Cats and Dogs Breed Classification

Mariana Santos (nmec: 93257)
DETI
Universidade de Aveiro
Aveiro, Portugal
marianasps@ua.pt

Pedro Silva (nmec: 93011)
DETI
Universidade de Aveiro
Aveiro, Portugal
pmasilva20@ua.pt

***Abstract*—Breed recognition of common household pets like cats and dogs is an important part of the work done by various organizations like kennels and catteries. This is because breed characteristics have a major impact on what type of care might be needed.**

The Cats and Dogs Breeds Classification Oxford Dataset contains 37 different classes with roughly 200 images each. By experimenting with various types of Neural Networks with this comprehensive data set it was possible to find a model capable of identifying a pet's breed with an accuracy superior to 90%.

***Index Terms*—neural networks, image classification, pet breed classification, transfer learning, InceptionV3**

I. Introduction

The following report serves as a summary of the various methods and analysis conducted in the context of the first project of TAA - Tópicos de Aprendizagem Automática, a course at Universidade de Aveiro, lectured by professor Pétia Georgieva. The knowledge applied in this project include both techniques covered in class and also some learned autonomously.

The goal of this project was to design an algorithm capable of distinguishing cats and dogs between 37 different breeds. Our main motivation for this topics are the applications that a classifier like this could have in the real world. In kennels and catteries, the registering of animals can be bureaucratic and cumbersome. Automating the breed classification part would improve and simplify this process, allowing animals to get the correct care they need faster. Other reason that motivated us was the fact that some breeds are hard to identify even for an experienced human observer, and this presented an interesting challenge to us.

In order to solve this problem, we started to analyse the provided data set, analysis present in section II. Reading other documents on this matter indicated us that the best results would be achieved though the use of a **neural network**. **Transfer learning** was also referred as a good method to not only improve the model performance, but also reduce computational complexity. [1] [2]

With this in mind, it was decided that we would experiment with a simple classic neural network architecture, and compare it with a much more complex model. This last one would not only use deep learning techniques, but also transfer learning.

We used 3-Fold Cross-Validation in order to find the best hyper-parameters for the models, and in subsection IV-B this process is discussed, among with the obtained results. The score and comparison between these 2 models is presented in section V.

Furthermore, with the goal of improving the model performance, we experimented with data set augmentation techniques, removing the background, and fine tuning. The procedure and results are in section VI.

II. Data set Analysis

The used data set can be divided in 2 parts: the images and annotations. Both are detailed below.

A. Images

There are a total of 7384 images of pets, with 37 breeds present, both cats' and dogs'. This means there are 37 different classes in which the model can classify an image. These images are set in a big variety of environments and locations, and are very different in style. Most importantly, they are casual pet images, which means that this solution could be used in real world applications where people with no experience and with no special equipment could use it.

The following figure (Fig. 1) presents one image of each breed present in the data set. These have been previously reshaped to be the same size of 299 by 299 pixels. The number on top of the images is just an indicator of the ID of the breed provided in the data set documentation.

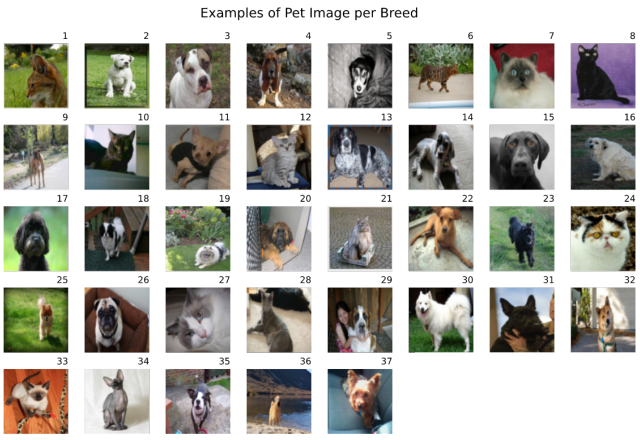

Fig. 1. Examples of images in the data set

It is important to guarantee that all classes have roughly the same representation in the data set to prevent the model

from not learning all breeds equally. Thus it is important to understand the distribution and representation of all classes. In order to have a better understanding of how the situation was in our data, we made the following chart (Fig. 2).

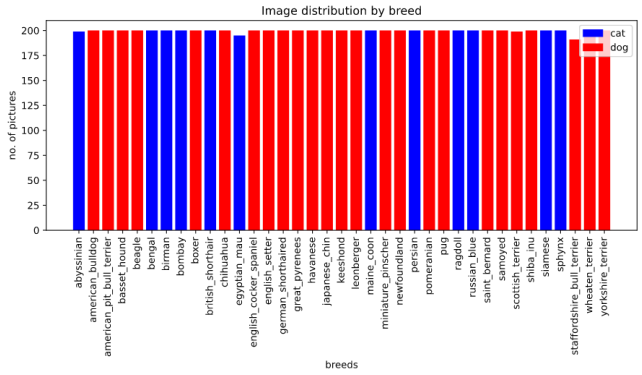

Fig. 2. Image distribution by breed

It is possible to see that all breeds have a similar representation, or number of images, in the data set. There is around 200 images containing pictures of pets from each breed. It is also noticeable the considerable difference between the numbers of pictures containing dogs and those containing cats, being that the dogs' have more than twice the number of images than cats'. However, this shouldn't be a problem since the classifier isn't trying to distinguish between those two classes, but between the 37 different breeds.

