Aves/Mammals EDA and Classification

MOHAMMED GHAITH AL-MAHDAWI

ZYQSYJ

ELTE
EÖTVÖS LORÁND
UNIVERSITY

Data mining and machine learning course

2021. December

Contents

1. Acknowledgment and motivation

I would like to thank the constructors and demonstrators of this course for conveying the material contents smoothly, being there when we needed them, and for their patience, guidance, and supervision.

My motivation for choosing this topic is my interest in Machine learning in general and Computer Vision in particular. I was -and still am- interested in how we can make computers understand their environment and interact with it.

2. Introduction and dataset sources

We, human beings, can classify data easily with our vision and brain. Even animals can classify data to some extent! Biological machines are, indeed, complex creations and reflect a part of the massive universe we live in, see, and interact with. However, computers are man-made machines that do not comprehend their environments as same as biological creatures. Therefore, a simple task as classifying two different types of creatures can be very tough to mimic and simulate on electro-mechanical machines. Which makes image classification an interesting problem to solve; and it is an active field of research.

We gathered images of two distinct vertebrate categories, mammals, and Aves/birds. Then, we train the computer to modularize these categories and predict what is the animal type of any given photo. Around 7,590 Images are gathered from various datasets on Kaggle. Raw pictures data can be found at *[1], [2],[3],[4], and [5]*. And a preprocessed Dataset has been put together at *[6]* Among the chosen 7590 images, 3800 of them are Aves, and 3790 are Mammals.

3. Data preprocessing and exploration

The set of images, of both birds and mammals, are not scaled to a singular global size. Since most of the images were of size 224x224x3, we rescale all the images to the size of 224x224x3. After resizing, to access a more training precision, we scale the image RGB color channels from [0, 255] into [0.0, 1.0]. Normalized data produce more accurate results when passed to the machine learning models. Accuracy increases because of the reduced numerical errors and the ease of calculation, for instance, multiplying 255 with 255 outputs 65,025, while multiplying 1.0 with 1.0 produces 1.0.

Among the preprocessed images, we know that 50% of them belong to one of the categories we are meant to classify for. Thus, we need to split the data into training, and testing evenly to preserve an unbiased trained model. For this task, we chose to split the data of each category into 85% for training and the rest 15% for testing. Hence, we have ~3,230 training images and ~570 testing images per

category. However, for the sake of simplification, we distribute the data as follows.

Sets / category	Training	Testing
Aves/Birds	~3,230 or 3200	~570 or 600
Mammals	~3,230 or 3200	~570 or 600

Represented Images for training and testing of each category. For simplicity, we divide the data into 3200 and 600 for the Scikit-Learn models. On the other hand, we choose a 0.15 validation value for the TensorFlow neural networks models.

After cleaning and preparing the datasets to be processed, now we can explore the data and build models to predict it. Some of the processed data images are shown in figure 1 below. On the left, we see 9 randomly selected Avian type animals, while on the right, we notice another 9 animals selected from the Mammals category.

Figure 1. left sheet comprises Aves/Birds samples, while the right one consists of Mammals samples.

4. Machine Learning

In this section, we compare different models and how well they perform. Beforehand, we know from the data distribution that the base accuracy of guessing randomly any of the categories correctly is 50%. Hence, the models we create should have a higher accuracy rate than guessing randomly.

Since we have a huge dataset, we cannot load all the data into the memory. Thus, we cannot use the naïve classifier tools provided to us through Scikit-Learn, for instance, Logistic Regression; we need to use another approach such as incremental training. Fortunately, Scikit-Learn provides a classifier called SGDClassifier, which uses a similar method to the gradient descent to find the minima of a function. Indeed, TensorFlow provides logistic classifiers as well. However, we use Scikit-Learn models for this task and use TensorFlow later for neural network classifiers. Herefore, we use the SGD classifiers and compare them. Remark, all models are trained with the same batch size for the SGDClassifier specifically, which is 100 images per fit, and with the same default parameters.

