Cats or Dogs?
CNN for Image Classification[*]

Matias Luraschi
matiassantiago.luraschi@studenti.unimi.it

May 26, 2023

Abstract

This empirical project has the scope of showing one of the usages of Convolutional Neural Networks (CNN) to do image classification. A dataset filled with images of Cats and Dogs is used in order to create a model that will allow us to do binary classification. We start with a baseline model and we start adding complexity. The models are compared between their accuracy. Once the final model is chosen, 5 Fold Cross Validation is performed.

1 Introduction

The scope of this report is to show one of the uses of CNN for image classification. A dataset containing 25000 pictures of cats and dogs has been used to train a model that will have the scope of telling if a given picture belongs to either one category. That said, this report will be organized as follows: this present section will be a mere introduction that will provide the preliminary organization of the whole report. Further on, in the second section, a very brief summary of the Theoretical Framework of the Convolutional Neural Networks is explained. The third section contains both a description of the content of the dataset used as well as some comments on its cleaning process and the data augmentation techniques that were applied to the images. On the fourth section the different models are shown. However, it needs to be understood, that several models were tried in order to improve performance. That said, it will be counterproductive to state and show them all. Therefore, only the models that have produced significant changes will be shown. Later on, on the fifth section the results of the 5 Fold Cross Validation that is performed on the final model is shown. Final comments will be found on the sixth and final section. Lastly references for some theoretical articles that have been used to decide a range of values for the hyper parameters will be shown on footnotes along the report.

[*]**Disclaimer:** 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.

2 Theorical Framework[1]

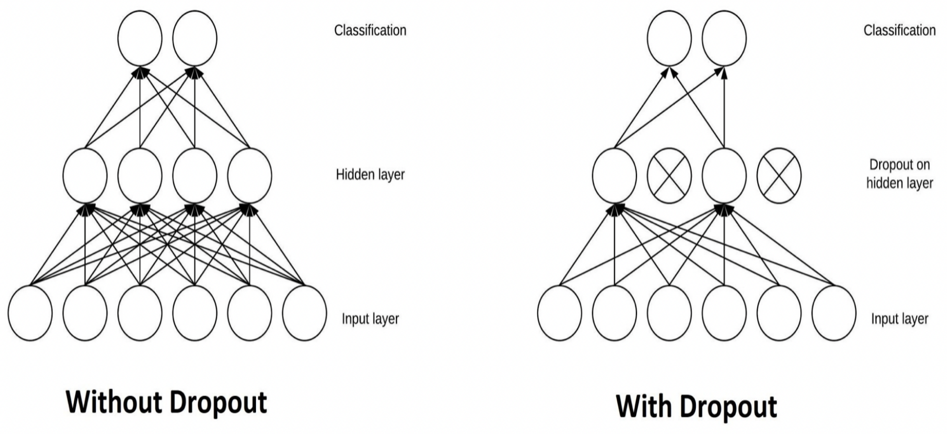
One of the uses of Convolutional Neural Network (CNN) is analyzing visual data. In a nutshell, a CNN basically consists of an input layer, hidden layers and an output layer. In a convolutional neural network, the hidden layers include one or more layers that perform convolutions. The convolutional layers convolve the import and pass its results to the next layer. Intuitively, the process feeds the pixels of the image in the form of arrays to the input layer. Then the hidden layers carry out feature extraction by performing different calculations and manipulations. Finally, there's a fully connected layer that identifies the object in the image. To explain it very easily, a CNN can recognize distinctive features or patterns from any position of an image, naturally, in our case, would be features of cats and dogs.

Layers in a Convolutional Neural Network:

- Convolutional Layer: is the core building block of the CNN. The Convolution Layer is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation, these filters help to extract specific patterns from parts of the image. Naturally, adding more filters will help us to recognize more patterns within an image, hence improving accuracy, but would have as a counter effect the increase in the amount of trainable parameters that at the end of the day could cause over fitting.
  Every image then is considered as a matrix of pixels. This matrix, then, will be used as the feature detector. The Convolution Layer performs a dot product between two matrices, one of them is the set of trainable parameters and the other will be a restricted portion of the receptive field.

- Rectified Linear Unit Layer (ReLU). ReLU performs an element-wise operation and sets all negative pixeLs to zero. It introduces non-linearity to the network and the generated output is a rectified feature map.

