Politecnico di Milano

Image Analysis and Computer Vision

Homework Report

Course teacher
Professor Vincenzo Caglioti

Project supervisors
Professor Giacomo Boracchi
Professor Diego Carrera

Perugini Alex 876359
Re Marco 873564
Scotti Vincenzo 875505

Contents

Chapter 1

Problem formulation

The proposed project consisted in the implementation of a convolutional neural network for the classification of sea lions using pictures extracted from aerial images.
The idea of the project is based on a Kaggle competition featured in 2017 (`https://www.kaggle.com/c/noaa-fisheries-steller-sea-lion-population-count`) where the objective was to provide a sea lions population count using fully convolutional neural network to analyze the aerial images taken by drones. Moreover it was required also to distinguish five classes among sea lions based on age and sex:

- adult male
- adult female
- subadult male
- juvenile
- puppy

The original aim of this competition was to automatize work done by biologist to keep track of the sea lions population. This manual work takes up to four months to count sea lions from those images. Due to this long time needed, automatizing this work would allow biologist to focus more on sea lions problems, rather on this counting task.
The requirement assigned to our group for the project was to provide a classifier that was able to distinguish only 2 classes:

- sea lion
- background

without any distinction between the sea lions subclasses.
To accomplish to the task we used the dataset provided by kaggle that included,

for each image, the ground truth expressed as a point centered on each sea lion with different colors for the different classes.

Figure 1.1: Sample image from the dataset

Chapter 2

State of the art

Image classification is a fundamental problem in computer vision since it forms the basis for other computer vision tasks. Although the task can be considered second nature for humans, it is much more challenging for an automated system. Traditionally handcrafted features were first extracted from images using feature descriptors, and these served as input to a trainable classifier. In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to overcome these challenges. Among them, CNNs have become the leading architecture for most image recognition, classification, and detection tasks.

The most significant advance, which has captured intense interest in DCNNs, especially for image classification tasks, was achieved in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. Since then, DCNNs have dominated subsequent versions of the ILSVRC and, more specifically, its image classification component.

Chapter 3

Solution Approach

Images in the provided dataset were taken from drones, thus they are very large and contains a lot of sea lions and different background areas. Due to the impossibility of using the whole images to train the network, the first step before was to create a suitable training dataset. To do this, we extracted patches from the provided dataset.
After having extracted them, we decided to apply data augmentation to obtain a more robust classifier. This allowed us to increase a lot the dimension of the original dataset, in particular for what concerns sea lions patches. In fact, due to the nature of the images, they contained much more background patches than sea lions ones. By data augmentation we were able to overcome, partially, this unbalance and to obtain better performance during testing phase.
Once performances over patches were satisfying, we moved to an higher level, modifying the network to take as input the whole image and providing an heatmap of it. This gives the possibility, given an image, to see where and how sea lions are distributed in the environment.

Implementation

3.1 Dataset creation

The first thing to do with images provided by Kaggle is to divide them in train and test set, in particular the first 750 images were used for training and the remaining ones (from 751 to 947) for testing.
The extraction of patches from Kaggle images has been done performing the absolute difference between the original image and its corresponding one with colored dots on the sea lions, in this way it's possible to gather from each image coordinates and class of all the sea lions.

(a) Original image (b) Dotted image

Figure 3.1: Comparison of train and train-dotted images provided by kaggle

Given these coordinates it's easy to cut from the original images a 96×96 area around them and save the patches labeling them as 'sea lion'.
To extract the background we used a sliding window of size 96×96 over the image and cut all the patches which were not intersecting with a sea lion patch and save them with label 'background'.

3.2 Model

The DCNN used to classify patches has input size of $96 \times 96 \times 3$ and output size of 2, where each output corresponds to the probability of belonging to a class. The network is made up of 9 layers

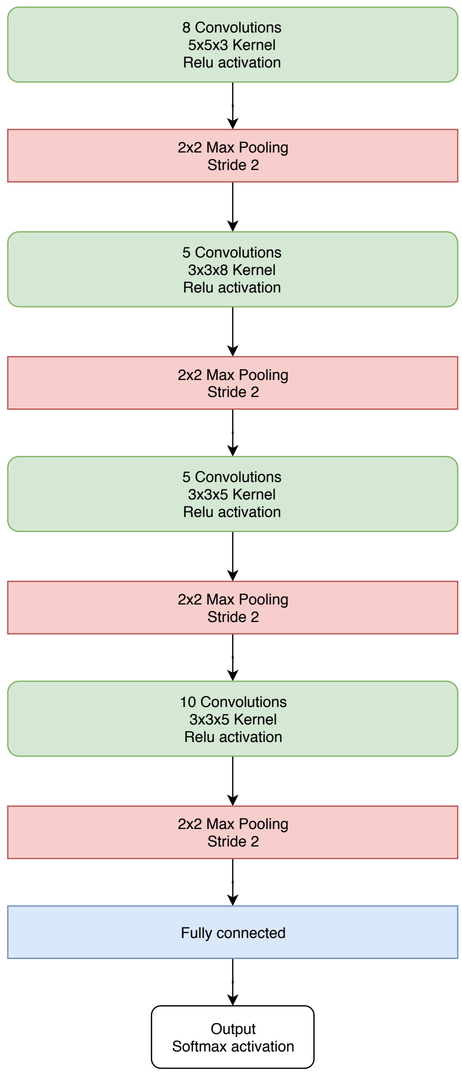


Figure 3.2: CNN model

Other characteristics of the model are:

- optimizer: Adam, a stochastic optimization method which uses gradients and second moment gradients to perform parameter update

- loss function: binary cross entropy, error function based on class probabilities
- metric: accuracy, evaluation of the model based on how much the classification is accurate

3.3 Data augmentation

As explained before, to increase the dimension of the dataset we applied data augmentation to the sea lions patches. In particular, what we applied is a random combination of four transformation:

- Rotation of a random degree in the range 0 to 360 degrees
- Flipping both vertically and horizontally
- Shifting of a maximum of 10 pixels, both vertically and horizontally
- Zooming of a maximum of 20 pixels

3.4 Semantic segmentation

As soon as the CNN building and training phases ended we proceeded to extract the fully convolutional model. To do so we reshaped the fully connected final layer to another convolutional layer maintaining the same connections and related weights of the original network, the new final convolutional layer used 2 filters (one for each label) and a 4×4 kernel, with a softmax activation function.
Since we dropped the constraints on the input shape it was possible to feed an entire image to the network that produced as output two heatmaps that highlighted respectively the pixels belonging to the sea lion and the background classes according to the probabilities predicted by the classifier.
It is important to stress out that we didn't use any shift-and-stitch or upsampling via deconvolution to yield the dense predictions, instead we applied a simple interpolation to the results.

Chapter 4

Experimental activity and results

4.1 Developement tools

In order to perform the tasks required by the project we used jupyter notebooks that allowed to write python code and to execute it section by section, in this way we were able to monitor the results as soon as each step was completed.
Thanks to the choice of python we managed to use Keras, an high-level neural networks API, with the TensorFlow framework as backend. The choice of Keras helped to simplify the development while TensorFlow allowed us to take advantage of GPU acceleration that sped up the entire training process significantly.
Another aspect that characterized the development was the use of dataframes that allowed a better management of all the informations about the data set.

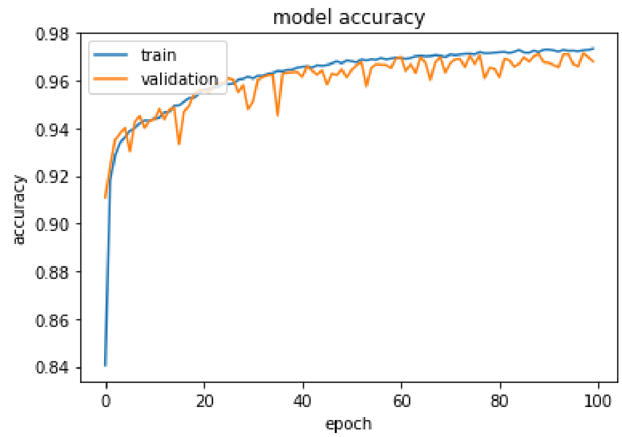
4.2 Patch extraction

The experimental activity started with the convolutional neural network trained with the first version of the training dataset. It consisted in a set of patches, extracted with the procedures explained before but without applying augmentation to sea lions ones.
The train set at the end of the extraction procedure includes 50079 'sea lions' and 1139531 'background' patches in the train set which will be split again in 40411, 9668 and 1129863, 9668 respectively for proper train and validation procedures. The test set instead has 13539 'sea lions' and 277390 'background' patches. Given this dataset divided into classes, train, validation and test set it's possible to train the model.
Going deep in the training and testing procedures we noticed some important problems with the dataset, first of all there were some mismatches in the ground truth provided by kaggle so we listed and removed all these images, secondly

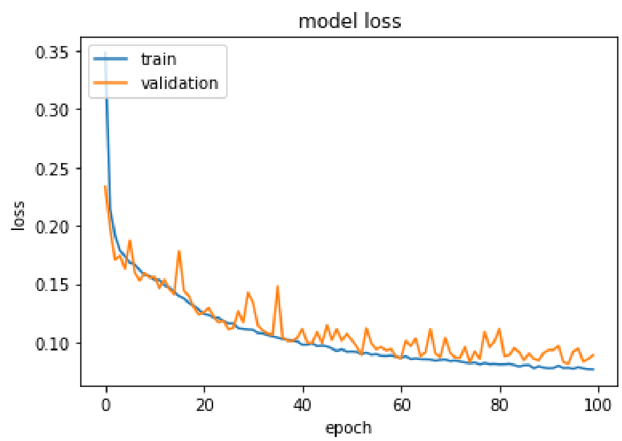
the blob detection technique provided some false positives that ended up labled in the wrong way, so we also inspected manually the patches to cope with this problem.