4.1 SGD with Logistic Regression Classifier

First, we start with the logistic regression classifier. This classifier outputs a decent result with an accuracy of ~60%, and an AUC of ~61%. The model outputs the results which you see in figure 2 below. Finally, we observe that this model is better in classifying Mammals than in classifying Aves. Consequently, this model can be chosen when classifying mammals is more

important than classifying birds since the falsie prediction is lower for mammals.

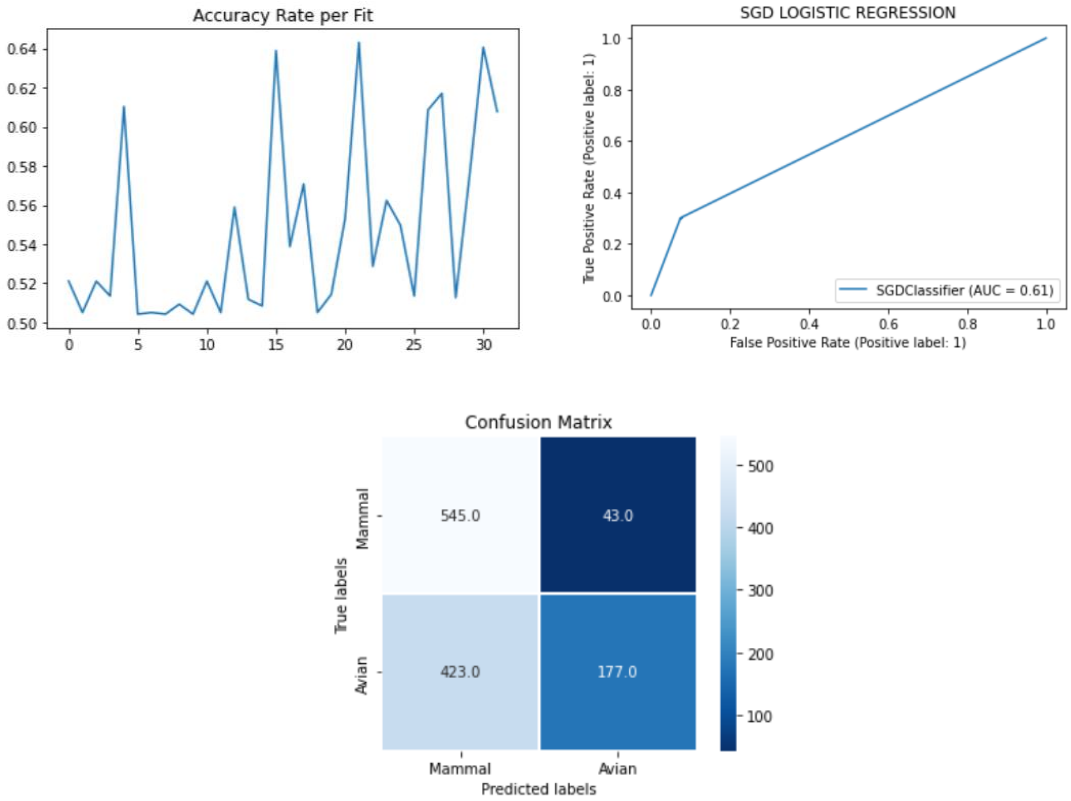


Figure 2. The graph on the left shows the model accuracy during training, while the one on the right shows the ROC and AUC of the SGD logistic regression classifier. The image below shows the confusion matrix of the training.

4.2 SGD with Support Vector Machine Classifier

In this model, we try to compare the results while using the same approach and parameters as the Logistic Regression. However, the SVM classifier did not do well in classifying the data. The model achieved an accuracy rate of ~50% and scored an AUC of ~50.4%. Thus, the results illustrate that this model is not suitable for solving the task we are addressing in this EDA.