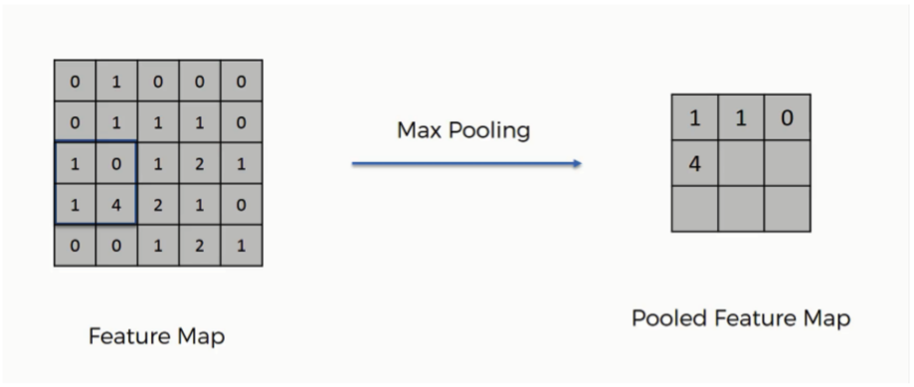
- Dropout Layer. This layer basically ”turns off” a proportion of the neurons and nullifies the contribution of them towards the next layer. This layer normally prevents over fitting. Its behaviour can be clearly understood in the image below.



- Pooling Layer perform a down sampling operation that reduces the dimensionality of the feature map. This means, that it replaces the output of the network by deriving a summary statistic

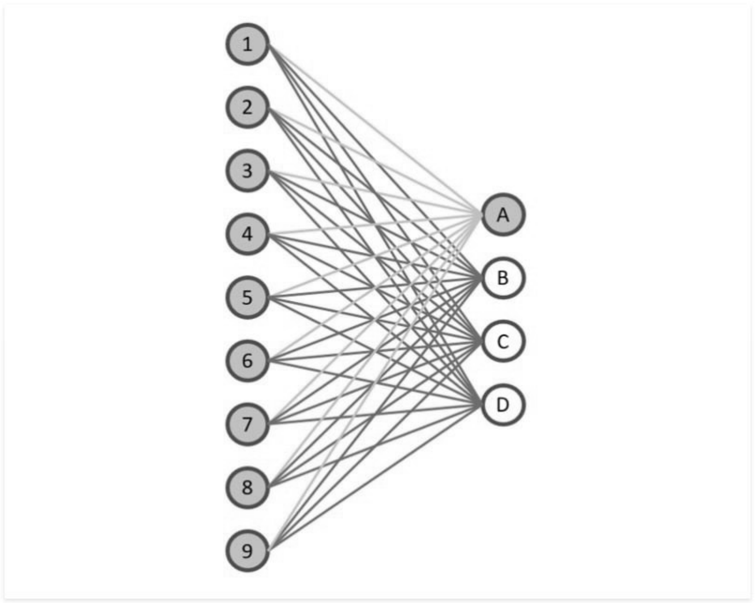
[1]Alzubaidi L. et all. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Available here.

of the nearby outputs. This helps reducing the spatial size of the representation which easens calculations. The pooling layer uses varius filters like average, L2 norm, weighted average and maxpooling. The latter is the most popular, because it extracts the maximum output from the specified grid. A very clear representation of what the Max Pooling Layer does is seen in the image below.



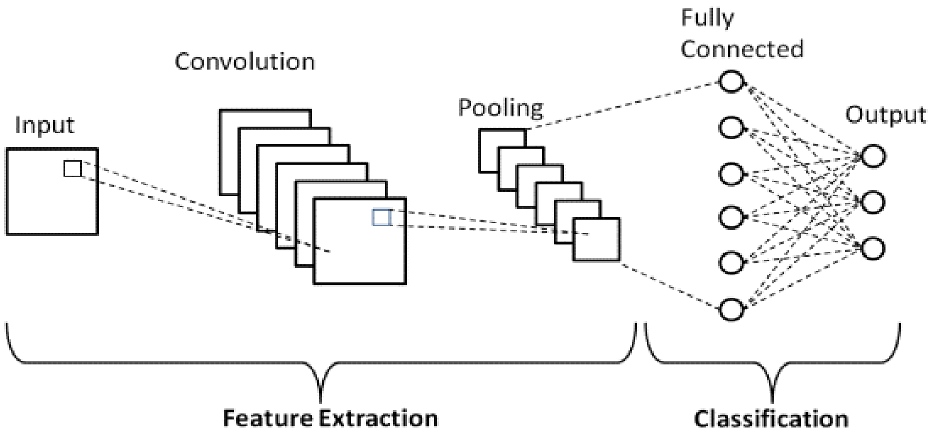
It can easily seen that the MaxPooling filter takes the higher value from each sub-grid and uses it to crreate a new grid that is a smaller version of the original one. That way, this layer helps with reducing the number of parameters, and, at the end of the day, to avoid over fitting.