B. Annotations

The data set also includes extra information, including:

a) Trimap Annotations: There is one trimap annotation for each picture indicating the pixels that are foreground and background. Background removal using these annotations are used used in subsection VI-B.

b) Head Bounding Box Annotations: Annotations in PASCAL VOC Format that were not taken advantage of in this project.

c) Other Documents: Files such as a text file containing information for each image (class ID, species, etc), and two files describing possible train and test image splits that were also not used.

III. Fundamentals

A. General Aspects

This problem is considered one of fine-grained image classification, because it focuses on distinguishing between classes with very similar features. We were aware that with classic methods of machine learning it would be impossible to develop a good solution.

Most documents analysed explored the use of convolutional neural networks (CNN), since they are the most viable for problems which involve large quantities of complex data from which abstract features like form and color must be obtained to get accurate results.

In all the models we created, the loss function used was **Categorical Cross-Entropy**, also known as **Softmax loss**, common for multi-class classification. The optimizer was **Adam**, an implementation of stochastic gradient descent, used with a default learning rate of 0.001. This parameter was one of the hyper-parameters varied during the hyper-parameter tuning phase, addressed in subsection IV-B. We chose **accuracy** as the metric to evaluate the model performance during training and tuning.

B. Simple Neural Network Architecture

We decided to try with a simple architecture, not applying the concept of transfer learning, but that still could be considered deep learning.

It is composed by the following layers, in order:

a) Pre-processing Layer: The first layer that will interact directly with the input data is a pre-processing one. It will rescale every value it receives to a value between -1 and 1.

b) Flatten Layer: This layer *flattens* its input, i.e. transforms the input to a single column. This is to prepare the data to be fed into the fully connected layer.

c) Dense Layer: A single fully connected layer with 256 neurons by default. Various experiences were made to find out the best number for this parameter, and these will be further addressed in paragraph IV-B1c. The activation function chosen was **ReLU**, which simply maps any negative input to 0, and any positive input to itself. It is widely used because it's a linear function for positive input and does not have any risk of saturation.

d) Dropout Layer: The purpose of which is to ignore randomly a certain percentage of layer outputs of previous layer depending on a regularization parameter. The goal of this layer is to prevent over-fitting of data. We also experimented with various values for this parameter in paragraph IV-B1b.

e) Batch Normalization Layer: With the purpose of scaling the output in order so that it has a mean of 0 and a standard deviation of 1.

f) Dense Layer: The output layer, a fully connected one with 37 neurons, each corresponding to one of our classification classes. Here a **SoftMax** activation function is used. This is a better activation function for a multi-classification problem when compared to a logistic function, like **sigmoid**. This is because SoftMax outputs a probability vector with each class corresponding to a value, allowing to easily understand what classes have higher values.

In Fig. 3 we can see an automatic generated diagram of the architecture of this network.

C. Complex Neural Network Architecture

As already mentioned, transfer learning seems to have a big importance in the development of new models. In this method of optimization, pre-trained models are used as the starting point, allowing improved performance in modeling the new problem. [1] Usually some new layers are added after the layers of the base model, that is already trained. The layers of this model should be "frozen" to avoid destroying

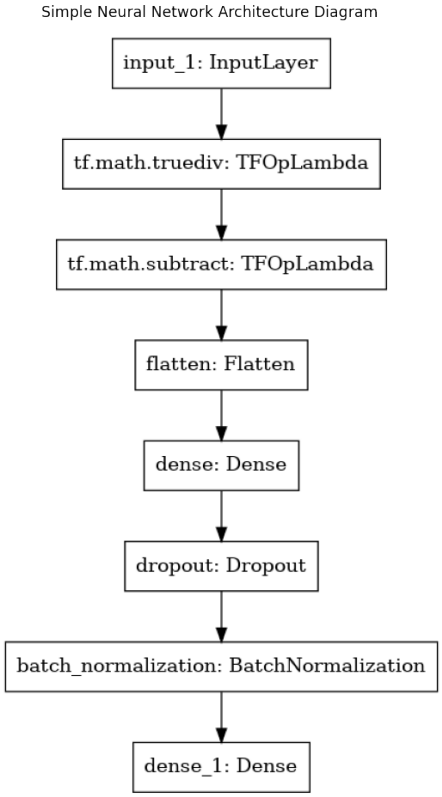

Fig. 3. Simple neural network architecture diagram

the information they already have during the fitting phase. The purpose of the new layers is to train on the new features and predict using the new data set.

The **InceptionV3** architecture pre-trained on the **imagenet** data set was mentioned various times in the papers we read as being a good architecture for a transfer learning base model for problems similar to this one. [2] [3] That's why we decided to use it.

This Neural Network is composed of multiple combinations of different layers from which we can identify the following ones:

a) Convolution Layer: Layers of this type work as filters to a given input. A matrix of inputs is multiplied by a smaller matrix of weights called **filters** through a dot product. This filter is usually applied sequentially to parts of the input matrix, usually with some stride/step value, until all parts of it have been covered. Then, each one of this products are summed, each one resulting in a single value, together forming the output matrix. The use of this layer allows us to capture higher abstraction features out of our inputs, for example, forms from a complex input such as an image.

b) Batch Normalization Layer: Just as stated before, this layer scales its input so that it has a mean of 0 and a standard deviation of 1. This is used as a way of stabilizing the learning process and reducing the number of epochs required.

c) Pooling Layer: The features obtained from Convolution Layers might be sensitive to their position on an image. These Pooling Layers are used to down sample these outputs by applying a filter. This filter will form pixel groups and apply an operation on them. In this model there are two examples of this layer: Max Pooling - which returns the maximum value for each group of pixels; and Average Pooling - which calculates their average.