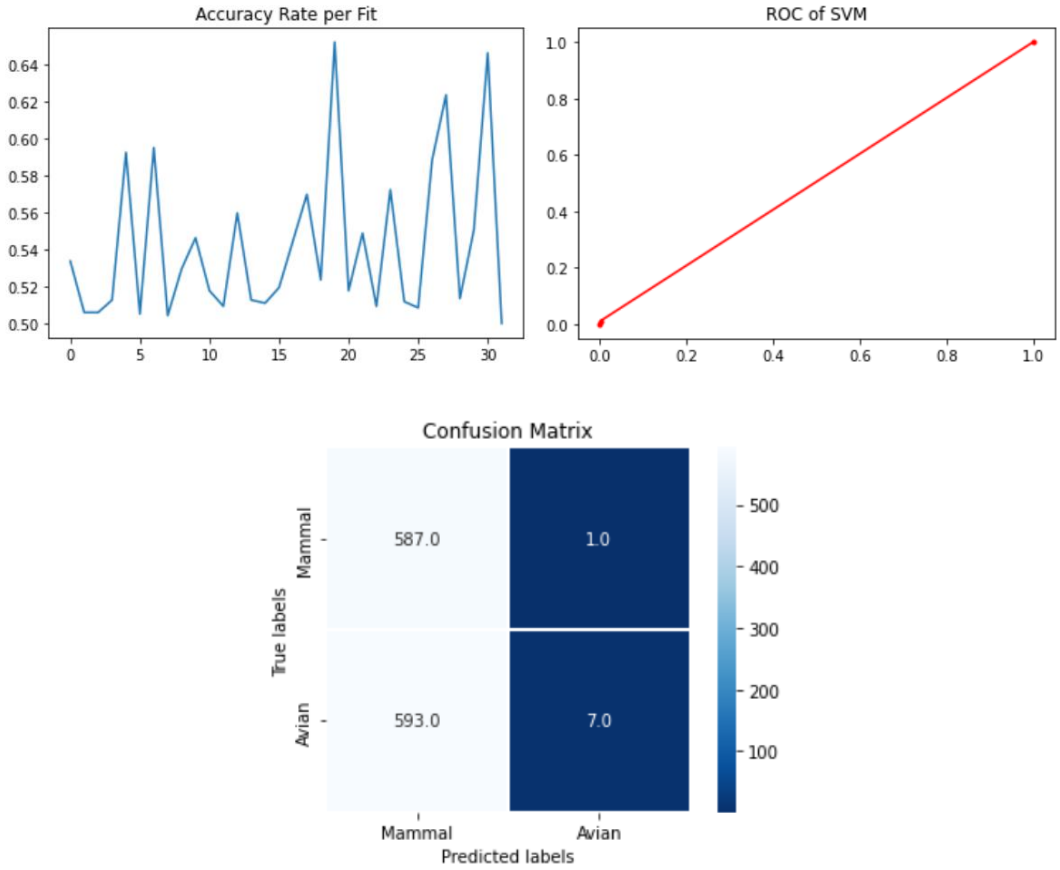


Figure 3 The graph on the left shows the model accuracy during training, while the one on the right shows the ROC of the SGD Support Vector Machine classifier. The image below shows the confusion matrix of the training.

4.3 SGD Conclusion and Proposals

After trying a variety of different loss functions for the SGDClassifier, we conclude that the results are very similar to the SGD SVM results. Thus, we are not including all the loss functions and their results. Other models implementations in Scikit-Learn, such as KNN and K-Mean clustering do not support data streaming. A remedy trick would be to re-implement them and make them suitable for data streaming; however, we are not doing

that in this presentation. Furthermore, we can reduce the image dimensions in the preprocessing stage into its feature points, or areas of interest, by using either ORB or SIFT features detector. After that, we can cluster the images data regarding the feature points. However, this is far away from the course material and this EDA. Hereafter, we focus on discovering Neural Networks to help us classify better.