- Fully Connected Layer: Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layers. It helps to map the representation between the input and the output. In a nutshell, the fully connected layer concludes the CNN. It aggregates and weights all information and generates the final classification.



Then, we see that every single one of the nodes A, B, C and D are connected to all the previous nodes.

Then, the whole process looks as illustrated in the picture below. Naturally, in our case, the output corresponds to a binary classification model.



We've just explaind, very briefly, how is it that a CNN works and what is that some of its layers do. Anyways, in order to improve the performance of our CNN, normally a Data Augmentation process is performed. Data augmentation is used to produce robust results. The process is basically modifying visually the original input image by doing process of rotating to different angles, flipping images along the different axes, resizing, translating, adding noise, etc.

- Flips: When performing a flip, the algorithm learns on having the image taken from "different angles". By using flipping, mirror images are created that help improving the learning process of the algorithm.

- Rotation: is normally performed at 90 degrees (to change the base of the image).

- Translation: This data augmentation technique shifts the main object of the picture in different directions.
- Scaling: This helps to give more diversity to the training data. It is used so that the CNN can recognize pictures that are taken more closely or more far away from the subject. That meaning, that sometimes even if within the picture an ear, or a leg of a dog is not present, the CNN will be more likely to be able to classify properly the image.
- Noise addition: This helps the CNN to learn with less perfect images. The noise are basically black and white dots. Then, when the algorithm is feed with real images that are not taken within the best photographic conditions, it will be more likely that it will classify it properly.

3 Data Processing

The dataset used for this project contains 12500 images of cats and 12500 images of dogs. Now, given that the dataset is balanced, we could say that if we were to use this one the results we would get from a binary classification would be quite accurate given that the dataset is balanced within the two categories of our binary problem.

The first 9 pictures for cats and dogs respectively are shown before just to give us a hint of the kind of pictures will be working with.

However, we have to take into consideration that the given dataset has to be cleaned of corrupted files and would avoid our algorithm and the libraries used to run properly. Then, after deleting 1590 images we get a final dataset of 23410 images. Given that we still have a fairly balanced dataset we assume is ok to make use of it.

Further on, as explained in the previous section, it is possible to perform some adjustments to the pictures in order for our algorithm to perform better. Then, within the Data Augmentation process, we decided to stablish two main characteristics.

- Horizontal Flip. As we explained before horizontal flip data augmentation is a technique used to increase the size of a dataset by flipping images horizontally. This can help improve the accuracy of a model by exposing it to additional variations of the same image. The mirroring effect can change the perception of the algorithm and help it to improve.
- Rotation Range at 30 degrees following the work done by Nakamura et all[2].
- Rescaling 1/255 in order to help us compute more efficiently and more rapidly our models.

4 Models

This section will be showing the results we got from running different models in terms of how accurate our models were. They will be shown from least accurate to the most accurate and also by the level of complexity of the CNN. Once again, it is important to highlight that here only a few models are shown. In order to get these models, several others models were tried.

4.1 First Model

For the very first model it was decided to use a baseline and simple model. That said, let's remember that Data Augmentation, as stated in the previos section has been applied and will be present within all the models that are going to be run.

