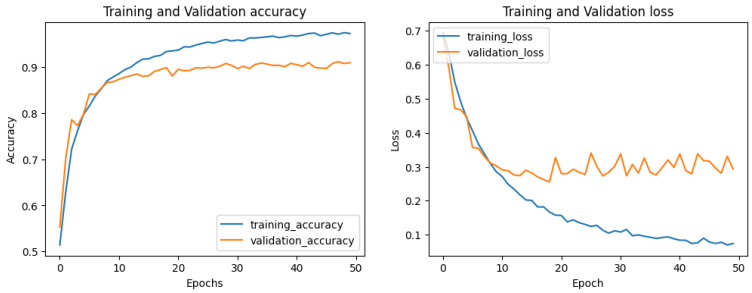
8 Models 5 and 6 : 4^{th}convolutional layer

At this point to adjust the models complexity another convolutional layer it's added from $Model4.2$, maintaining the same dropout rate; results are shown in the table below.

	Model 5	Model 6
filter size	32,64,128,128	32,64,128,256
n of parameters	535873	978369
dropout rate	0.2,0.2,0.2,0.2,0.5	0.2,0.2,0.2,0.2,0.5
test accuracy	0.9124	0.9165



Model 5



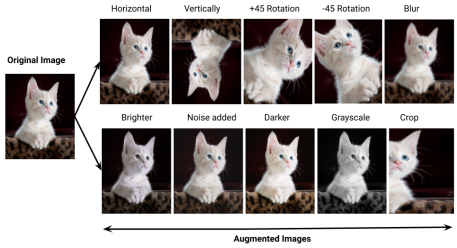
Model 6

Here it is worth noting that the validation accuracy reached a relatively stable point after the overfitting was addressed. This stability implies that the model's performance on unseen data became more reliable and consistent, which is a positive outcome.

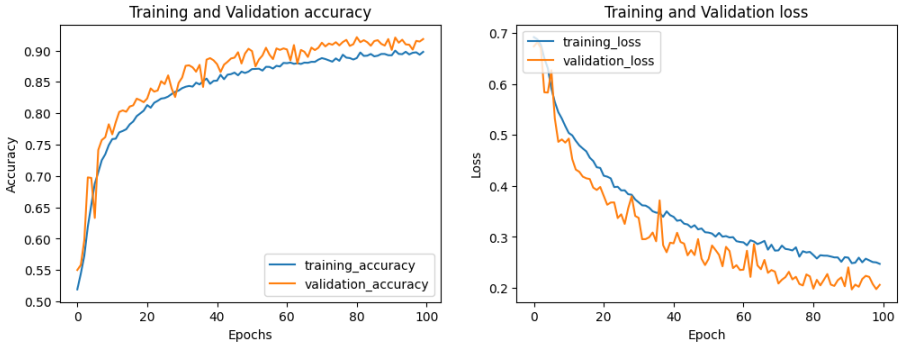
To further study the improvements these 2 last models, by trying to reduce even more the problem of overfitting, I've implemented the function of Data augmentation.

9 Data Augmentation

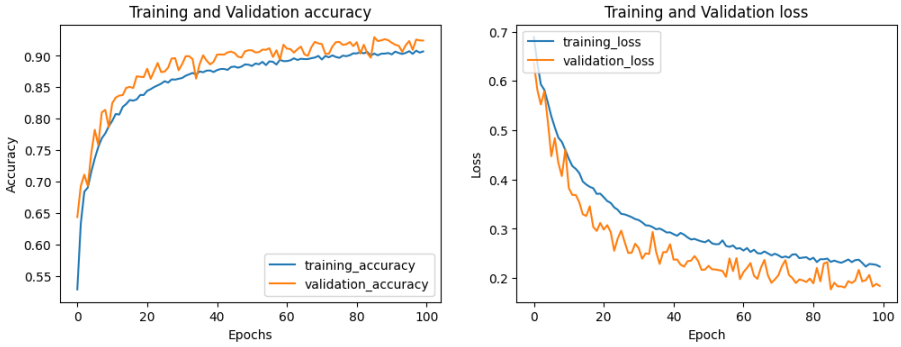
Data augmentation helps the model since it forces the CNN to identify the images when these are being slightly modified. In particular in my example, I've implemented **random rotation** and **random flip**. This is supposed to help the neural network model as it expands the training dataset and so the ability of the models to recognize what they have learned to the new images, imposing to the model to not just memorize the trends.



For this reason, the model has been trained over a higher number of epochs (100). In particular the models with data augmentation improved thier test accuracy and after 100 epochs the test accuracy is 0.9329 for model with progressively increasing size of filters ($Model8$) and 0.9261 for $Model7$.



Model 7



Model 8

Overall, the accuracy curve graph demonstrates that the inclusion of data augmentation has been successful in resolving the overfitting problem and improving the model's generalization ability. The accuracy curves do not seem to have reached a plateau still after 100 epoch, meaning that the models could improve even further, but due to limitations of access to GPU, I had to stop the research after the 100th epoch.

10 K-fold Cross Validation

*Model*8 is the most performing model so I applied k-fold cross validation to it. Due to the impossibility of accessing faster GPU the k-fold cross validation is estimated only over 50 epochs per fold.

The accuracy of each fold :

- 0.9158479571342468
- 0.8902178406715393,
- 0.9058094620704651,
- 0.9117898344993591,
- 0.8906450271606445]

The average accuracy is 90.28%. The k-fold average accuracy is lower than the test accuracy obtained before (93.29%). Cross-validation provides a more robust estimate of a model's performance compared to a single train-test split. By partitioning the data into 5 subsets, it allows for multiple evaluations of the model on different combinations of training and test sets.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.