Project Report
Statistical Methods for Machine Learning: Cats vs Dogs Binary Classification

LUCA DONGHI
luca.donghi1@studenti.unimi.it, 982862

30/03/2023

1. Task

"Use Tensorflow 2 to train a neural network for the binary classification of cats and dogs based on images from *this dataset*. Images must be transformed from JPG to RGB (or grayscale) pixel values and scaled down. Experiment with different network architectures and training parameters documenting their influence on the final predictive performance. Use 5-fold cross-validation to compute your risk estimates. While the training loss can be chosen freely, the reported cross-validated estimates must be computed according to the zero-one loss."

2. Introduction

The binary classification of cat and dog images is a classic beginner project in machine learning that helps to understand the functioning of these models. This report will explain every part of the project, from the data preprocessing to the computation of the risk estimates of the final model. To accomplish the task, several Convolutional Neural Network (CNN) architectures have been trained, and each of them has been evaluated and compared to previous models to modify the models and their hyperparameters accordingly.

This research, and consequently, this report, is structured as follows. Section 3 describes the provided data, its organization, and preprocessing. Section 4 presents the model architectures and the detailed process used to arrive at the final model, including hyperparameter tuning and model selection. Section 5 evaluates the final model, showcasing the types of errors it makes and presenting the cross-validated risk estimate. Finally, Section 6 concludes the report by presenting essential observations and lessons learned throughout the project.

3. Data

The *CatsVSDogs* dataset comprises 25,000 images, including 12,500 cat images and 12,500 dog images. The images have varying sizes, qualities, and orientations.

3.1. *Data organization*

The entire dataset was utilized for training, validating, and testing the models. The original dataset comprised two folders containing cat and dog images, respectively. To preprocess the images for input into the CNN models, a new folder named "data" was created that contained all 25,000 images. To preserve the label information, the images were renamed before merging them into the new folder, with a ’0’ prefix added to the cat images and a ’1’ prefix to the dog images.
While loading the data into the coding environment two images were discarded as they were corrupted and they would have created problems later on. Then, the data was split into training, validation, and test sets using the train_test_split function from scikit-learn, with a test size of 20% and a validation size of 25% of the remaining data. (Training set = 10498 images, Validation set = 5000 images, Test set = 5000 images).

3.2. *Preprocessing*

The preprocessing of the images in the dataset involved four key steps. Firstly, all the images were resized to 150x150 to avoid any computational or memory capacity issues. Secondly, the images were converted into NumPy arrays, retaining their RGB channels, as opposed to converting them into grayscale, as I believed color to be a significant factor in distinguishing between cats and dogs. Thirdly, the pixel values were scaled down to values between 0 and 1. Normalization of the images during the preprocessing stage is essential for CNNs to improve the training convergence and prevent numerical problems. Lastly, I shuffled the images to avoid the model learning patterns based on the order of images in the dataset. Without shuffling, the model might identify patterns specific to the order of the images, leading to overfitting and inadequate generalization to new data.

4. Algorithm implementation

In this section, I will describe the model architectures and the process used to select them. To achieve this, I will use Convolutional Neural Network (CNN), which is a type of neural network that utilizes convolutional layers to extract features from images. The convolutional layer convolves a filter over the pixels of the input image to detect patterns and extract features. With each subsequent convolutional layer, the network can detect more complex features. In addition to convolutional layers, pooling layers (usually max pooling) are added to reduce the dimensionality of the feature maps and decrease the number of parameters to learn. CNNs are highly effective for image recognition tasks as they can learn to recognize complex features in images. This makes them a suitable choice for classifying cats and dogs. By selecting the appropriate architecture and hyperparameters, the models can extract relevant features from the images and classify them accurately. Therefore, the selection process will be focused on finding the optimal combination of CNN layers, filter sizes, activation functions, and regularization techniques.

4.1. *Model Selection*

During the model selection process, different architectures and hyperparameters were experimented with, and each adjustment was based on the previous model's performance. A total of 10 models were trained using two metrics: binary cross-entropy as the loss function and accuracy as a measure of how well the model can classify input images. The loss function measures the difference between the predicted probability and the actual class label. On the other hand, accuracy indicates the percentage of correctly classified images. By monitoring both metrics, we can ensure that the model is not only minimizing the loss but also achieving high accuracy, which is crucial for the model to generalize well to new and unseen data.