Some of the characteristics chosen for the first model are:

- Filters chosen are 16. The higher the filters the more features can be extracted from the images.
- Kernel of 3x3. Normally, smaller kernel sizes are preferred to larger ones because we need a filter that is comprehended by an uneven number and at the same time we need to capture information from the neighbouring pixels. That said, 1x1 is not considered because it would have no information about the neighbouring pixels, on the other hand 5x5 might be too big. Then, 3x3 is chosen.
- The pictures are load in a shape of 128x128 pixels to reduce their size, and make computation faster and above all feasible. Also, homogeneity is preferred and 128x128 is a standard size.
- A dropout parameter of 0.2 is chosen for the Flatten layer.

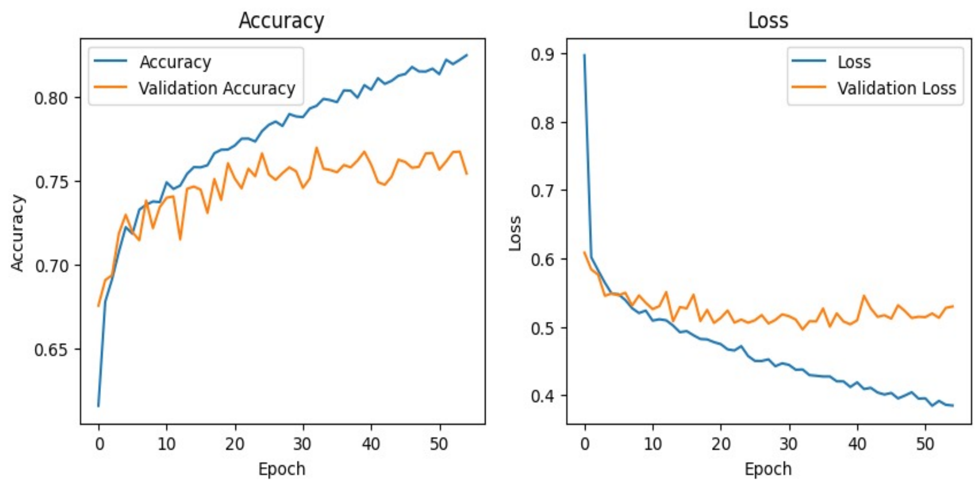
  When getting the summary of the model it can see that we have more than 16 million trainable parameters. Which obviously is a lot.

[2]Nakamura N. et all. Efficient Parameters for Rotation Processing of Data Augmentation. Available here.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 activation (Activation)     (None, 126, 126, 16)      0