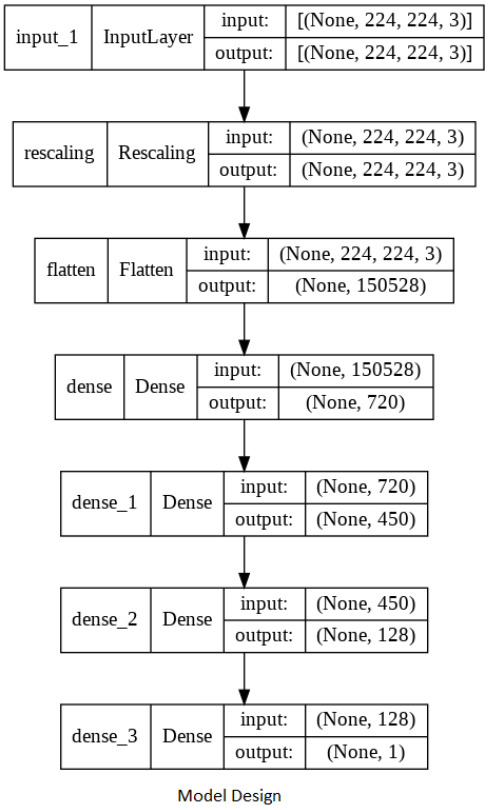
5. Neural Networks

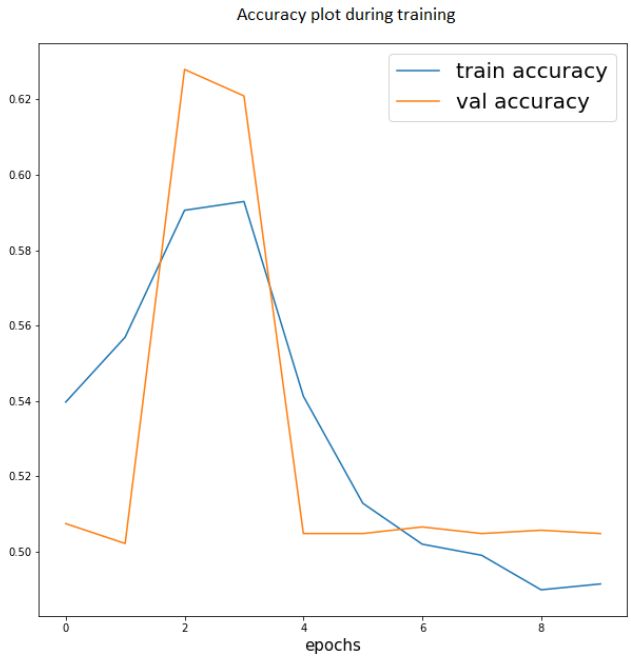
For this task, we choose to use TensorFlow since it provides several models and processing tools. In this section, we discover a naïve neural networks implementation, then we explore some convolutional neural networks to get a more precise prediction.

5.1 Base Neural Network Model

We design a simple neural network model to act as a base model in our exploration. We also introduce some image augmentation to increase the reliability of the model prediction.

The model design, which contains the processing stages, is shown in figure 4. We run the Neural Network model 10 times on the Dataset. And observe the result at each iteration. The accuracy peak is at the second iteration with ~60%. Therefore, we can notice that the best epoch with the lowest error and loss, and with the highest accuracy is the second one.





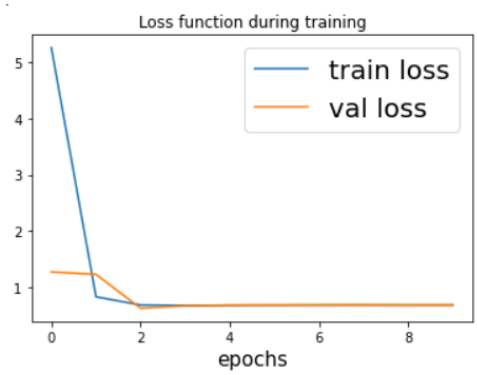


Figure 4. Our base neural network model. The image on the top left shows the model design, while the other images show the accuracy and loss results.