- **Model1**

The $model1$ serves as the baseline for the analysis as it is a very simple convolutional neural network. Its architecture is structured as follows:
- Conv2D layers: These are the convolutional layers that learn to extract features from the input images. The first layer has 32 filters, while the following layers have 64 and 128 filters. The filter size is (3, 3) for all layers, and the activation function used is *Relu*, which introduces nonlinearity and helps the network learn more complex patterns.
- MaxPooling2D layers: These layers perform downsampling, reducing the spatial dimensions of the feature maps. This helps reduce the number of parameters in the model and thus reduces the computational cost, while also improving the model's ability to recognize features in different scales and translations.
- Flatten layer: This layer reshapes the output of the previous $MaxPooling2D$ layer into a one-dimensional tensor, so it can be fed into the dense layers.
- Dense layers: These are fully connected layers that perform classification based on the extracted features. The first dense layer has 512 units with *Relu* activation. The last dense layer has only one unit with *Sigmoid* activation, which outputs the probability of the image being a dog (dog=1).
- Model compilation: The model is compiled using the *Adam* optimizer, which is an adaptive learning rate optimization algorithm. It adjusts the learning rate during training, making it a popular choice for training neural networks.

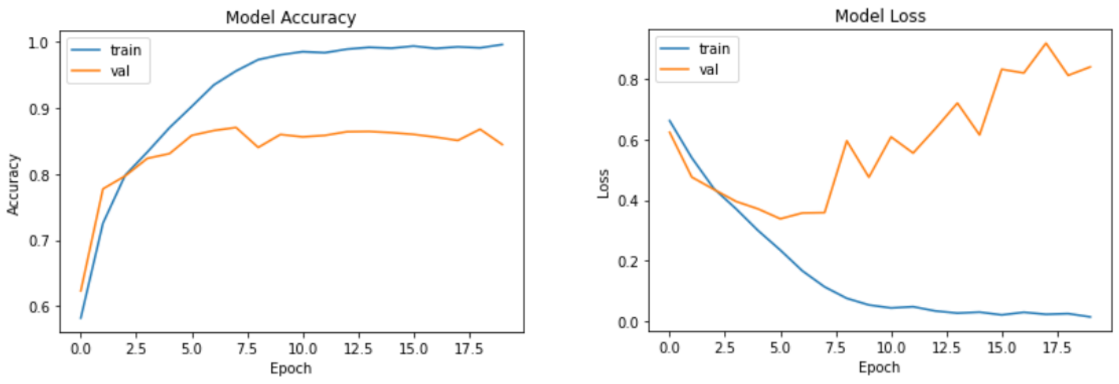

Figure 1. Model1 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

As we can see from Image 1, *model*1 exhibits significant overfitting as it achieves a high training accuracy of 99.59% but a much lower validation accuracy of 84.46%. The increasing validation loss after the 6th epoch, while training loss continues to decrease, indicates the model is fitting the training data too closely and struggles to generalize to new, unseen data. Addressing overfitting is crucial to improve the model's performance on real-world data.

- **Model2**

The *model*2 adds a Dropout layer, which combats overfitting by randomly disabling 50% of input units during training, encouraging the model to learn more generalizable features.

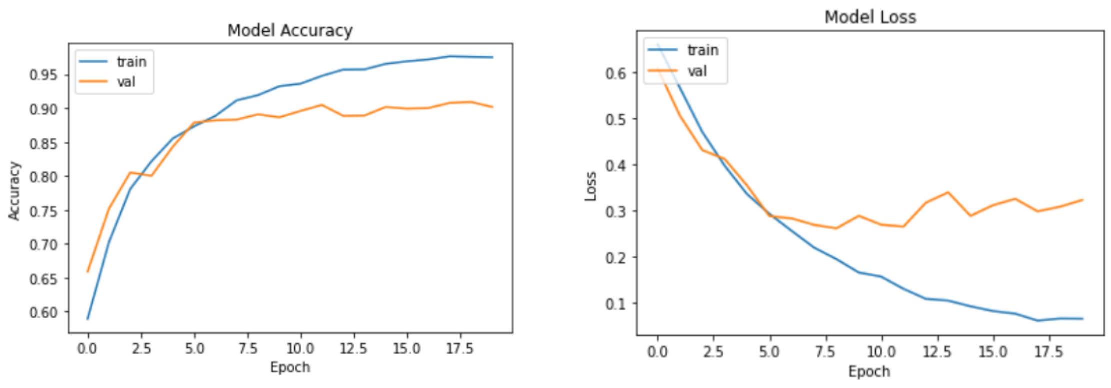

Figure 2. Model2 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

As we can see in Image 2 *model*2 has a lower training accuracy (97.49%) and higher validation accuracy (90.16%) compared to *model*1. The smaller gap between the training and validation accuracies indicates a reduced overfitting issue. Although the model still exhibits some overfitting, it generalizes better to the validation set than the first model. The validation loss starts increasing after the 12th epoch, while the training loss keeps decreasing, indicating that, even if this model is much better, there is still room for improvement in addressing overfitting.

