Neural Networks for Binary Image Classification

Sofia Introzzi

Academic number: 967164
sofia.introzzi@studenti.unimi.it

May 2023

Abstract

This project focuses on evaluating the performance of different Neural Network (NN) models for binary image classification. For this purpose, a number of experiments were conducted using different network architectures and parameters values to observe the change in performance of the models. This task has been addressed with two different NN: the Multilayer Perceptron and the Convolutional Neural Network.

1 Introduction

Computer vision is one of the possible applications of Neural Networks. The purpose of this work is to explore this field for a binary classification problem, with particular focus on the employment of different hyper-parameter values to assess their influence on the final outcomes.
The dataset provided is a collection of images of cats and dogs. First, files have been preprocessed and then split into training, validation and test set.
This project has been addressed considering two different models: the Multilayered Perceptron and the Convolutional Neural Network.
Finally, the models have been tested on a 5-folds Cross Validation to evaluate the risk estimates.
The entire work has been implemented in TensorFlow and Keras frameworks.

2 Pre-processing

The dataset consists of 25000 images, equally divided into two classes: cats and dogs, in jpg format.

Figure 1: Dataset

First of all, some corrupted images had to be removed. In order to identify them, a function has been defined to eliminate from the analysis all images that could not be opened, specially those that do not satisfy the following two requirements: having the channels dimension equal to three or mode corresponding to 'RGB' (red, green, blue). The analysis has been carried out with coloured pictures only. After cleaning the data, the total number of objects is 24758. The dataset has been split through the take and skip functions, in a ratio of 70% for training, 15% for validation, and 15% for testing.

Before jumping into modeling, some parameters had to be configured.
Images have many different sizes, thus they have been resized into a square format, with both width and height set to 104 pixels.
Image shape has been chosen taking into consideration the limited computational power of the available resources. The batch size has been set to 40 samples.
Moreover, RGB values, that are in range [0,255], will be rescaled into [0,1] range to reduce the computational complexity. This normalisation is done inside the network architecture as first layer.

3 Models

As previously stated, in order to address this binary classification task, two type of Neural Networks have been performed. The first section presents the results of the Multilayer Perceptron NN (MLP) whilst, the second is focused on the Convolutional NN (ConvNet).

3.1 Multilayer Perceptron Neural Network

3.1.1 First model

The Multilayer Perceptron is a simple directed, acyclic graph with fully connected layers.

In this context, each image is a 104x104 pixels object with three channels (for R,G,B), therefore after the normalisation layer, through the flatten layer, our data point becomes an array of (32448, 1) dimension.
For this very first setting, two dense layers have been defined, with the first containing 80 nodes and the second 44. The final layer is the output layer, which contains two nodes, for the classes.
The activation function for those hidden layers is the Rectified Linear Unit (ReLu): a non-linear function that does not suffer from vanishing gradient.
As this is a binary classification problem, the Softmax function has been assigned to the output layer to return a probability distribution, the Adam Optimizer as optimization function, and the Sparse Categorical Crossentropy as loss function. The accuracy will be used as evaluation metrics.
In this case, the model is trained for 10 epochs.

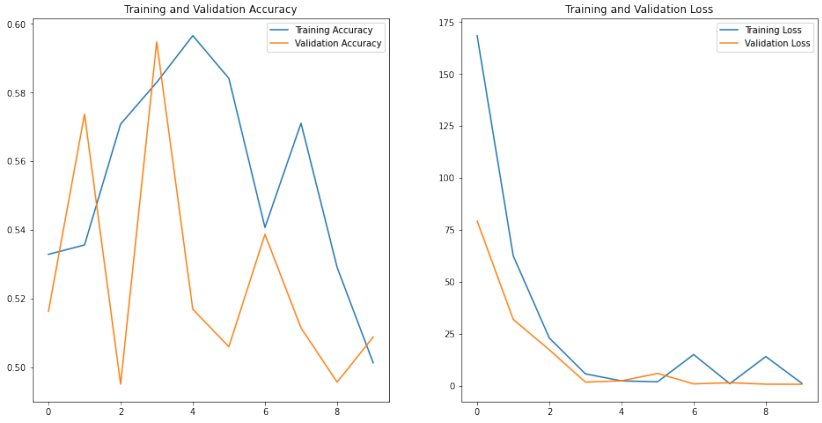

Figure 2: MLP - Model 1

It can be observed that there are fluctuations in accuracy. Here the validation loss is decreasing until epoch 2 then it starts overfitting.
In this model, training and validation accuracy keep growing until epoch 9 after that, the test accuracy starts decreasing, entering the overfit area.

