Sergey Volchkov

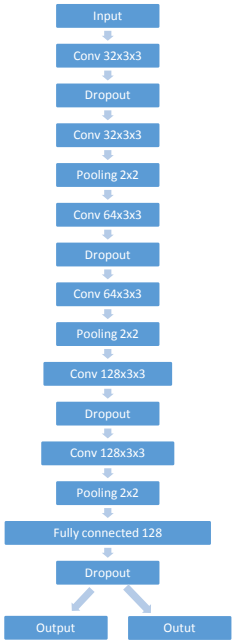
VISUAL RECOGNITION OF IMAGES FROM SNAPSHOT SERENGETI DATASET USING DEEP LEARNING METHODS

FINAL REPORT

Approach used and differences from the original report

Upon discussion with the project supervisors, instead of the original approach involving transfer learning, it was decided to instead build a small network with 5-8 convolutional layers and train it from scratch. The decision was made in view of hardware limitations as it was expected that training a large network – even if training involved only part of the network - might require significant processing power. It was also estimated that even a small network can achieve an accuracy of about 90% on the animal detection task.

The baseline architecture of the network is shown in Fig. 1. It includes two layers with 32 filters of size 3x3, two layers with 64 filters of size 3x3 and two layers with 128 filters of size 3x3. These layers are interspersed by two other layer types



- **Dropout** is a layer type that is use to temporarily freeze some of the neurons in the previous layer in order to exclude them from the learning process. The purpose of using dropout layers is to prevent situations where some neurons get an excessive weight during training which might result in overfitting.
- **Pooling** is a layer type that is used to gradually decrease the input size. In the present case, we are using 2x2 max pooling layers which retain the maximum value of the neurons contained in a 2x2 window from the previous layer.

The stack of convolutional, dropout and max pooling layers is followed by a fully-connected layer that connects all the neurons from the previous layer with each other. This is the layer with the greatest number of parameters within the network. Finally, at the end we have two output neurons that estimate the probability of the presence of animals in the image.

Fig. 1 Baseline network architecture

The network was expected to solve an object detection task as a binary classification task i.e. it was to produce one of the two possible outputs: “animal” or “no animal”. The reason why the final output layer uses two neurons is that, by convention, the number of neurons in the final output layer corresponds to the number of classes which simplifies some data preparation and computation tasks. The prediction accuracy was determined based on the number of correct classifications.

It should be noted that finding an optimal network architecture is a bit of a dark art as there are multiple parameters involved and there are no hard-and-fast rules to prescribe the optimal settings for each particular dataset. Therefore I used the above network as a benchmark and then tried fine-tuning various parameters to improve the network performance. The reason for selecting the above architecture as baseline was that I previously used a similar network for another visual recognition task and achieved an accuracy of over 90% on it. I also knew that this architecture can be handled by my home machine without getting too many out-of-memory errors and completing one epoch of training within a reasonable time.

Some further implementation details are given in the attached Jupyter notebook.

Performance benchmark

As I mentioned in the initial report, in most cases it is optimal to compare performance of the network with that of humans on the same task. However, in the present case it is only possible to obtain a rough estimate for the human (volunteer) performance. The dataset is accompanied by a spreadsheet containing data on species identification performed by experts on the SS dataset ("gold standard data"). At the same time, the spreadsheet only contains data on 4,149 capture events. It appears that the spreadsheet only includes data on those events where animals were found to be present in the image. I calculated the percentage of images that had been identified by exports as containing animals but were identified as containing no animals by the volunteers. It allowed me to calculate the accuracy figure of 98.55% for the volunteer assessment. However, the actual accuracy figure should be somewhat lower as in some cases volunteers might have mistakenly identified some images as containing animals while experts found no animals in those images.

The following factors are expected to negatively affect network performance vs human performance.

- Images are reduced to 96x96 in size in the pre-processing stage. This is done because of hardware limitations. On same images, animals are far away from the camera so when the image is downsized, they might become too small and therefore not recognisable by the network.
- Humans are likely to be shown several images from the same camera in succession. In this case, the appearance of new objects in the camera's field of vision is immediately obvious. The network might only see a single image from a camera in some cases as we're using less than 1% of the dataset.