5.2 Base Convolutional Neural Network (CNN)

Here we try to create a base CNN model. A convolutional network is the best fit for images classification since it filters it first, then uses a neural network to classify it. We fit the model with 10 epochs again. However, this time we reach up to an accuracy of ~85% in the 7th epoch, then we start getting into an overfitting status. This model achieved a much better result in comparison to the Base Neural Network model we tested earlier. Overfitting is a result of reading noise in the dataset, and it can be addressed by feeding the model with a bigger dataset or adding a variant of Image-Augmentations. For this case, we only did basic image augmenting. The results of our model and its design are attached in figure 5.

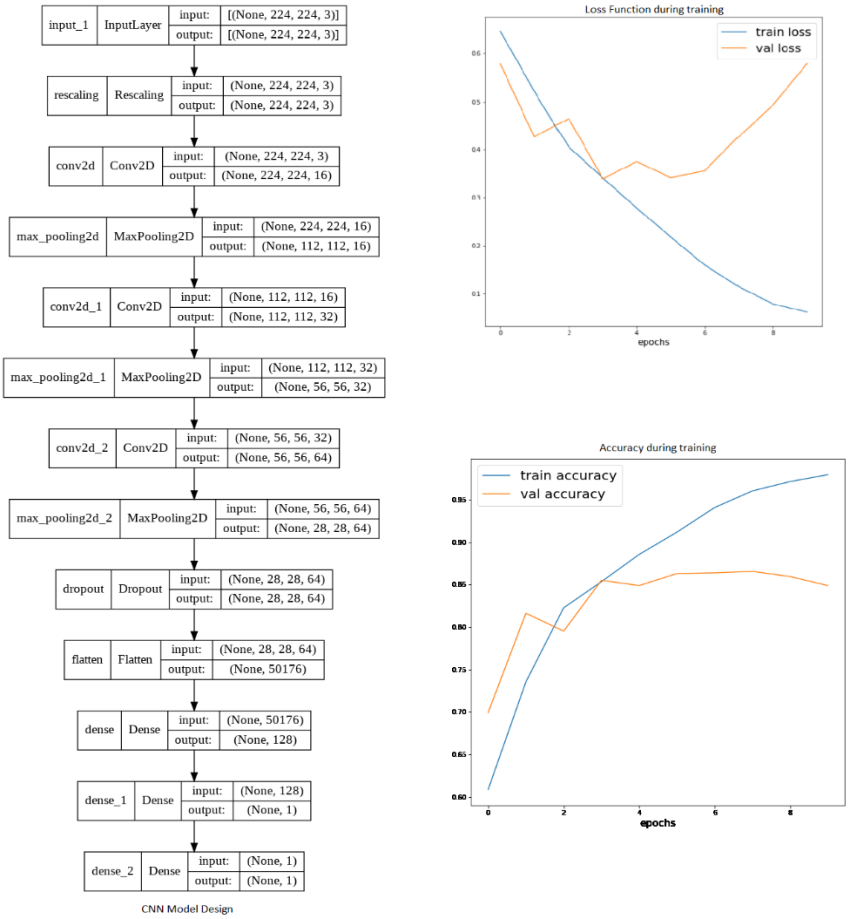


Figure 5 explains the settings of the Convolutional Neural Network model. For guidance on how to interpret the diagram, see figure 3.

5.3 Residual Convolutional Neural Network

This model shows a much better result than the previous two models. The zenith of the accuracy is reached in the 9th iteration with ~90%. This model did not fall under the overfitting during the training, unlike the previous models. The minimum loss during training is ~0.24 with a mean squared error of ~0.075. This model is far superior in classifying the Aves from mammals. The training observations are plotted in figure 6, while the model design is in figure 7.