3.1.2 Second model

For the second model, the same configuration has been kept, but with a decreased the number of nodes: this time, the two hidden layers have respectively 56 and 32 nodes, attempting to reduce the complexity of the model.

The model has slightly increased in some accuracy percentage points, although at the cost of higher validation loss. Moreover, overfitting is still observed from epoch 2, and the performance presents the same fluctuating trend of the before. The model is still not reliable.

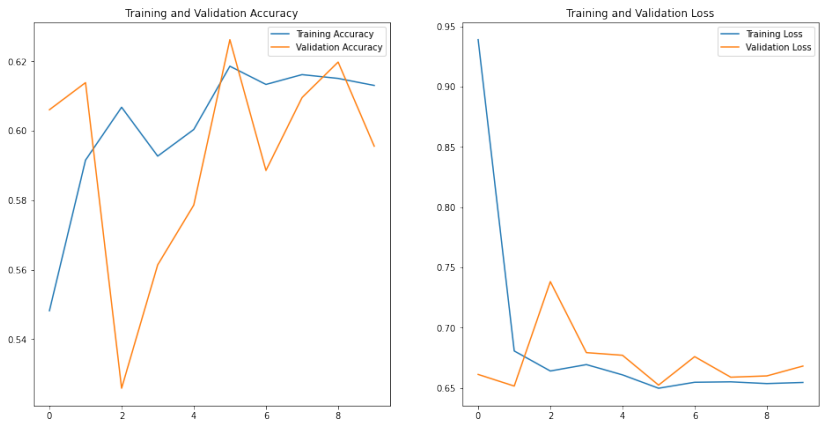

Figure 3: MLP - Model 2

3.1.3 Third model

In general, it has been proven that neural networks perform better with an higher number of layers but with a small size. It is for this reason that for the third model, an increased number of layers has been chosen. In addition to this, in order to reduce fluctuation the learning rate this time, instead of the default value 0.001 has been brought to 0.00001. The number of hidden layers are now four, with respectively, 42, 28, 14, 14 nodes.

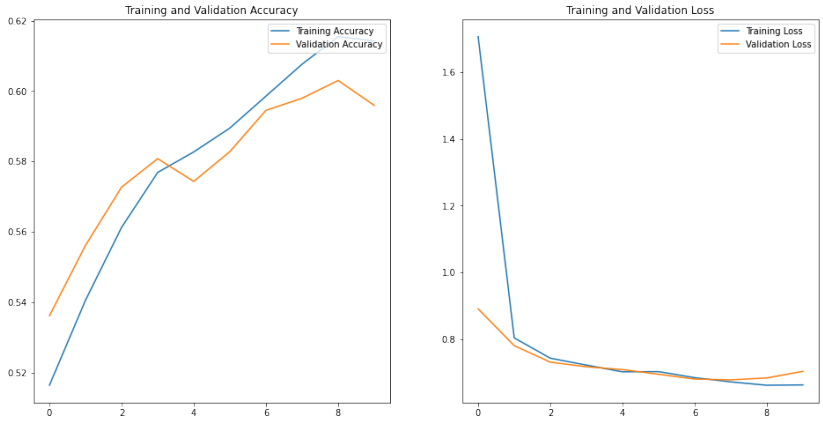

Figure 4: MLP - Model 3

As it is observable from fig. 3 the model seems to have quite improved.
The training is more stable with positive results on validation.
Finally, the optimal value is in correspondence of epoch 4. Underfitting and overfitting areas are now here clearly distinguishable.

We can now move on to the second section of our analysis.

3.2 Convolutional Neural Network

Concerning the field of images recognition, Convolutional Neural Networks are widely deployed. Namely, their ability in extracting patterns from images independently of their position, makes them a quite suitable model for this domain. This property is made possible by the kernel: a moving window of pixels that slides along the image. The geometrical structure therefore becomes fundamental for recognizing features.

In a similar manner to the previous approach, the models in section 3.2 have been trained for different configurations and evaluated the results.