 flatten (Flatten)           (None, 254016)            0

 dense (Dense)               (None, 64)                16257088

 activation_1 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

 activation_2 (Activation)   (None, 1)                 0

=================================================================
Total params: 16,257,601
Trainable params: 16,257,601
Non-trainable params: 0
_________________________________________________________________
```

Then, it is decided to train the model for 55 epochs. The longer the epochs the better. However, naturally, the higher the amount of epochs, the longer the training time of the model. The right amount of epochs would depend on the problem and on the complexity of the model. The optimum number of epochs would also be reached when getting a stable level on the accuracy curves, hence, when after an iteration the values are not changing significantly. However, due to computational and time constrains it is decided to train the model with 55 epochs.
It can be seen however that at the very beginning, meaning, up to 10 epochs, the Accuracy and the Validation Accuracy are similar. Nevertheless, after the aforementioned threshold it begins to divert. Moreover, by the end of the iterations the gap betwen them is quite high. This latter effect is a sign of over fitting. It is a problem that has to be addressed.

These results can be also seen on the two graphs that are shown below. On the left hand side of the subplot the Accuracy can be appreciated and on the right hand side the Loss can be seen.



By the 55th epoch the following results are obtained:

Model 1			
Accuracy	**Validation Accuracy**	**Loss**	**Validation Loss**
0.8249	0.7543	0.3849	0.5296

The over-fitting is a little over 7% on the first model. Since the gap between the curves is unacceptable, it is decided to run different models and to obtain better results and to fix this problem.

On the other hand, while testing the model with the images that have been reserved and have not been subjected to data augmentation, the model gets a 76.66% accuracy. We can clearly see the over fitting problem that the model has. The problem will be addressed in the following models.

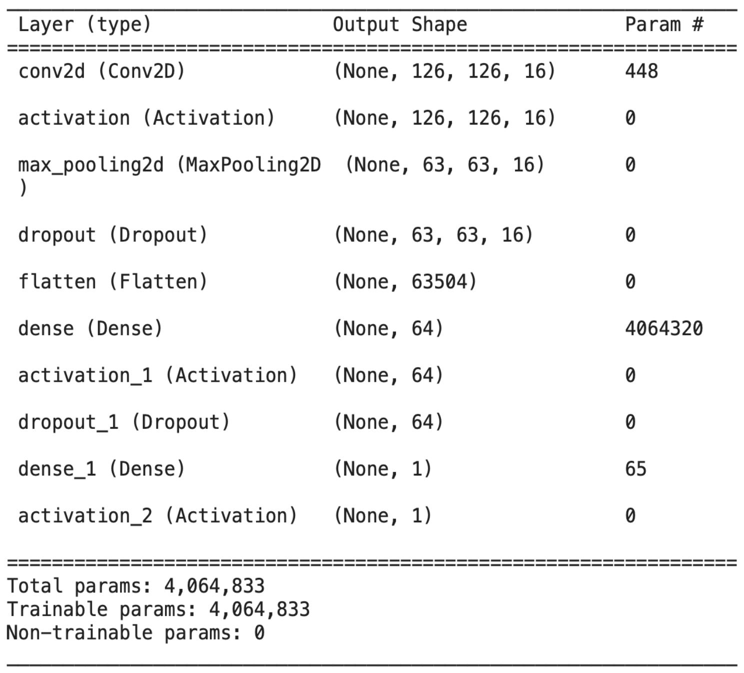
4.2 Second Model

The idea behind this second model is trying to solve the over-fitting problem that the first model had. That said, the second model introduces two things to avoid over-fitting. Then, MaxPooling and Dropout are introduced.

Since the models are a natural evolution one from another, the next set of items will explain only the new attributes included, given that, the baseline model is kept. That said, the following items show some of the characteristics chosen for the second model, that are different from the first one are:

- Dropout layer with parameter 0.3 and 0.6. The dropout layer helps avoiding overfitting by 'muting' a proportion of the neurons to the next layer.
- MaxPooling 2x2. Which means, that from a 2x2 matrix, the highes number will be chosen and kept. This reduces the amount of information to 25% of the original amount.

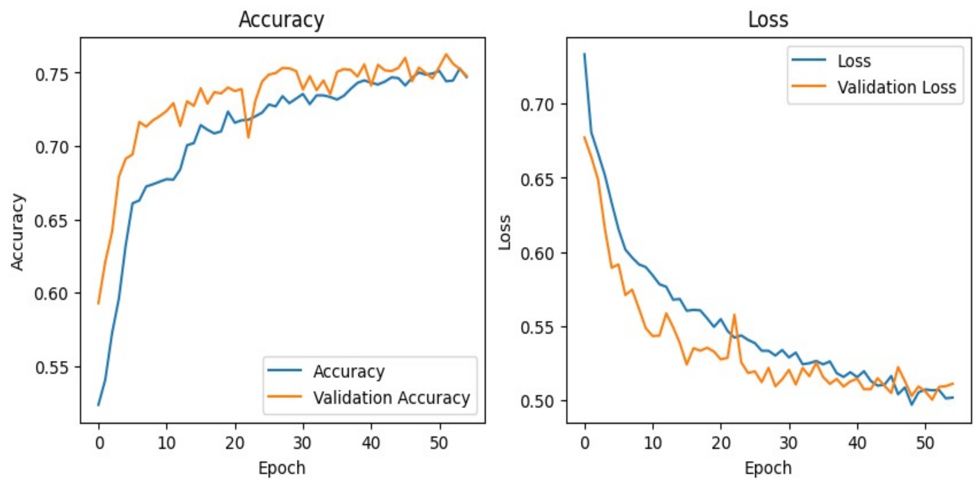
The model summary can be seen in the picture below.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 activation (Activation)     (None, 126, 126, 16)      0

 max_pooling2d (MaxPooling2D  (None, 63, 63, 16)       0
 )

 dropout (Dropout)           (None, 63, 63, 16)        0

 flatten (Flatten)           (None, 63504)             0

 dense (Dense)               (None, 64)                4064320

 activation_1 (Activation)   (None, 64)                0

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

 activation_2 (Activation)   (None, 1)                 0

=================================================================
Total params: 4,064,833
Trainable params: 4,064,833
Non-trainable params: 0
_________________________________________________________________
```

It can be clearly seen, that after dropping out a fraction of the parameters and adding MaxPooling the number of trainable parameters has reduced drastically. That said from originally 16 million to only 4 million. Even though the number is sill high, it has been greatly reduced.