- **Model3**

The *model*3 introduces several changes compared to the previous model:
- Additional Dropout layers: In *model*3, Dropout layers are added after each $Conv2D-MaxPooling2D$ block, with dropout rates of 10%, 20%, and 30% respectively. These additional Dropout layers help to further reduce overfitting.
- Early Stopping: An *EarlyStopping* callback is used during training, monitoring the validation loss with a *patience* of 3 epochs. This means that the training will stop if there is no improvement in validation loss for 3 consecutive epochs. The best weights, corresponding to the lowest validation loss, will be restored to the model at the end of training.

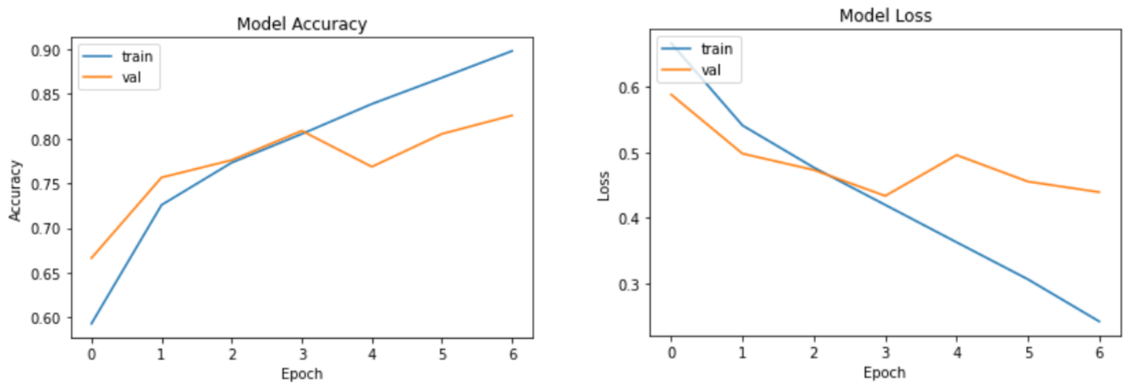

Figure 3. Model3 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

By the 7th epoch, the *model*3 achieves a training accuracy of 89.81% and a validation accuracy of 82.60%. While the smaller difference between training and validation accuracy suggests less overfitting, the fluctuation in validation loss and the overall inferior performance may indicate underfitting. This suggests the need for further refinement of the model architecture and training parameters.

- **Model4**

The *model*4 architecture introduces several changes compared to *model*3:
- Batch Normalization: In this new model, Batch Normalization layers are added after each *Conv*2*D* layer and before the Dense layer. These layers help normalize the inputs to each layer, which can speed up the training process and improve generalization.
- Activation functions: Instead of including the activation functions directly within the *Conv*2*D* and *Dense* layers, they are now separated into individual Activation layers. This allows Batch Normalization to be applied before the activation functions.
- Dropout rate: The dropout rate has been changed to a constant value of 0.2 for all Dropout layers.
- Learning rate: The learning rate for the *Adam* optimizer has been set to 0.0005, which is a smaller value compared to the default learning rate. This can help the model converge more steadily, potentially leading to better performance.