Because the gold standard spreadsheet only contains the ground truths for slightly more than 4,000 images, I decided to use the consensus values from the volunteer assessments as the ground-truth data labels. Because volunteers' accuracy on the animal detection task was nearly 100% compared with the gold-standard data, there should not be much difference with using expert data as labels. In some cases the ground truth labels appeared to be incorrect (some examples are given below) but that is the case with most datasets and the network is expected to be able to handle some small amount of noise caused by incorrect data labels.

Collecting and preparing data

- Images were randomly selected from the SS dataset.
- Night-time images were filtered out. For images containing animals, the time when an image was taken could be reliably determined as it was specified in one of the spreadsheets accompanying the SS dataset. For images, not containing animals, it had to be done by

analysing the image histograms. This latter method is not very reliable so some night-time images not containing any animals could have been retained in the dataset thus introducing some bias. It is not expected to be very significant, though.

- A 100-pixel-high strip was removed from the bottom of each image. On most images, that area was occupied by a white strip containing the image name and the date and time when the image was taken.
- Images were resized to 96 x 96 pixels.
- Special care was taken to ensure that the ratio of images containing/ not containing animals was approximately 50/50.
- The total number of images used for the task was originally 5,568 and was later increased to 11,171 images of which 5,602 contained animals.
- 90% of the images was randomly assigned to the training dataset and the remaining 10% to the validation dataset.

Testing results

To estimate network performance, I used training/ test accuracy charts which allowed me to estimate

- Average test accuracy after the network's performance is stabilised and test accuracy fluctuations are not too wild
- Amount of discrepancy between the train and test prediction accuracy – too much discrepancy indicates overfitting
- Any continuing trends – i.e. if the test accuracy appears to still be increasing after initial 40-50 epochs of training, it might be worthwhile to save the network parameters and continue training it for a few more epochs until there's no more upward trend

After training the baseline network for 40 epochs (each epoch involved the network going through all images in the training and the test dataset), I obtained the training accuracy/ validation accuracy results as shown in Fig. 2.

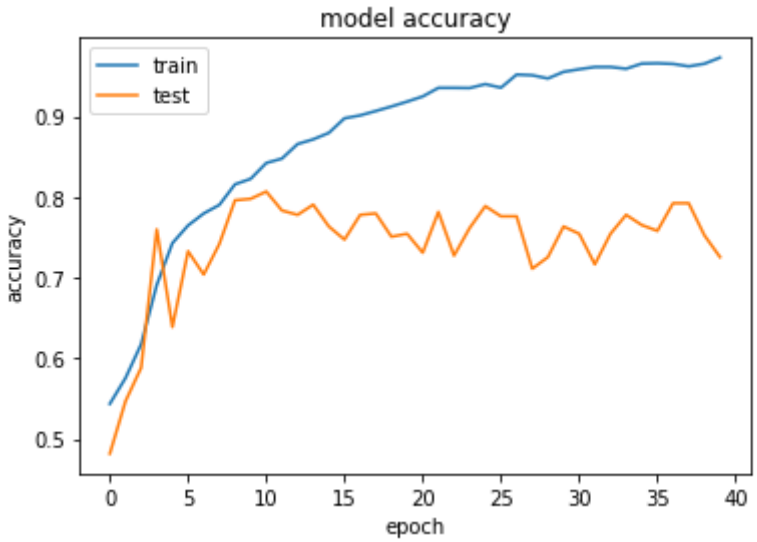


Fig. 2 Train/test accuracy for baseline configuration

We can see from the graph that

- Prediction accuracy on the validation dataset was around 75%
- There was essentially no improvement in the test accuracy (the orange line marked as 'test' on the graph) after approximately epoch 8-9 although the training accuracy keeps improving until epoch 40 and reaches nearly 100%. In other words, the model starts exhibiting clear signs of overfitting.

After obtaining the baseline result, I used several adjustments aimed at improving prediction accuracy and decreasing the amount of overfitting.

1) **Randomly shuffling input data.** Initially, the project implementation involved loading the image file names from the image directory in the alphabetical order and then splitting the resulting list so that the first 90% of the images were assigned to the training set and the remaining 10% to the test set. That is not the optimal approach as the image file names reflect the capture event codes so that images from the same capture event or from similar capture events will most likely be all assigned to either the train set or the test set. If we shuffle the images before assigning them to either the train/test sets, we could hope to get better results.