As a consequence, it can be seen that the over-fitting - meaning the distance between the accuracy and the validation accuracy curves - has be reduced enormously. That can be appreciated in the graph placed below.



By the 55th epoch the following results are obtained:

Model 2			
Accuracy	**Validation Accuracy**	**Loss**	**Validation Loss**
0.7470	0.7479	0.5018	0.5111

For the final epoch, a 74.70% of accuracy is observed vs a Validation Accuracy of 74.79% . Now, given that the over-fitting problem has been greatly reduced, and that the accuracy can still be improved, the next models will be focusing on the latter.

Further on, when testing the model on the data that has been reserved and has not been altered at all (no data augmentation), the model shows a 76.27% of accuracy.

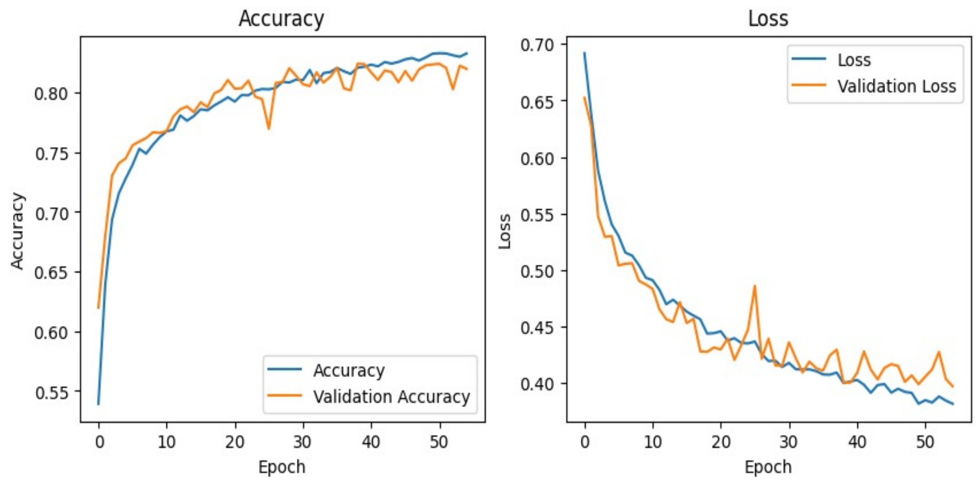
4.3 Third Model

For the third model, focus has been set on improving accuracy. That said, it was decided to increase the number of filters from 16 to 32, which allows to capture more features from our pictures and adding a Convolutional Layer. Given that the model was built with a Ceteris-Paribus methodology, then, as done before, we avoid explaining the characteristics of the model that were explained previously. Nevertheless, the results of the architecture of the model can be seen below.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 activation (Activation)     (None, 126, 126, 32)      0

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 dropout (Dropout)           (None, 63, 63, 32)        0