3.2.1 First Model

The base design of a ConvNet is composed of a convolutional layer usually followed by a pooling layer, a flatten layer, and finally a dense layer.
Hence, initial model has been set with two convolutional layers, each one with a squared 3x3 filter size of 16 and 32 respectively. Following the same methodology, we have injected non-linearity through the rectification function.
Both convolutional layers are paired to a MaxPooling layer that downsamples the previous outputs by taking, in this case, the maximum value in a 2x2 dimensional pool.
This time, the flatten layer has been reasonably set before the output layer, precisely to be previously able to exploit the spatial hierarchy of the pixels.
The output layer is dense layer with the Softmax function.
The model has been trained for 10 epochs with the Adam optimizer and the Sparse Categorical Crossentropy loss function.

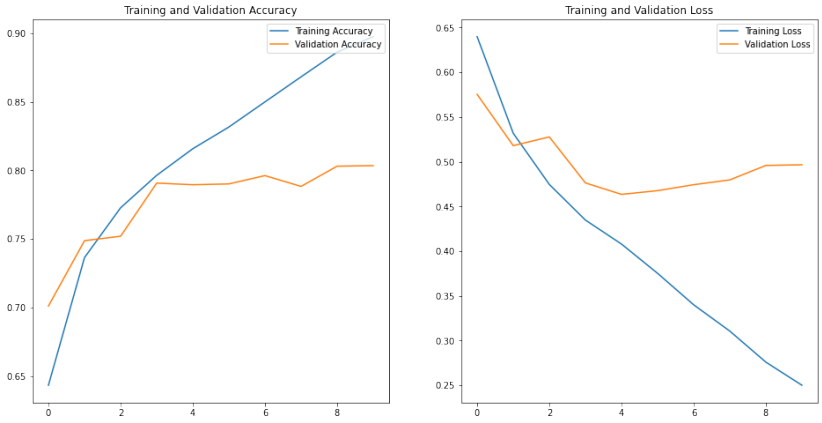

Figure 5: CNN - Model 1

From fig. 4, it can be noted that the validation loss is fairly decreasing up to epoch 5 after which it starts increasing.
The optimal performance seems to be located at epoch 2.

3.2.2 Second Model

For the second CNN model a new block is added to the previous architecture: Convolutional layer of 64 filters and a 2x2 MaxPooling layer, giving therefore a total of three convolutional and MaxPooling blocks. Also, an additional dense layer of 32 neurons has been included.

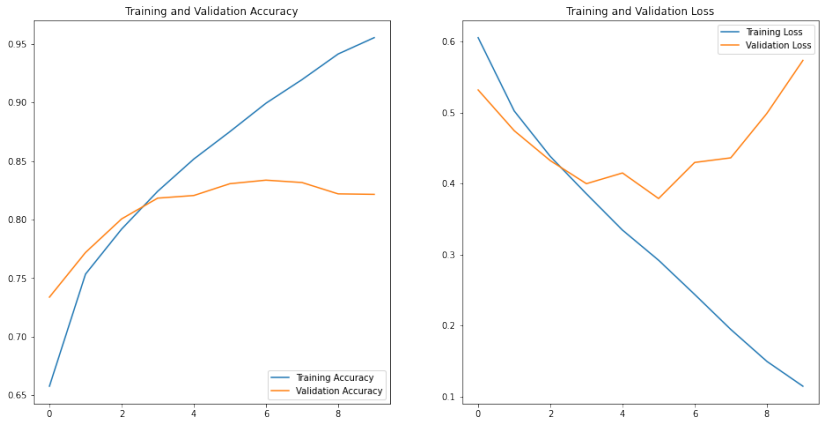

Figure 6: CNN - Model 2

From observation of fig. 5, the validation loss takes now smaller values than before of some decimal points. At level of the optimal behaviour, at epoch 3, the accuracy has increased of five percentage points. Although after epoch 3, the model is loosing its predictive power.

The drop out layer layer has been introduced to assess the behaviour of the model.

3.2.3 Third Model

The drop out layer is a regularization technique that helps in reducing overfitting. It works by randomly setting to zero some units at the specified percentage rate in the configuration. This technique has been implemented in the third model.
The drop out layer is set after each block. For the first two it has been fixed to 0.1 and to 0.2 the last one.

The architecture is therefore structured as follow: first, have the usual rescaling layer, three layers blocks composed each one of: a Convolutional layer with 3x3 filter, a MaxPooling layer of 2x2 and Dropout layer. After these three blocks, the layers we are the flatten layer and the dense layer with the SoftMax function.
As the optimal behaviours seemed to manifest quite early, the model haw been trained the model for 5 epochs only.