After randomising the train and the test sets, I ended up with the model accuracy graph shown in Fig.3 after 40 epochs of training. While there's still about a 10% difference between the train and the test accuracy, the difference is much less pronounced (about 10% vs 20% for unshuffled data), the final validation accuracy is much higher (86.82% vs 72.5%) and there are less pronounced fluctuations in the validation accuracy graph:

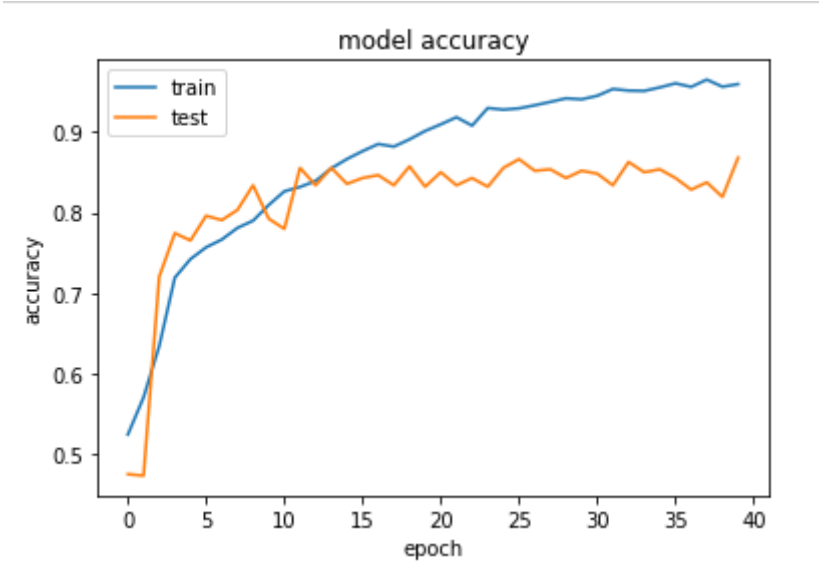


Fig 3. Randomised train/test split

2) **Increasing the amount of data available for training and testing.** The original number of images used by the project was 5,538. The images were split into training and test sets using 9:1 ratio, i.e. 90% images were assigned to the training set and the remaining 10% to the test

set. Since there are many additional images available in the SS dataset, we can easily increase the amount of training data without resorting to data augmentation.

The increase in the amount of data resulted in an increase in the average test prediction accuracy which went up from 84% to about 87%. Overfitting was still rather pronounced, though.

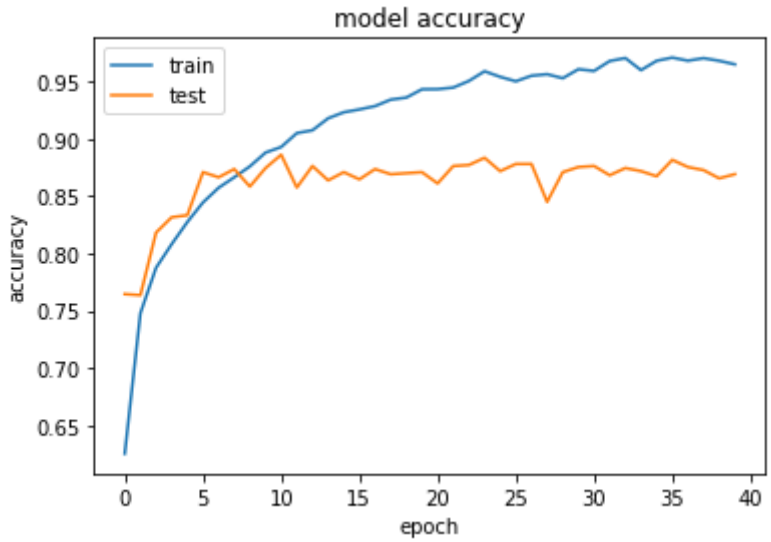


Fig. 4 Increasing the amount of data by approx. 50%

3) **Changing the network architecture.** Here, I have tried two mutually exclusive approaches.
   - **Increasing the network depth.** This involved adding two convolutional layers of 160 filters each. It did not produce any improvement in the test prediction accuracy which was, in fact, down to about 84%.