As mentioned before, more layers are added to the output of this base model in order to allow it to predict data based on the new information. The layer that will interact directly with its output is an Average Pooling, and it is followed by all the layers described in subsection III-B, by the same order, except the first one. This first one, just like in the other network, is the first layer with which the input data interacts, and is positioned before the base model. It's goal is to prepare the images to be in the right format to be passed to the InceptionV3 model. Like already mentioned, pixels are re-scaled to values between -1 and 1.

The diagram in Fig. 4 represents the layers of the developed model architecture.

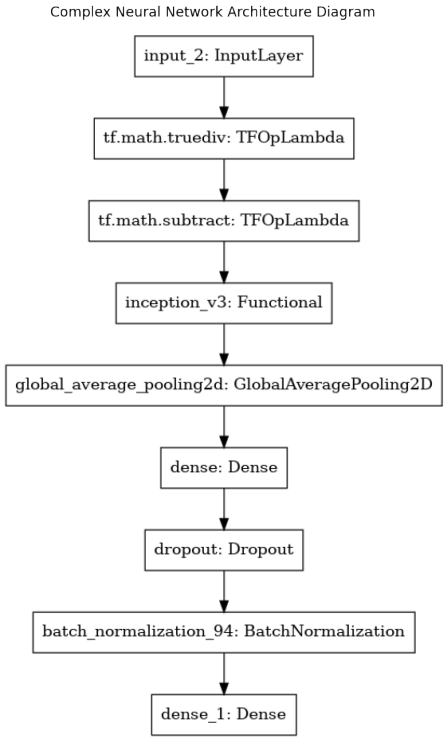

Fig. 4. Complex neural network architecture diagram

IV. Pre-processing and Hyper-parameter Tuning

A. Image Pre-processing

In order for the images to be ready to be used in the models, they had to go through a pre-processing phase. This phase included reshaping them to (299, 299, 3) tensors, which is the recommended shape of the InceptionV3 model input. Changing the color map of the images to black and white, which is a common pre-processing process, was not considered. This is because for this problem the colours of the

animals often have an importance to distinguish them, and also because the recommended input is expecting a 3 dimensional input.

Then, the data was split in 2 parts: train data and test data. The train data corresponds to 70% of the total images, while the test data corresponds to the remaining 30%. Test data was left aside for the processes covered in this section.

In order to find the best model configuration we experimented with each one of the models described while varying various hyper-parameters. This way we were able to select the best values for the parameters so as to design the most adequate model. The selection was based on the values of accuracy and loss, both for the training and validation data.

B. Hyper-parameter Optimization

With the goal of optimizing the model's configuration, each one of them went through a iterative process. This process included varying various hyper-parameters, namely the learning rate, the dropout percentage, and in one of the models also the number of neurons in the hidden layer. The method applied was 3-Fold Cross-Validation.

A list of values we thought could be good to test was created for each of the parameters. Only two values were experimented with for each of the two hyper-parameters. This is far from ideal, but justified by the scarcity of time and low computational resources. Then, for each of these parameters, 3-Fold Cross-Validation was performed, and the history information was saved. This information contains data about the progression of the loss function and accuracy over each epoch, and graphics using it will be shown in this section.

K-Fold Cross-Validation works by dividing the training data in K parts, or folds. Then the model will be trained on K-1 parts, and validated on the remaining part. This is done K times, being that every fold will have an opportunity to have the role of validation. Then an average is made on the results. This tries to solve the problem that the validation data used can give a poor representation of the model performance. Because all data is used for validation indirectly, this situation is not going to happen.

In this case, the data was divided in 3 folds mainly because of low resources and lack of available time, since the most common number of folds seem to be 5 or 10. [1] Each 3-Fold Cross-Validation process took between 30 minutes and 1 hour to finish.

1) Simple Neural Net: The goal of this experiment was to find the best values for the learning rate, the number of neurons in the hidden dense layer, and the dropout percentage. We decided on training the model with a 32 batch size, and 35 epochs. Ideally these two last mentioned values would also been subject to experiment.

a) Learning Rate: Two values for the learning rate were used: 0.01 and 0.001. Though the analysis of the charts present at Fig 5, we can note that having a higher learning rate helped reducing the train loss more quickly, even though in none of the experiences the model converged in the 35 epochs. Because the lower learning rate had less discrepancy between the train and validation loss, we decided to use it. None of the settings show good results, though.

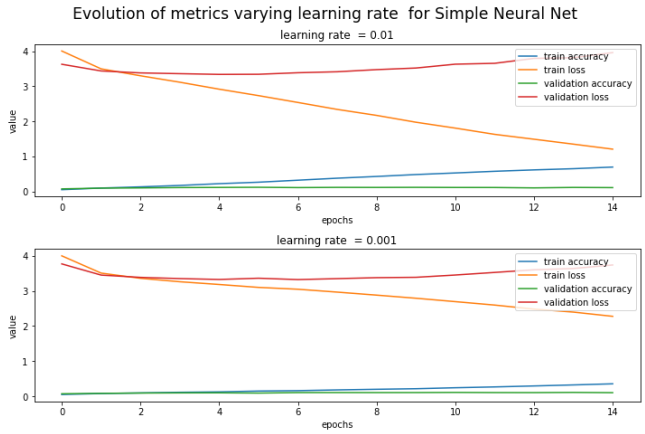

Fig. 5. Simple neural network results for learning rates 0.01 and 0.001

b) Dropout Rate: Another parameter observed was the dropout value. As we can see in Fig. 6, having a higher value lead to a lower slope of the train loss, while the other metrics stayed roughly the same. Even though neither of the two configurations showed acceptable results for us, the one with the higher dropout rate was the chosen one because of the less difference between final values of train and validation loss.