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        9248

 activation_1 (Activation)   (None, 61, 61, 32)        0

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 30, 30, 32)        0

 flatten (Flatten)           (None, 28800)             0

 dense (Dense)               (None, 64)                1843264

 activation_2 (Activation)   (None, 64)                0

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

 activation_3 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,853,473
Trainable params: 1,853,473
Non-trainable params: 0
_________________________________________________________________
```

It can be seen that the number of trainable parameters has also been reduced as a consequence of having included another convolutional layer. Then, from the 4 million the previous model had, now the model has less than 2 million.

The graph for accuracy and loss curves can be seen below.



By the 55th epoch the following results are obtained:

Model 3			
Accuracy	**Validation Accuracy**	**Loss**	**Validation Loss**
0.8327	0.8198	0.3816	0.3971

Further on, when tested the model on data that has not seen and that was left aside for that reason, without doing any data augmentation to it, the performance of our algorithm is 83.98%.

It is noticeable that the accuracy has gone up, and the loss gone down, while the gap between the train and validation functions has been kept decently close. That said, and even though the changes are favourable, it is decided to try to improve the performance of the model even further.

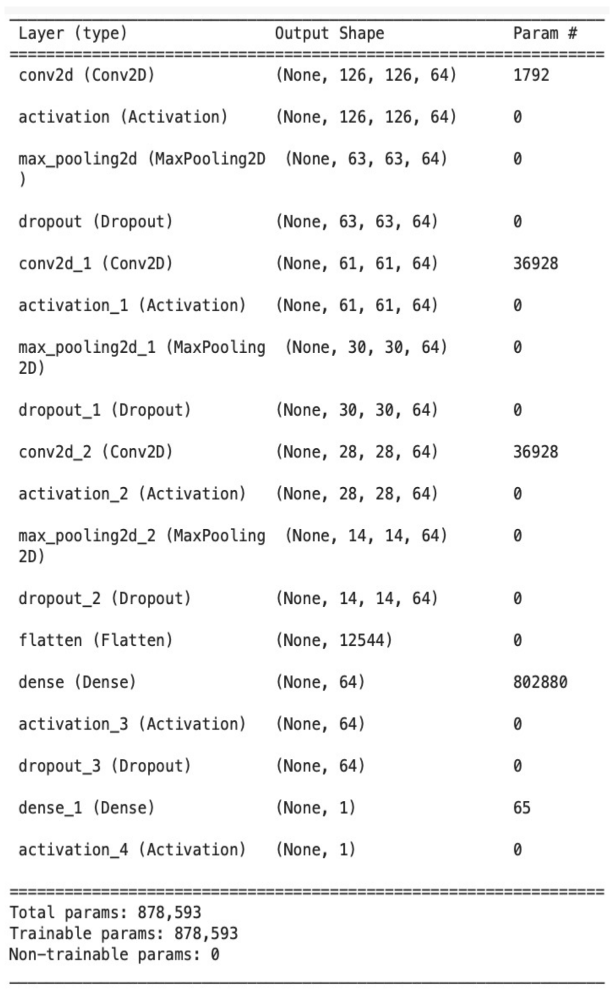
4.4 Fourth Model

The fourth model takes into consideration everything learned until this moment. Now, since it has been seen that adding a layer increases the accuracy of the model - naturally, taken into consideration the early stages of the model and the complexity it is bearing - and also that adding filters allow us to get more features from the images, which seems to translate in a higher accuracy for our algorithm, it is decided to go that way.

That said, the fourth model, increases the number of layers by one in comparison with the previous model and also doubles the number of filters from 32 to 64 in every single layer.

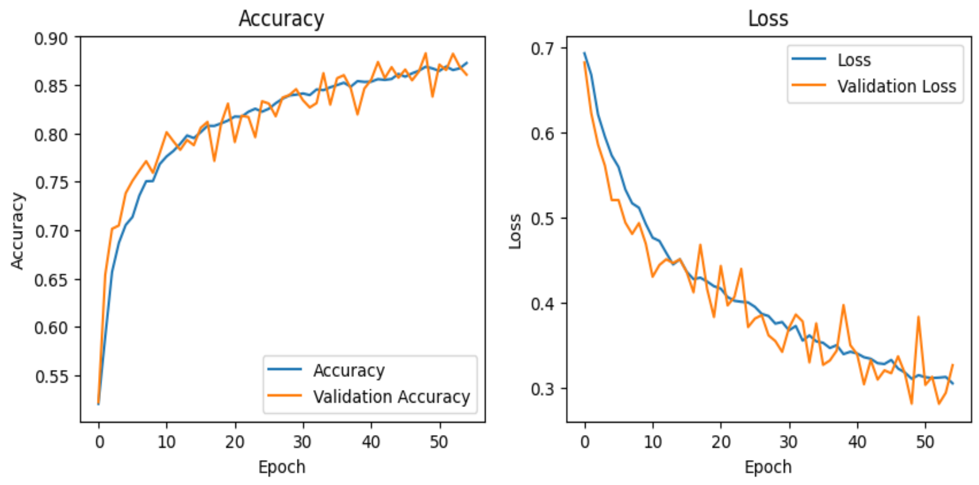
As a quick recap, it is noticeable that the number of trainable parameters has again been reduced from 1.8 million to less than 900.000. Let's remember that for this model, the rest of the hyperparameters has been left constant. The only two changes are: Only one layer has been added and the number of filters has been doubled.

The summary of the characteristics of the model can be seen in the image below.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 64)      1792

 activation (Activation)     (None, 126, 126, 64)      0

 max_pooling2d (MaxPooling2D  (None, 63, 63, 64)       0
 )

 dropout (Dropout)           (None, 63, 63, 64)        0

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        36928

 activation_1 (Activation)   (None, 61, 61, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 30, 30, 64)        0

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        36928

 activation_2 (Activation)   (None, 28, 28, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 14, 14, 64)        0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 64)                802880

 activation_3 (Activation)   (None, 64)                0

 dropout_3 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

 activation_4 (Activation)   (None, 1)                 0

=================================================================
Total params: 878,593
Trainable params: 878,593
Non-trainable params: 0
_________________________________________________________________
```