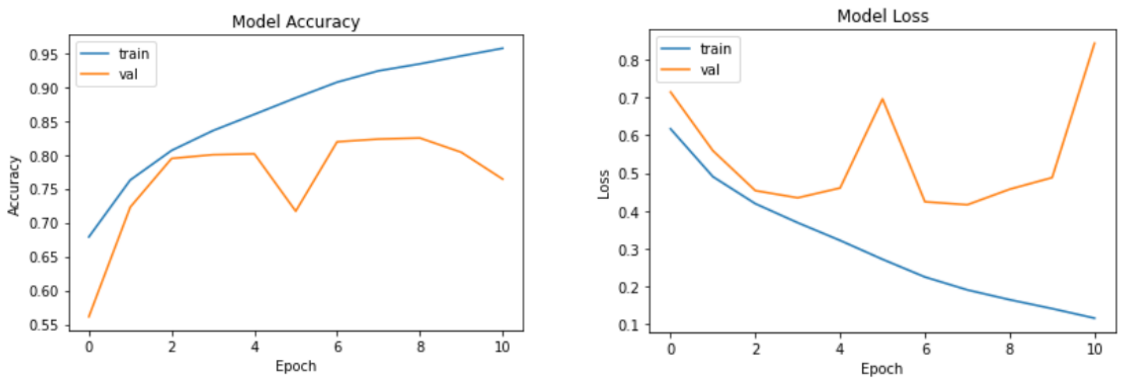

Figure 4. Model4 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*4 shows improvement in learning from the training data with a training accuracy of 95.71% compared to *model*3 (89.81%). However, it experiences again overfitting as evidenced by the increasing validation loss after epoch 7 and a significant difference between training and validation accuracy. The highest validation accuracy achieved by *model*4 is 82.48% at epoch 9, which is slightly lower than *model*3's best validation accuracy of 82.60%.

- **Model5**

The *model*5 again introduces several modifications compared to the previous model:

- **Reduced dropout rate**: The dropout rate is reduced to 0.1.
- **L2 regularization**: L2 regularization (with a lambda of 0.001) is added to the *convolutional* and *dense* layers, which helps prevent overfitting by penalizing large weights in the model.
- **Data augmentation**: The training data is augmented using rotation, width and height shifts, horizontal flips, and zoom. This increases the diversity of the training data and helps the model generalize better to unseen data.
- **Increased dropout rate after third convolutional layer**: A higher dropout rate of 0.5 is applied after the third convolutional layer to reduce overfitting on deeper layers.

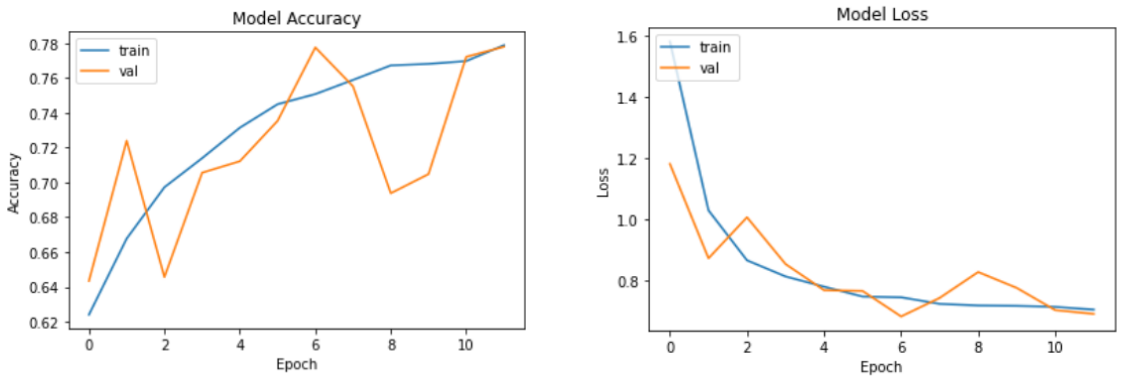

Figure 5. Model5 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

As we can see looking at Image 5 *model*5 The model doesn't seem to be overfitting, but its overall performance is bad. The training accuracy increases up to 77.89% and the validation accuracy reaches its peak at 77.76% in epoch 7 and fluctuates afterward. It's also worth noting that validation accuracy is sometimes higher than training accuracy due to regularization techniques applied only during training and data augmentation.

- **Model6**

The *model*6 differs from the previous model by adding two more *convolutional* layers, making the architecture deeper. It also includes an additional *Dense* layer with 1024 units. These changes aim to improve the model's capacity to learn more complex and hierarchical features.