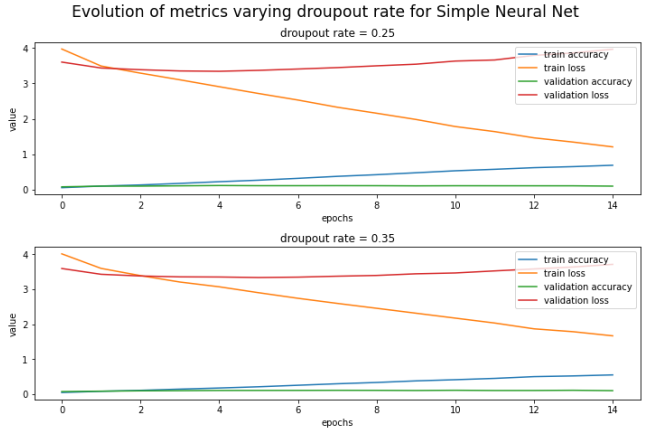

Fig. 6. Simple neural network results for dropout rates 0.25 and 0.35

c) Number of Neurons: The last parameter tuned was the number of neurons in the penultimate layer of network, a fully connected layer. As can be observed in the Fig. 7, changes to this layer are very sensible. While a low number of neurons resulted in a somewhat reasonable loss, doubling them resulted in astronomical values of loss, proving that an architecture like this one is unable to cope with such large quantities of information.

2) Complex Neural Network: For the more complex neural network, only the learning rate and the dropout percentage were tuned, since the number of neurons in the hidden layer

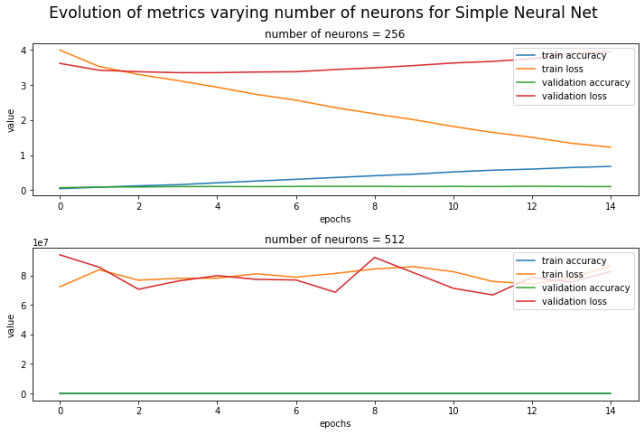

Fig. 7. Simple neural network results for differing number of neurons for penultimate layer

had to be fixed due to the architecture. We set the number of epochs to 15, and a batch size of 32. These are hyper-parameters that should've also been through experimentation to understand the best value. Once again, our resources didn't allow us to perform such tasks.

a) Learning rate: The values tested for this parameter were 0.01 and 0.001. The graphs in Fig. 8 show the evolution of the accuracy and loss function, both for the train data and the validation data. As referred above, these are the average of the results gotten from all the iterations in the 3-Fold Cross-Validation.

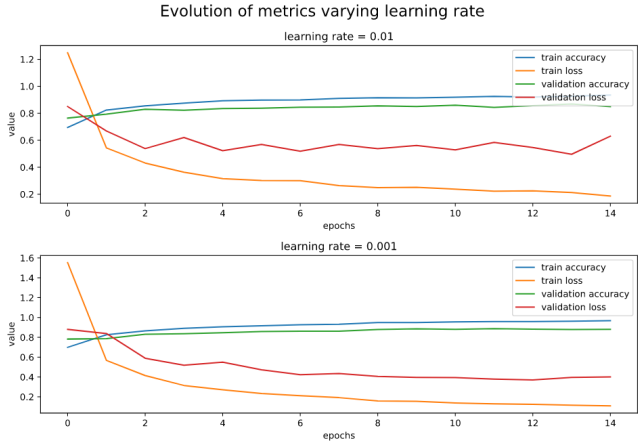

Fig. 8. Complex neural network results for learning rates 0.01 and 0.001

When analysing the graphics, it is clear to see how the lower learning rate benefits the model. The accuracy's of both the training and validation show similar curves and values, and even the train loss is similar, even though it is somewhat lower with the lower learning rate. The biggest difference is in the validation loss. With the larger learning rate, this curve did not converge to a value, probably because it was "hopping" through the local minimum, due to the larger step. In this experience we concluded that the 0.001 is the best learning rate of all compared.

b) Dropout: For this hyper-parameter, the values 0.25 and 0.35 were experimented with. Similarly as with the learning rate, the charts in Fig. 9 show evolution of the metrics along the epochs.