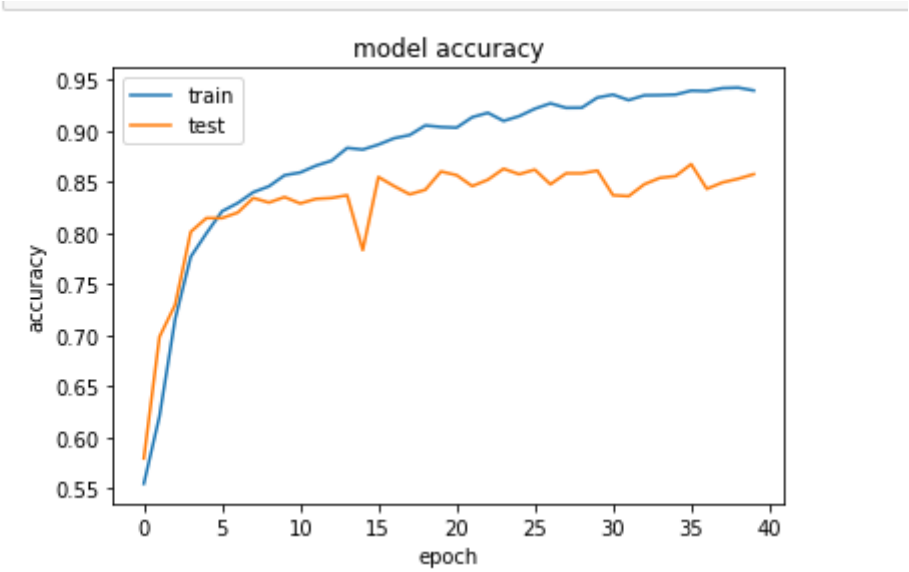


Fig. 5 Increasing the number of convolutional layers from 6 to 8

   - **Decreasing the network depth**. This involved removing one 32-filter and one 64-filter layers. The test prediction accuracy was again about 87% with the final accuracy of

88.1%. In addition, the training time per epoch was decreased by about 15%. This is an important factor as training neural networks is rather time-consuming. Therefore I ran my further tests using primarily this simplified network architecture.

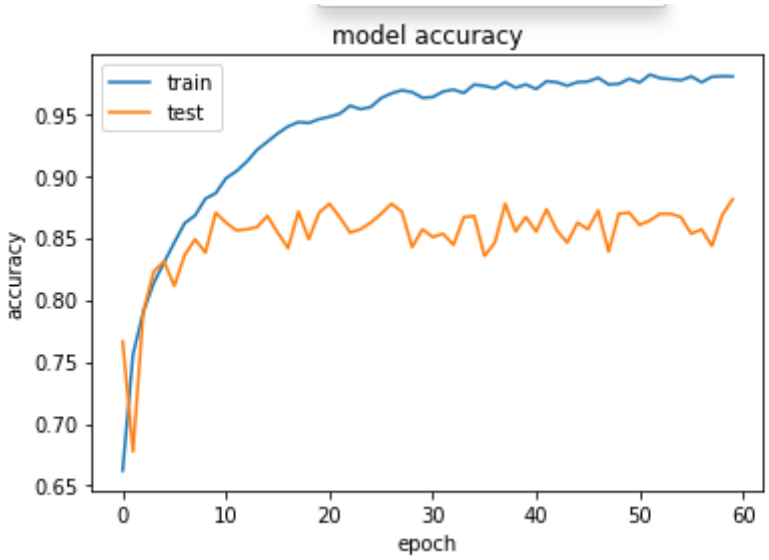


Fig. 6 Decreasing the number of convolutional layers from 6 to 4

The time savings achieved allowed me to increase the number of epochs I trained the network for to 60, however as can be seen from the above graph, there was not much improvement in the test accuracy.

4) **Increasing dropout and using regularisation.** The dropout technique was already described above. I tested the hypothesis whether increasing the dropout rate might result in less overfitting but it did not seem to be the case. **Regularisation** is yet another technique used to reduce overfitting whereby the network imposes a "penalty" for neuron weights that are too large. Thus no neuron is given too much "importance" in the network, and the network detects a more balanced set of features. However, this technique did not seem to produce an improvement in the test accuracy either.

5) **Data augmentation.** Data augmentation is a technique that involves making small changes to the images in the dataset to generate more data. It is often used when the original dataset is too small and additional data is hard to come by. If we rotate an image of a cat by 5 degrees, we obtain essentially a new image, however, because it is not very different from the original one, it is arguably not as good as using a completely new image.