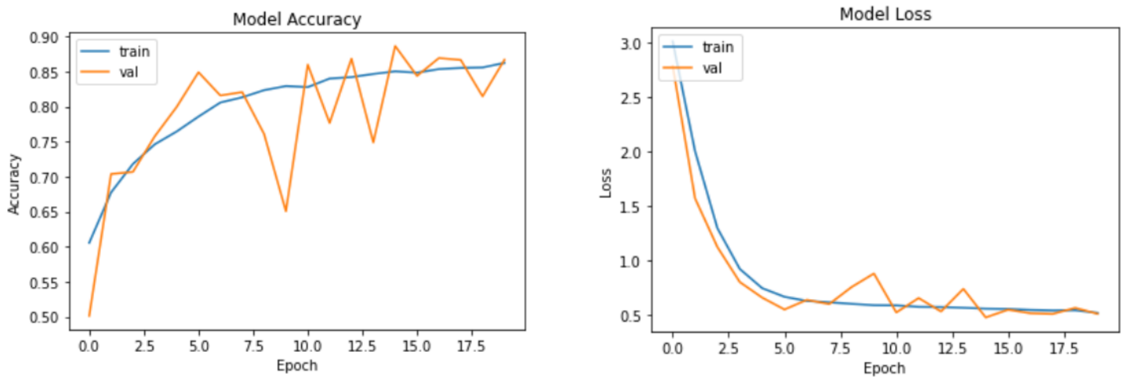

Figure 6. Model6 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*6 shows great improvement in both training and validation accuracy compared to the previous model. Any overfitting is minimal, thanks to the use of regularization techniques and data augmentation. It achieved 88.64% validation accuracy at epoch 15, thanks to its deeper architecture and additional Dense layer. Even if this model has more convolutional layers, the training times are similar to the ones of the previous models.

- **Model7**

The *model*7 introduces a *GlobalAveragePooling*2*D* layer instead of the *Flatten* layer used in the previous model. Global average pooling reduces the spatial dimensions of the feature maps by computing the average value of each channel, resulting in a smaller and more compact representation. This change helps reduce the total number of parameters, which can help prevent overfitting and improve training efficiency.

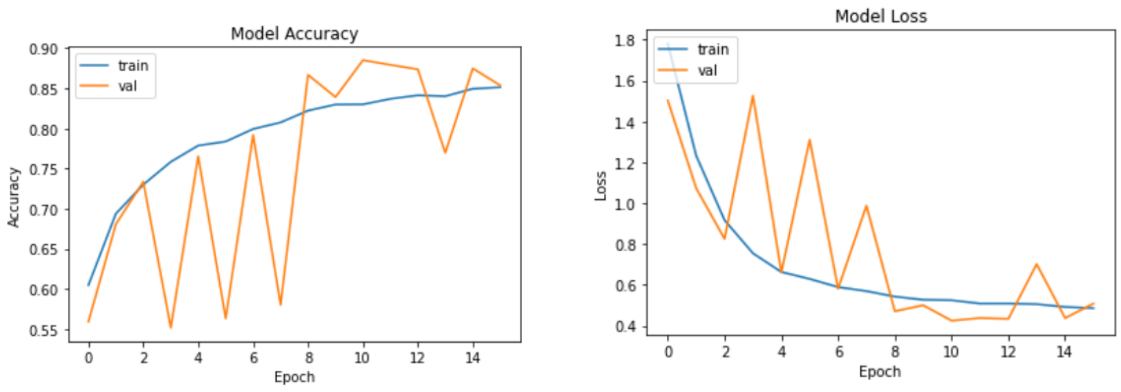

Figure 7. Model7 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*7 exhibits a steady improvement in validation accuracy, reaching its highest point at 88.50% in epoch 11. When compared to *model*6, which achieved a peak validation accuracy of 88.64% at epoch 15, *model*7 demonstrates slightly worse performance. Like *model*6, *model*7 also experiences fluctuations in validation accuracy and loss, indicating potential benefits from further fine-tuning or additional regularization techniques for more stable performance.

- **Model8**

The *model*8 introduces several changes compared to the previous one:
- **Reduced dropout rate**: The *dropout_rate* is reduced to 0.1
- **Lower learning rate**: The *learning_rate* is lowered to 0.0005, which may help the model converge more smoothly and find better local minima.
- **L2 regularization**: It is added to the *convolutional* and *dense* layers with an *l2_lambda* value of 0.001.
- **Increased batch size**: The *batch_size* is increased to 64, which can provide a better estimate of the gradient and potentially accelerate training.