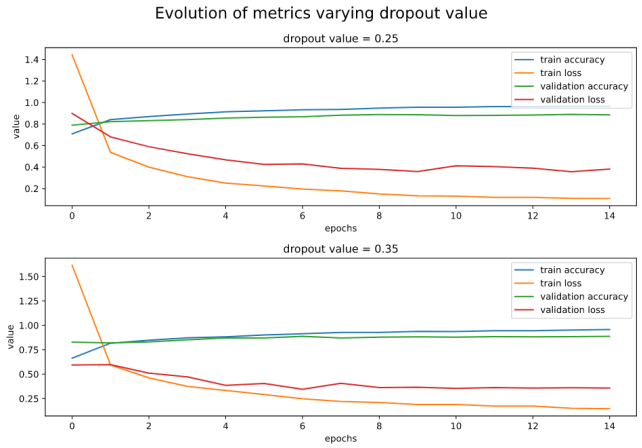

Fig. 9. Complex neural network results for dropout 0.25 and 0.35

In this case, the differences seem to be even less noticeable. The train accuracy with 0.25 dropout is better only by 0.007 in the last epoch, and has also a lower train loss by approximately 0.037. When referring to the validation data, the accuracy is lower by 0.003 than with a 0.35 dropout, and has a larger validation loss by 0.025. There isn't a clear "winner" here, but it will be assumed that 0.35 is a better value because of the slightly better results in the validation data.

V. Model Evaluation

A. Methodology

The train set constitutes 30% of the total data set, as already referred. This corresponds to 2216 images. We used a stratified split, which guarantees that the proportions of number of images in each class is maintained in the test set. The confusion matrix and other metrics presented in this section where made with the model predictions of this test set. At this point the model is already using the most adequate hyper-parameters found in the previous hyper-parameter tuning phase, described in subsection IV-B.

B. Simple Neural Network Evaluation

In Fig. 10 there is a heat map representation of the obtained confusion matrix for the simple neural network model.

It can be observed by this confusion matrix that the model has big problems predicting most of the classes. Instead, it seemed to favour two of the breeds, choosing them most of the time. These two are the Bombay - a cat breed - and the Shiba Inu - a dog breed. An image of each one of these breeds can be seen in Fig. 11.

Other metrics were also calculated and are summarized in the table I.

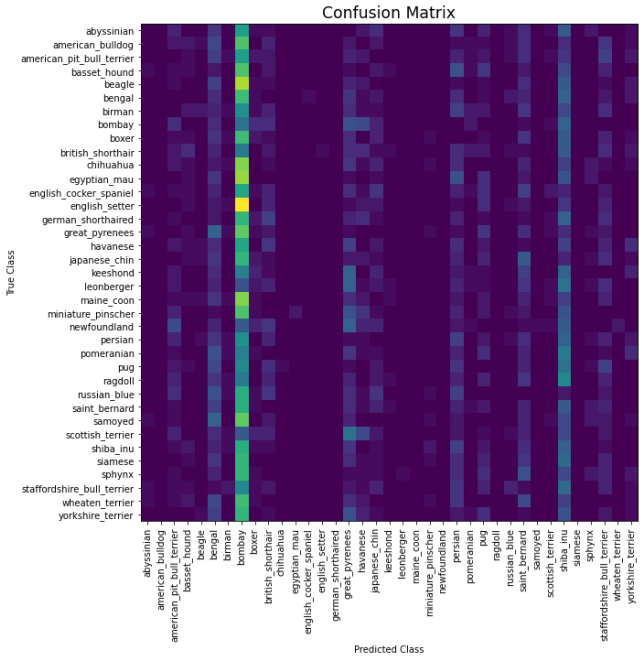

Fig. 10. Heat map representation of confusion matrix obtained from the simple neural network model

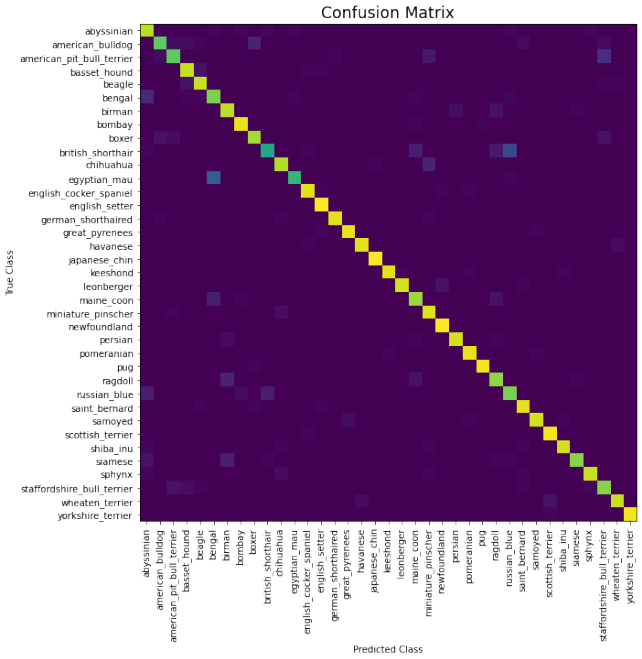

Fig. 12. Heat map representation of confusion matrix obtained from the complex model

Fig. 11. Images of the two most predicted breeds by the model: Bombay and Shiba Inu

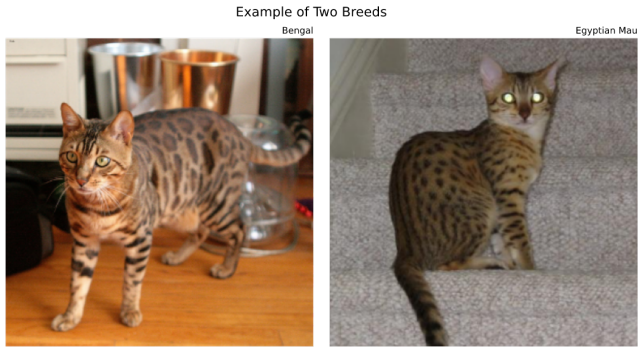

Fig. 13. Images of two similar cat breeds: Bengal and Egyptian Mau

C. Complex Neural Network Evaluation

Fig. 12 is a heat map representation of the obtained confusion matrix.