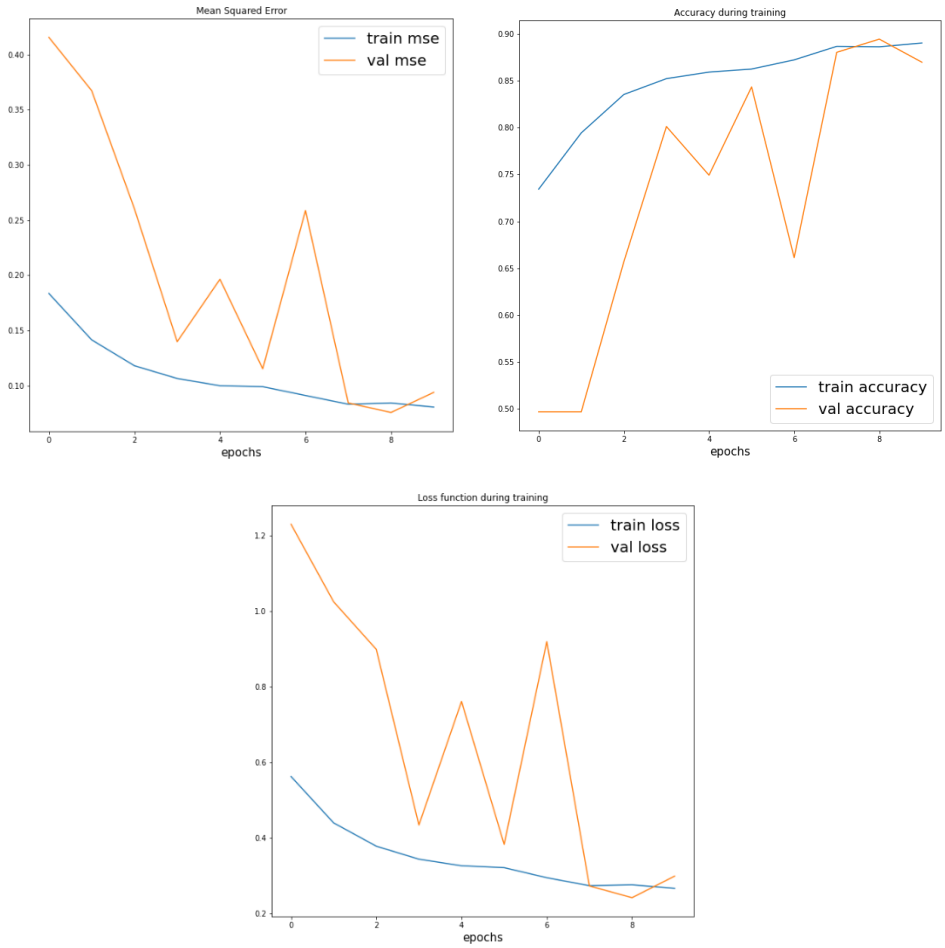


Figure 6 shows the model behavior during training.