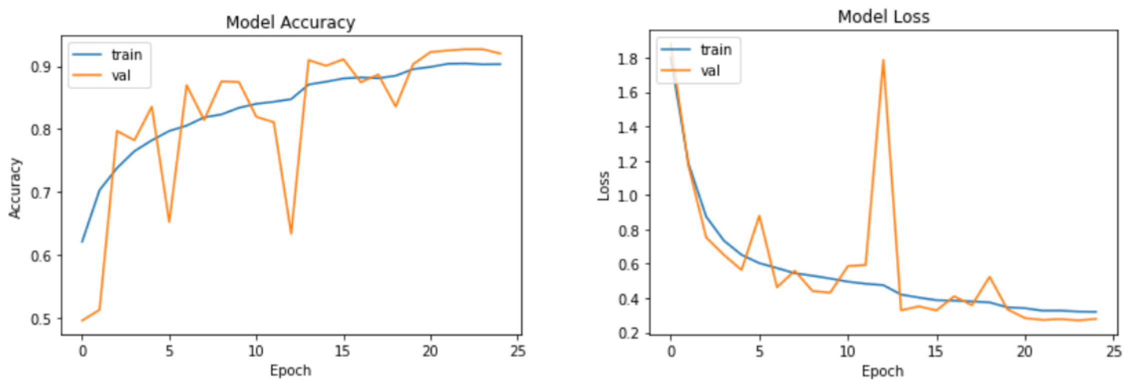

Figure 8. Model8 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*8 demonstrates a noticeable improvement over *model*7. The highest validation accuracy achieved by *model*8 is 92.68% at epoch 23, compared to *model*7's peak validation accuracy of 88.50% at epoch 11. Moreover, *model*8 appears to have a more stable performance, with smaller fluctuations in validation accuracy and loss, compared to *model*7. This suggests that *model*8 has a better balance between learning capacity and generalization, leading to a more robust and accurate model. However, I think that the model could do even better if trained for more epochs.

- **Model9**

In *model*9, the maximum number of epochs has been increased from 25 to 60, allowing the model more time to learn from the dataset. This change aims to explore whether additional training time could further improve the *model*8's performance. All other aspects of *model*9 remain the same as in *model*8, including the architecture, L2 regularization, dropout rates, and data augmentation.

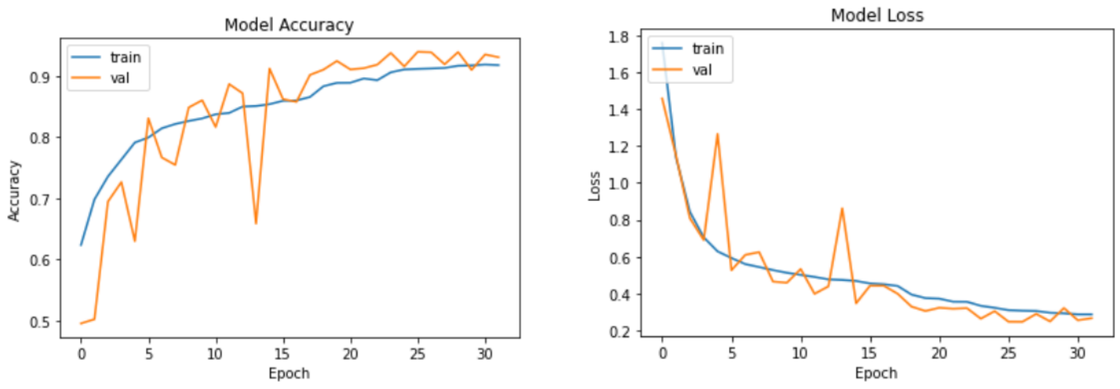

Figure 9. Model9 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*9 exhibits a promising overall performance concerning accuracy and loss. It reached a peak validation accuracy of 93.94% at epoch 26, indicating that extended training led to better results. In comparison to prior models, *model*9 consistently displayed an increasing and relatively stable validation accuracy throughout the epochs. Its accuracy progressively improved over time without any evidence of overfitting.

- **Model10**

The *model*10 introduces a few modifications compared to the previous *model*9 to make it deeper:

- Additional convolutional layers: The *model*10 incorporates an extra convolutional layer in the 64, 128, 256 filter sections, which could potentially help the model extract more complex features from the images.