In this confusion matrix it is possible to see that the model had some trouble predicting the Egyptian Mau cat breed. It often predicted the breed Bengal instead of it. When looking at the images, it is possible to see why this happens, since they are fairly close to one another, even though sometimes they have different fur colors. Two examples can be seen in Fig. 13.

Other metrics were also calculated and are summarized in the table II.

TABLE I
METRIC EVALUATION SIMPLE MODEL

metric	***precision***	***recall***	***f1-score***
macro average	0.03	0.03	0.02
weighted average	0.03	0.03	0.02

TABLE II
METRIC EVALUATION COMPLEX MODEL

metric	***precision***	***recall***	***f1-score***
macro average	0.90	0.90	0.90
weighted average	0.90	0.90	0.90

D. Comparison

We can attest by the results of the two tables already presented (table II and table I) that the use of transfer learning techniques employed in the more complex model lead to better results in all measures when compared to the more simple model, even after choosing the best hyper-parameters. The

difference between both of them is giant, with the first one barely managing to classify an image correctly, and the other doing that almost every time.

VI. Improvements

A. Data Augmentation

Even though the obtained results with the more complex neural network were not bad, it had still room for improvement. One technique often mentioned when the goal is to improve a classifier performance, especially when training data is not a lot, is data augmentation. [1] [2] This method has a special relevance when there is not enough data to train the model, but it also benefits models where this is not an issue. It consists in enlarging the data set with synthetically generated data, in this case, images. These new pictures can be formed by rotating, translating, changing illumination, or transforming in many other ways the original image.

To test if this was advantageous, we added two more layers to the previous model, right after the input layer and before the base InceptionV3 model. These new layers perform transformations on the input data. The first one, **RandomFlip**, rotates horizontally the images; the second, **RandomRotation** rotates them. This last one has a factor of 0.2, which means that the images can be rotated by a random amount in the range [-20% * 2pi, 20% * 2pi].

1) Hyper-parameter Tuning: Similarly as with the complex model without data augmentation, 3-Fold Cross-Validation was performed. The same parameters were tested as before, with the results displayed in figures 14 and 15.

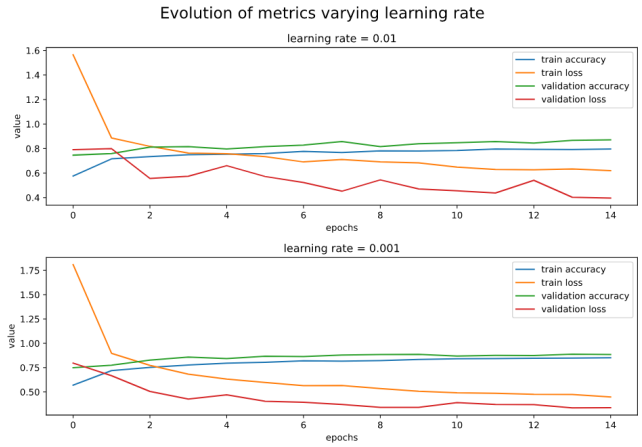

Fig. 14. Neural network with data augmentation results for learning rates 0.01 and 0.001

It is clear to see in the first image (Fig. 14) that, as what happened in the other model, the lower learning rate benefited the results. Both the train and validation loss were able to reach lower values in the same amount of epochs, and the train and validation accuracy are also higher in the second experiment. This proves that the most adequate learning rate is 0.001 out of the two.

The variation of the dropout percentage isn't, once again, so noticeable, with the difference between both of them barely

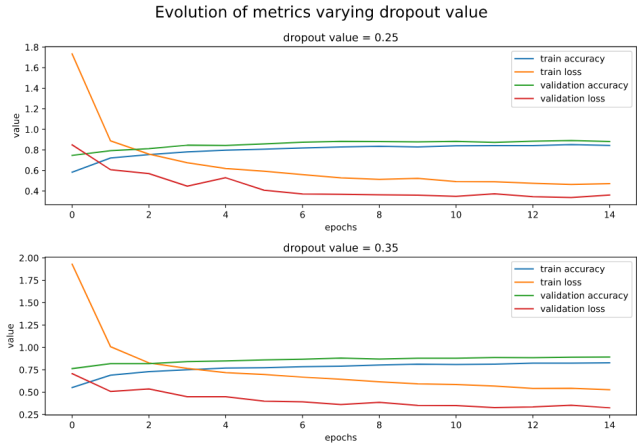

Fig. 15. Neural network with data augmentation results for dropout values 0.25 and 0.35

any. However, the validation accuracy is 0.012 values higher with the 0.35 dropout value than with 0.25, and has a lower validation loss by 0.036, so this was the chosen value.

2) Model Evaluation: Through the heat map representation of the confusion matrix in Fig. 16, and the table III it is clear that the results did not improved when compared to the model without data augmentation.