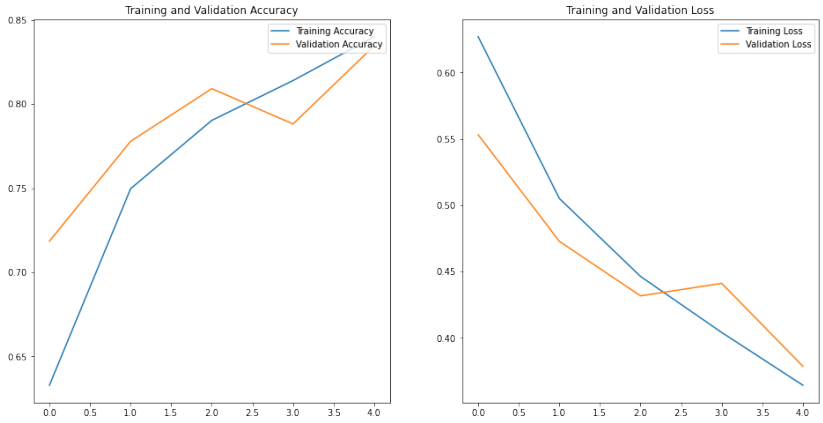

Figure 7: CNN - Model 3

The improvement after this adjustment is visible. The validation loss exhibits a decreasing trend, eventually indicating a potential convergence at the end.
The model has presented a better performance compared to previous experiments.

In conclusion, after the hyperparameters tuning, the final stage is the assessment of the effective algorithm performance on the test set.

4 K-fold Cross Validation

The aim of the cross validation is to evaluate the proper performance of the algorithm. This has been performed on the models that presented the best outcomes in training and validation for the two different Neural Networks seen in this project.
For this task, the sample that was generated for the test set in the early stages has been employed.
With the same take and skip approach where the training, validation, and test sets were created from the dataset, the test set is divided into k folds.

For each fold, the classifier is trained on k-1 folds and tested on the remaining one. In this scenario, a value of k equal to 5 was chosen, (i.e. the process was iterated 5 times). Therefore, the risk estimates was computed according to the zero-one loss, that simply counts the number of misclassification.

The selected models were evaluated firstly, for of the Multilayer Perceptron and secondly, for the Convolutional Neural Network.

Concerning the MLP, it has been verified that the general performance of our model

for image classification is quite poor. The resulting risk estimate is indeed 0.49.
Relatively to the CNN instead, the risk estimate is 0.37.

5 Conclusions

This project aimed to assess various architectures for a binary classification problem with the purpose to analyse how different choices could affect the model behaviour.

First, the Multilayer Perceptron has been implemented. It has been shown that a simple feedforward neural network actually presents difficulties in learning from images without rapidly incurring in overfitting, and therefore consequently problems in generalizations.
The above risk estimate results for this classifier have indeed confirmed this tendency, showing that performing this model would not be more accurate than making a guess.
Secondly, the Convolutional Neural Network has been exploited.
As expected, this class of NN has exhibit a better performance, once again confirmed by the risk estimates.

Concerning the first classifier, an hyper-parameter that has proven significant for the model, is the learning rate. It has indeed benefited from the slower convergence of the gradient in the learning process.
On the other hand, the CNNs due to their nature, are very effective. Precisely, they are considered a state-of-the-art methods in image classification.

In accordance with the literature, we have shown that increasing the number of hidden layers in the model, leads to an improvement in predictive power.

Moreover, the pooling and the dropping out operations have introduced some translation invariance of patterns in images, and independence of the model from the data, enhancing the performance of a model.

Finally, the expected risk estimates have underlined the importance of testing the model on unseen data samples to assess the true performance of the classifier.
Further investigation could involve testing gray scale images, or the implementation of additional regularization techniques.

References

N. Cesa-Bianchi. Neural networks and deep learning. https://cesa-bianchi.di.unimi.it/MSA/Notes/deep.pdf, 2022.

S. Shalev-Shwartz and S. Ben-David. Understanding machine learning: From theory to algorithms. *Cambridge University Press*, 2014.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *conference paper at ICLR 2015*, 2015.

TensorFlow. Sequential model. https://www.tensorflow.org/guide/keras/sequential_model, 2023a.

TensorFlow. Load. https://www.tensorflow.org/tutorials/load_data/images, 2023b.

I/We declare that this material, which I/We now submit for assessment, is entirely my/our 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/our work. I/We 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/us or any other person for assessment on this or any other course of study.