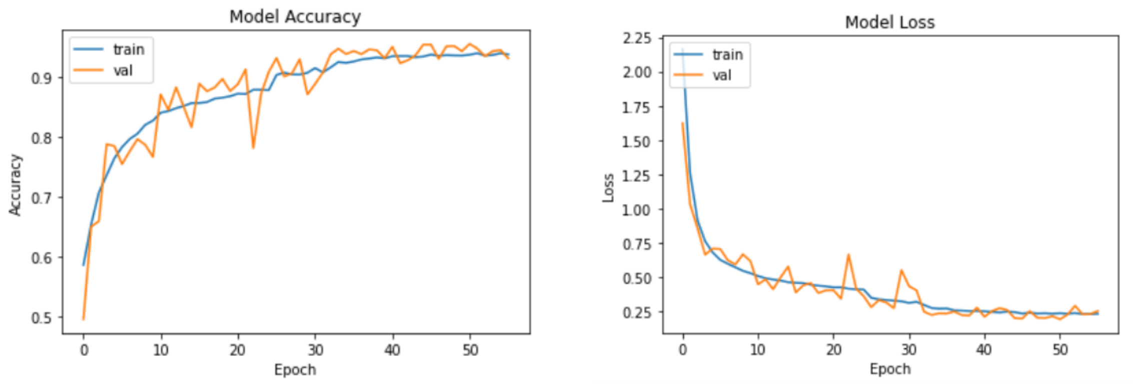

Figure 10. Model10 Training and Validation Metrics. The left figure shows the accuracy while the right one shows the loss.

The *model*10 stands out as the best model trained this far. It achieved a peak validation accuracy of 95.52% at epoch 51, indicating that the deeper architecture helped the model comprehend more complex features. Throughout the epochs, *model*10 consistently displayed increasing and relatively stable validation accuracy. Its accuracy progressively improved over time without any indication of overfitting. However, the training time doubled compared to *model*9 suggesting that achieving slightly higher performances will require a considerable amount of time and computational resources. Therefore, *model*10 is a satisfying final model.

5. Final model evaluation

In this Section, I will evaluate *model*10 performances on the test set, analyze the misclassified images, and present the cross-validated risk estimates.
The test set metrics for *model*10 are:
- loss: 0.1972
- accuracy: 0.9552
95.5% of the test set images have been correctly classified.

5.1. *Misclassified images*

In Figure 11, the confusion matrix shows that the performance is pretty balanced: the proportion of mistakes on dog images and on cat images is almost the same.

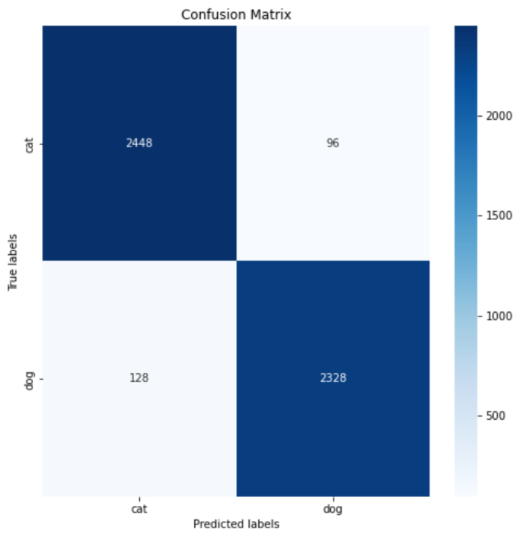

Figure 11. Model10 Confusion Matrix.

In Figure 12, a random sample of images that were misclassified by *model*10 on the test set is shown. While some mistakes may be due to difficult images with poor quality or unusual positioning, it's also apparent that there are some relatively straightforward images. This suggests that there is potential for improvement and that with more time and computational resources, it may be possible to train a better model.