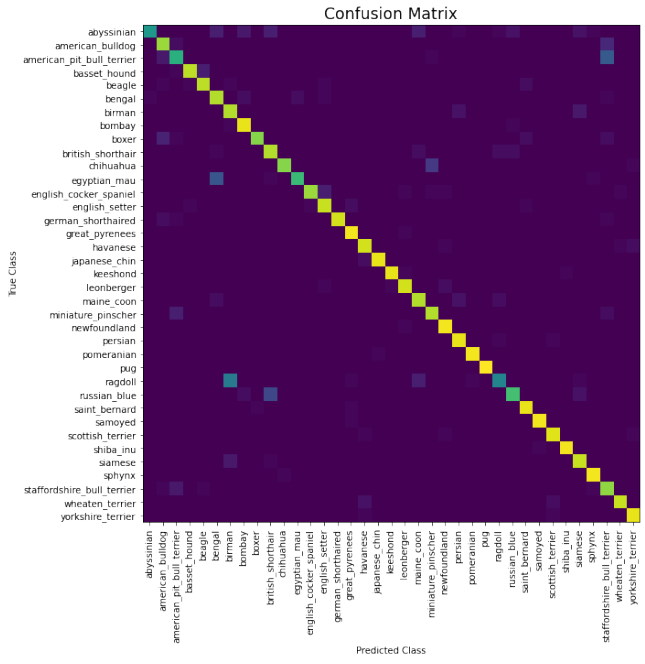

Fig. 16. Confusion Matrix heat map obtained from the complex model using data augmentation

There is a lot more confusion when predicting different classes than there was before, and the metrics show a poorer performance.

TABLE III
METRIC EVALUATION COMPLEX MODEL WITH DATA AUGMENTATION

metric	*precision*	*recall*	*f1-score*
macro average	0.89	0.88	0.88
weighted average	0.89	0.88	0.88

B. Removing Background

Taking advantage of the trimap annotations included in the data set it is possible to remove the background of the images, making them show only the animal. The goal of this is so that the model will only take in account the animal features and not the background, and thus it should produce better previsions. This method has a disadvantage, however. The test data provided to the model should also be without a background, which is not a problem. But if this model were to be used in a real application, there would have to be pre-processing to the pictures input so that they are passed to the model without background. This implies a possibly big overhead.

1) Hyper-parameter Tuning: Charts with the evolution of metrics are presented in Figs. 17 and 18.

In the first one, following the tendencies seen in the previous charts, we can see that the validation loss had trouble stabilizing in the 15 epochs with a learning rate of 0.01, oscillating constantly. Besides, train and validation accuracy are higher in the experience with the largest rate, and both losses are lower. Then we can conclude that the learning rate 0.001 is better than 0.01.

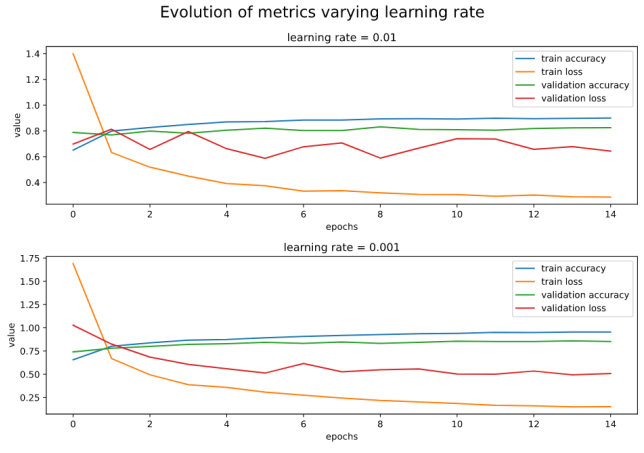

Fig. 17. Neural network model trained with pictures without background results for learning rates 0.01 and 0.001

In the second one, the results are also similar to the other charts of the same type analysed. Varying the dropout value from 0.25 to 0.35 didn't result in many differences. As seen previously, the 0.35 value has a slightly higher accuracy for the validation data and lower validation loss, while with the 0.25 there is higher training accuracy and lower training loss. We decided to go for the 0.35 value, because it seemed to have improved validation results, which probably means that it will predict better results once the training part is over.

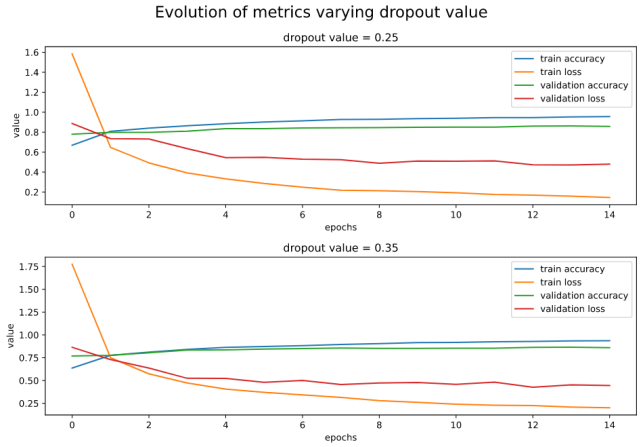

Fig. 18. Neural network model trained with pictures without background results for dropout values 0.25 and 0.35

2) Model Evaluation: Looking at the results presented in Fig. 19 and table IV we can see that this hasn't resulted in any improvement when faced to the original complex model. It even scored worse than the previous experiment with data augmentation, even though this model didn't incorporate any of those techniques.