The graph for accuracy and loss curves can be seen below.



By the 55th epoch the following results are obtained:

Model 4			
Accuracy	**Validation Accuracy**	**Loss**	**Validation Loss**
0.8727	0.8602	0.3056	0.3270

Once again, it seems that the technique chosen has worked. Both accuracy has gone up, loss has gone down and the gap between train and validation curves is fairly small. It is noticeable that the accuracy has gone up, and the loss gone down, while the gap between the train and validation functions has been kept decently close. That said, and even though the changes are favourable, it is decided to try to improve the performance of the model even further.

Lastly, when tested the model on data that has not seen and that was left aside for that reason, without doing any data augmentation to it, the performance of our algorithm is 88.40%. Then, it could be stated that no overfitting problem is present within this model.

5 Cross-Validation

As for the final section of this empirical paper, it is decided to perform 5-Fold-Cross-Validation. Given that computing Cross-Validation without the proper computational power can be quite time consuming, it has been decided to only run it on the final model, hence, the Fourth Model.

On the other hand, it is important to highlight to the reader that even though the great effort put by the writer of this report, it was not possible to run 5-Fold-Cross-Validation with the same amount of epochs as it has been done in the past for every single model.

The problem naturally relies in the lack of sufficient computational power and hardware infrastructure capable of bearing with the task. That said, and after many trials, it was decided to reduce the amount of epochs to 35 per every iteration of the 5-Fold-Cross-Validation. Naturally, even though the results won't be exacly the same, after careful consideration and analysis of the values obtained up to now, it has been concluded that the result should not differ much from the value that would be obtained if more than 50 epochs were to be run. That is, because, as for the last model, the one used to run 5-Fold-Cross-Validation, a quite decently a fair stability is reached on the curves as for the epoch number 30.

That said, the table result of the 5 Fold Cross Validation is shown below.

5-Fold-Cross-Validation				
Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
0.8611	0.8667	0.8521	0.8566	0.8485

Having as a final result the mean of all of them which is shown in the table below.

Mean 5-Fold-Cross-Validation
0.8570

6 Final Comments

Through this report we got to experience one of the uses of CNN. We managed to correctly classify 85,7% of cats and dogs that were in the dataset. That said, the process to get that level of accuracy was not linear and took quite some time of hyper parameter tuning. Keeping that in mind, the next table offers a summary of the models that have been explained in the previous pages. The scope of the table is to try to catch more easily which are the factors that could help with the hyper parameter tuning on a CNN.

Summary Table: Models					
Model	**# Trainable Parameters**	**# Conv. Lay. (Filters)**	**Added Features**	**Train Acc.**	**Test Acc.**
1	16M	1 (16)	Augmentation. Dense Drop (0.2)	0.8249	0.7543
2	4M	1 (16)	Augmentation. Dense Drop (0.6). MaxPooling	0.7470	0.7479
3	1.8M	2 (32)	Augmentation. Dense Drop (0.6). MaxPooling	0.8327	0.8198
4	800K	3 (64)	Augmentation. Dense Drop (0.2). MaxPooling	0.8727	0.8686

Mean 5-Fold-Cross-Validation
0.8570

It can be seen that starting from the baseline model, the ones that followed have improved quite significantly. There is, however, place for improvement, particularly in reducing the volatility that can be seen within the graphs. With all this in mind, the final model can recognize on average almost 86% between dogs and cats - or if seen differently almost 9 out of 10 images of cats and dogs-. That at the end of the day is a decently fair result considering a first approach for the use of CNN.