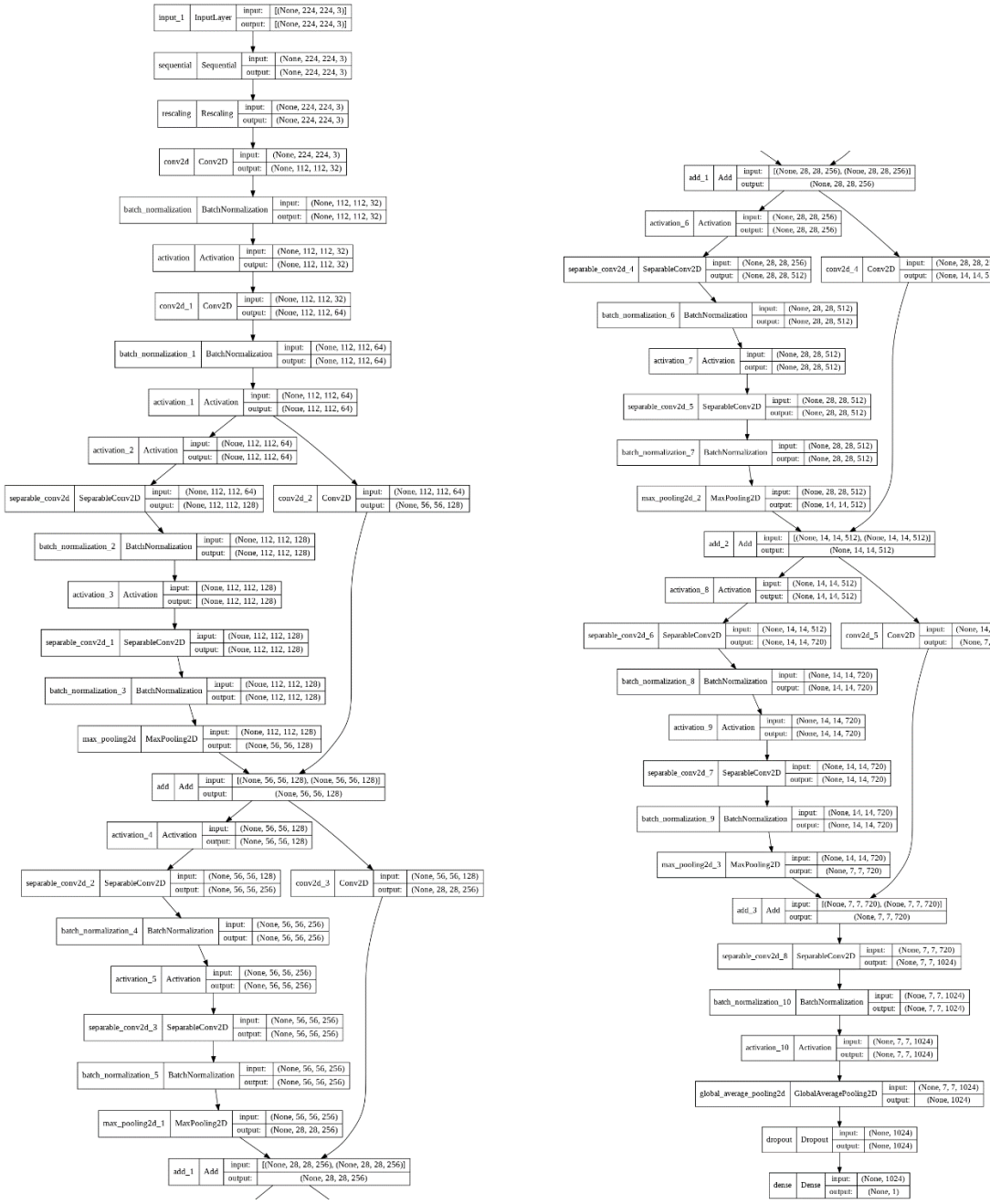


Figure 7 shows the residual neural network used in this classifier. The last most node of the left diagram is the first node in the right diagram.

5.3 Neural Networks Conclusion

After trying several neural network models, we observe that the best model is the model which accounted for residuals. The Residual Neural Network outputs a very good prediction on the given dataset. Although, there are a lot of different and distinct models that we have not covered. However, for this classifying problem, we are satisfied with a prediction accuracy of ~90%. Therefore, no need to explore more image classification models.

6. Summary and Conclusion

In summary, we address an image classification problem, which is classifying the image into two categories, which are Aves and Mammals. After exploring a diverse set of models and approaches, we settled on using a Residual Convolutional Neural Network model. The model design and results are attached in this document, in section 5.3. In conclusion, the model presents an accuracy of 90% in predicting the image category. Thus, it is the best model among the models we covered so far.

7. References

[1] https://www.kaggle.com/navidre/alberta-wildlife-dataset

[2] https://www.kaggle.com/gpiosenka/100-bird-species?select=train

[3] https://www.kaggle.com/anirudhg15/mammals-classification

[4] https://www.kaggle.com/edenbenson/north-american-predator-dataset

[5] https://www.kaggle.com/iamsouravbanerjee/animal-image-dataset-90-[6] different-animals

[6] https://github.com/Mohido/Aves_Mammals_EDA