   One reason why I did not use this technique for the present task from the start was due to the specific nature of the SS dataset. Animals where they are present in an image are not centred in the image so it might happen that as a result of an image transformation – such as rotating or shifting the image, the animal might no longer be in the image which will make the label incorrect and increase the amount of "noise" in the dataset. The only safe transformation in

this situation appeared to be horizontal flip. However, in the end I decided to give the technique a try using horizontal flip and some moderate amounts of shearing, shifting and rotation to produce augmented images.

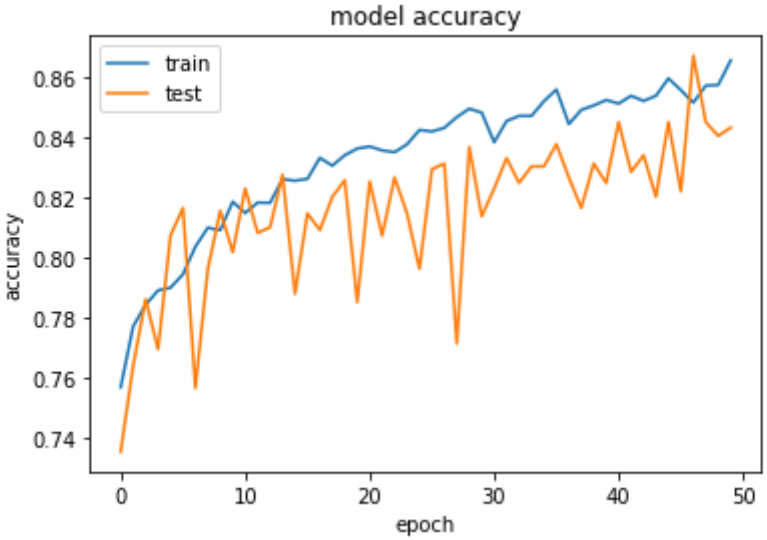


Fig. 7 Data augmentation. Epochs 0-50

As can be seen from the above graph, using data augmentation produces considerably less overfitting. The difference between training and test accuracy in most epochs did not exceed 1-2%. While the average test accuracy for the last 10 epochs appeared to be around 83%, there was still a noticeable upward trend so I decided to test the network for another 50 epochs obtaining the following results:

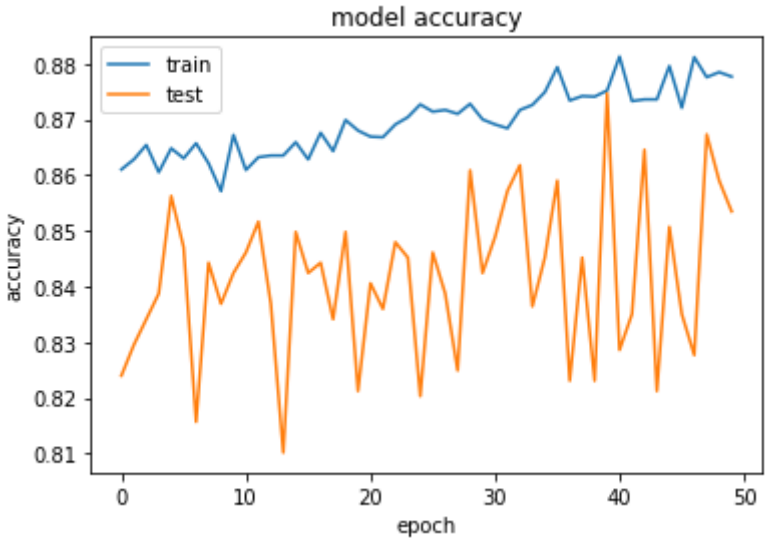


Fig. 8 Data augmentation. Epochs 50-100

And then for 50 more epochs:

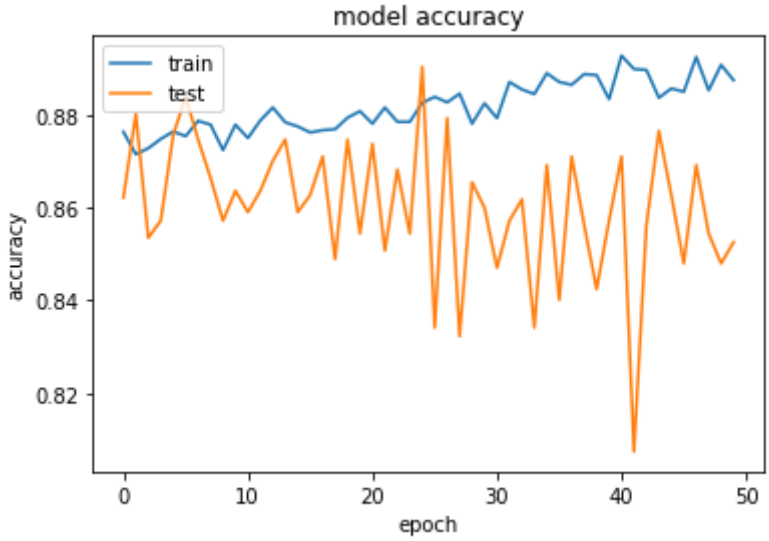


Fig. 9. Data augmentation. Epochs 100-150

From the above images, it appears that the test accuracy achieved its maximum level around approximately epoch 125 (achieving a maximum accuracy of over 89%) and then there was a slight drop so further training of the network does not appear to be useful.

Analysis

I analysed network predictions by visually inspecting both correctly and incorrectly classified images.

Correct predictions

`ASG00087oz, Predicted=Animal Correct=Animal`

ASG0007evs, Predicted=No animal Correct=No animal

ASG000883x, Predicted=Animal Correct=Animal

ASG000tljd, Predicted=Animal Correct=Animal

From the analysis of the correctly classified images, it can be hypothesised that the network can most easily classify an image as containing an animal where the animal is in the foreground and has a coat of either a solid colour distinct from the surroundings (wildebeest, elephant) or having a distinct pattern (zebra).

Incorrect predictions

Examples of incorrect classification are often more useful than correctly classified as they can shed more light on the inner workings of the network.

ASG0003q0x, Predicted=Animal Correct=No animal

In the case of the above image, it is likely that the dark object in the foreground is part of an animal such as an ear or a horn. We need to remember that the primary task of the volunteers working on the SS dataset was to classify the animal species and in the cases where reliable identification was not possible, the image could have been classified as empty.

ASG0006r8o, Predicted=Animal Correct=No animal

Possibly, the clump of trees in the background was misidentified as an animal.

ASG0011f0u, Predicted=No animal Correct=Animal

There is only the head of an animal visible on the right side of the image.

ASG000a809, Predicted=No animal Correct=Animal (predicted correctly by some configurations)

It is rather hard to find the animal in the above image. The dark mass in the bottom right corner could be an animal or a mound of earth. However, if the volunteers were viewing several images from the same camera in succession, it would be easy for them to notice new objects in the camera's field of vision and thus distinguish between animals and immovable objects.

ASG0011f64, Predicted=No animal Correct=Animal

Animals are too far away from the camera, they are unlikely to be seen in the downsized image.

ASG0011hon, Predicted=No animal Correct=Animal

ASG0011hon, 96 x 96 version

The birds in the above image are rather small, and are no more than 2-3 pixels in the downsized image.

Predicted=No animal Correct=Animal

ASG0009nwl, Predicted=No animal Correct=Animal

ASG00007kc, Predicted=No animal Correct=Animal

In the above three images, animals blend well into surroundings which might be the reason why the network failed to detect them.

Conclusions

The tested network configurations appeared to achieve an average test accuracy of about 85%, with some configurations achieving the maximum accuracy of 88-89%. This is significantly better than random prediction accuracy of 50% but quite possibly well below human prediction accuracy. The techniques that appeared to have a positive effect on the network performance included the following:

- Randomising the train/ test split to make sure that the inter-sample variability between the train and test data samples
- Adding more data
- Decreasing the network size – it might be a good idea to always start with a small network gradually adding layers which might save processing time. Also, large networks are prone to overfitting when used on small datasets as they are capable of memorising the dataset without extracting useful features
- Data augmentation – while this technique did not seem to produce a noticeable increase in the overall prediction accuracy, it reduces the amount of overfitting and allowed to train the network for longer. The greatest maximum prediction accuracy of over 89% was achieved using data augmentation.

Based on the analysis of incorrect classification examples, the following measures might be beneficial for further increasing the prediction accuracy of the network:

- Increasing the number of training images of particular species that blend well into surroundings such as impalas and hyenas
- Increasing the image size to improve the detection of animals that are far away from the camera
- Using an ensemble of network configurations and averaging their predictions. It has been noticed that some images are correctly classified by some configurations but not by others so aggregating predictions might improve performance.