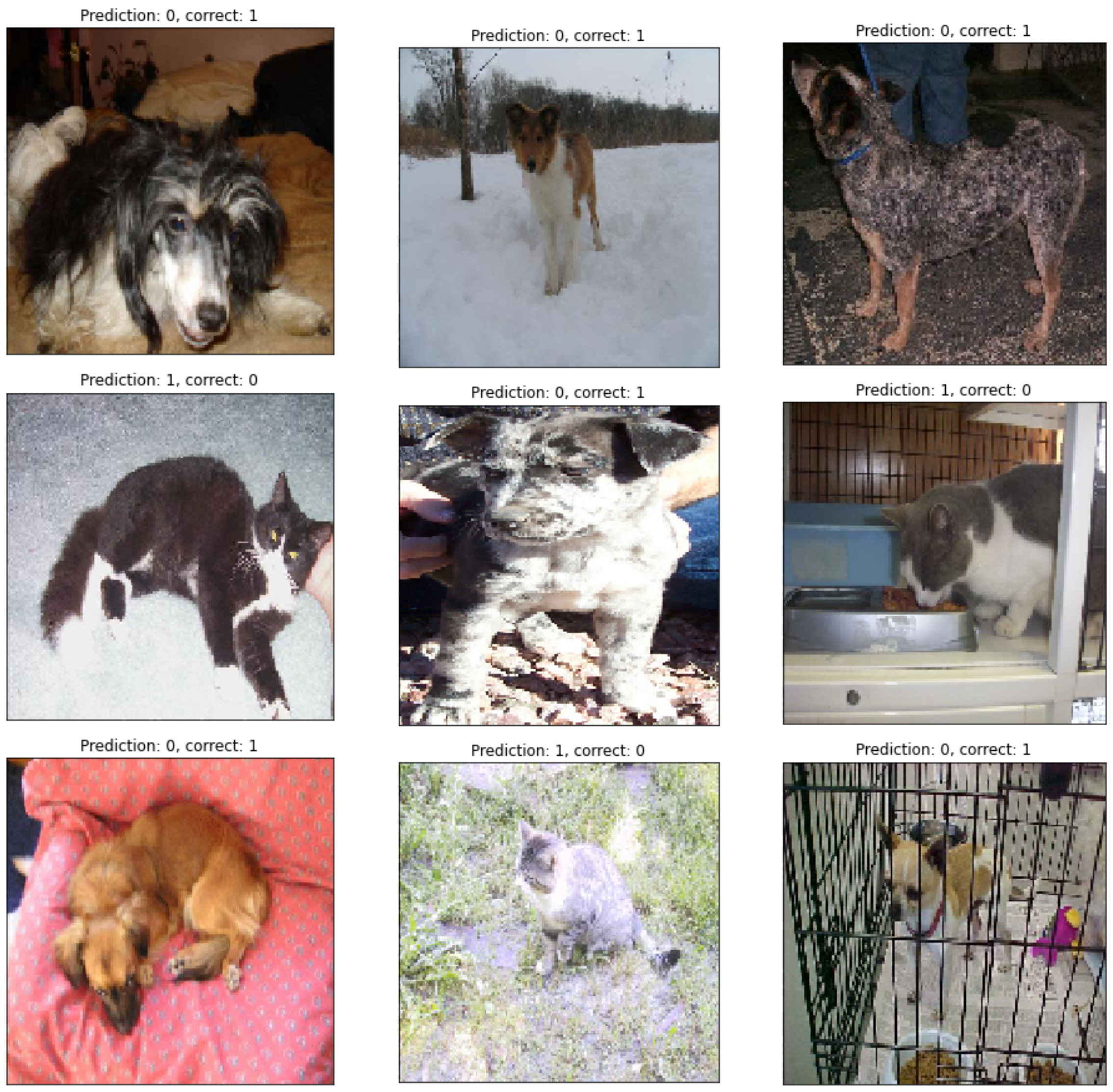
Prediction: 0, correct: 1
Prediction: 0, correct: 1
Prediction: 0, correct: 1
Prediction: 1, correct: 0
Prediction: 0, correct: 1
Prediction: 1, correct: 0
Prediction: 0, correct: 1
Prediction: 1, correct: 0
Prediction: 0, correct: 1

Figure 12. Model10 test set misclassified images.

5.2. *Model10 cross-validation*

Finally, I used 5-fold cross-validation to compute the risk estimates of the final model. As required by the task, even if I used the binary cross-entropy to train all the models, I used the zero-one loss for the cross-validation.

Zero-One Loss	*Model10*
Fold 1	0.0487
Fold 2	0.0403
Fold 3	0.0497
Fold 4	0.0428
Fold 5	0.0363
Mean	**0.0436**

Table 1: Model10 Cross-Validated Risk Estimate.

As shown in Table 1 the final cross-validated risk estimate of *model*10 is 0.0436.

6. Conclusions

In this report, we have explored the development of a deep learning model for the classification of cats and dogs in images. We started by training a basic model and gradually introducing improvements to its architecture, regularization techniques, and data augmentation. After evaluating several models, we arrived at a final model, *model*10, which achieved satisfying 95.52% accuracy on the test set and 0.0436 as a cross-validated risk estimate.
While the final model showed strong overall performance the analysis of misclassified images showed that there were still some challenging images where it struggled to correctly assign the label. These results suggest that with more time and computational resources, it may be possible to train even better models that perform more consistently across a wider range of images.

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.