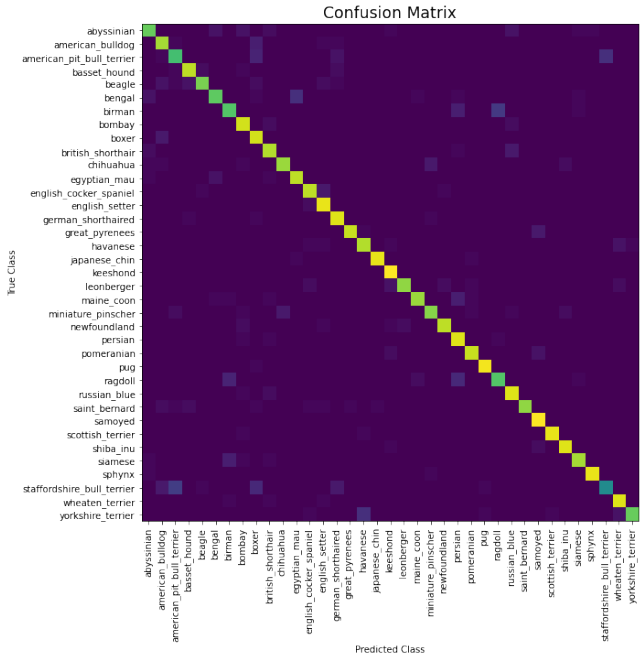

Fig. 19. Confusion Matrix heat map obtained from the model trained with images without background

C. Fine Tuning Base Model

After having found the best possible model configuration and having experimented with data augmentation techniques,

TABLE IV
METRIC EVALUATION MODEL TRAINED ON IMAGES WITHOUT BACKGROUND

metric	*precision*	*recall*	*f1-score*
macro average	0.88	0.87	0.87
weighted average	0.88	0.88	0.87

TABLE V
METRIC EVALUATION COMPLEX MODEL WITH DATA AUGMENTATION AND FINE TUNING OF BASE MODEL

metric	*precision*	*recall*	*f1-score*
macro average	0.92	0.92	0.92
weighted average	0.92	0.92	0.92

came the time to experiment fine tuning the base model InceptionV3 used. Until this moment, we have had the thousands of trainable parameters in the various layers of this network frozen during training, in order not to loose the information it had previously "learned". The only adjustable parameters were the ones of the layers in front of it, the ones we added. This is so that we can use this base model as a framework to understand common features in image classification tasks, like shapes and colors.

In this section we analysed what happened after "unfreezing" the top one third of the InceptionV3 model's layers, i.e. the 103 first layers. After this process, the model is then trained for 10 more epochs, using a very low learning rate of 0.0001. The results can be observed in Fig.20.

1) Model Evaluation: As we can observe, there was a very slight increase in performance for some categories. For example, it is noticeable the fact that the model has an easier time distinguishing Bengal cats from Egyptian Mau cats than before. The results obtained for the performance metrics comply with this slight increase in performance, as we can see in table V.

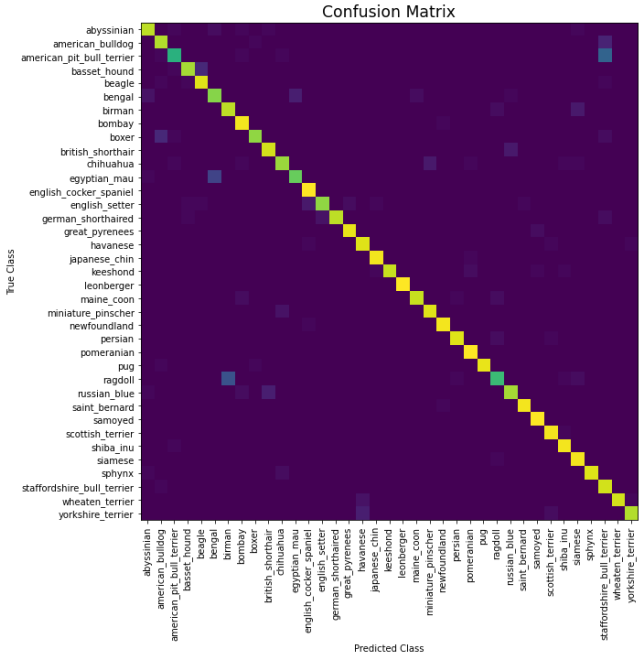

Fig. 20. Confusion Matrix heat map obtained from the complex model using data augmentation and fine tuning of base model

VII. CONCLUSIONS

After experimenting with two very distinct model architectures, tuning our hyper-parameters for each one, trying to take advantage of data augmentation techniques, and even fine-tuning parts of the base model, we were able to extract some conclusions.

Namely, that it is possible to obtain a fairly accurate model for classification of our 37 different classes. We designed a model that scored an average accuracy of 92% for all classes, which is a higher value than some researches whose papers we have read have accomplished in similar problems.

There is no doubt the model is successful in its task, but there is definitely room for improvements. The restrictions we had on time and computational power prevented us to try with a larger set of values for each hyper-parameter, and even restricted on the amount of these parameters we could test. For example, the number of epochs and batch size, and even the hidden layers should have definitely been varied in order to find the best value for our specific problem. It is for this same reason that we were unable to find the best percentage of layers that should be unfrozen while fine tuning when using transfer learning.More data augmentation techniques could also have been experimented with.

A. Division of Labor

Both students worked on the project and analysis in an equal amount.

- Pedro Silva - 50%
- Mariana Santos - 50%

REFERENCES

[1] Z. Ráduly, C. Sulyok, Z. Vadászi, and A. Zölde, "Dog breed identification using deep learning," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, 2018, pp. 000 271–000 276.

[2] P. Borwarnginn, W. Kusakunniran, S. Karnjanapreechakorn, and K. Thongkanchorn, "Knowing your dog breed: Identifying a dog breed with deep learning," *International Journal of Automation and Computing*, vol. 18, no. 1, pp. 45–54, Feb 2021. [Online]. Available: https://doi.org/10.1007/s11633-020-1261-0

[3] H. Zhang, , H. Song, L. Zhu, and S. Deng, "Dog classification into 120 breeds," 2019.