4.3 First training

As can be seen from these numbers the dataset is unbalanced, there are a lot more background patches than sea lion ones, but the first training was meant to be as simple as possible so balanced train and test set were created performing random sampling from background and augmentation technique was not applied. This simplified a lot the first training of the network and produced very good results.
For what concernes the training procedure, we ran it for 100 epochs using a learning rate of 0.0001.



(a) Model accuracy



(b) Model loss

Figure 4.1: Loss and accuracy history

To measure performances we considered not only the accuracy, but also the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. Up to this point, we achieved an AUC of 99.29%. Here follows more detailed tables about the results.

	Loss	*Accuracy*
Train set	0.0768	0.9733
Validation set	0.0891	0.9679

(a) End of training performances

Prediction accuracy	96.60%
AUC	0.9929

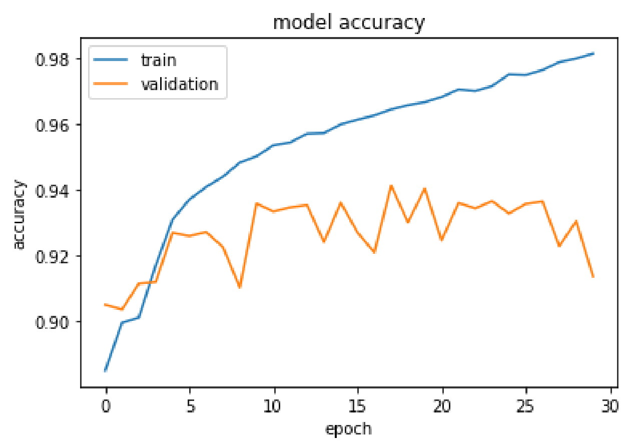
(b) Testing results

Table 4.1: First training results

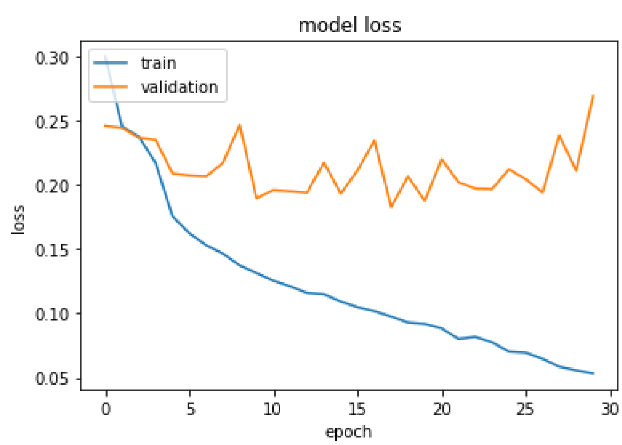
4.4 Second training

After that, we decided to use the whole dataset both for training and testing rather than a balanced one because the real problem is to identify sea lions among a lot of background. This caused different problems, in particular due to memory usage and to high number of background patches with respect to sea lions ones.
Memory related problems were linked to the fact that training dataset was of some Gigabytes, thus it couldn't fit all in the memory at the same time. To overcome this we created a batch generator which retrieved a limited number of patches from the memory and fed them to the network at each epoch. In this way only a little amount of patches were moved into the RAM at the same time. After they were used by the network, they were flushed.
This generator also enabled us to introduce data augmentation in the training step. As stated in the previous sections, it was needed due to the high difference in number between sea lions and background patches and also to obtain more robust results in classification, in fact with this technique it's more likely to be rotation and scale invariant.
In this first attempt to add augmentation we used an handcrafted tool to apply the transformations, this choice allowed us to work with higher size sea lions patches that were cropped after the transformation to fit the network input. In this way we could avoid interpolation on eventual missing pixels that was necessary with Keras built-in functions since we applied the transformations at runtime.
To train the network we decided to create a balanced data set composed of all the sea lions patches with augmentation and an equal number of background patches random sampled from all the background patches. This dataset is used for one epoch and then changed, creating a new one with the same criteria. In this way the network sees all the sea lions patches different times but overfitting is prevented by the wide amount of modifications introduced by the augmentation. While the background patches are used in an efficient way because network sees a wide variety of them while keeping the procedure memory efficient.
Although these premises, results were not good and the network very bad, being

not so better with respect to a random classifier.
This training allowed us to identify three new problems: changing the dataset at each epoch is time consuming, because loading the patches from disk to memory is a bottleneck in the whole procedure, moreover because of the handcrafted augmentation tool even the batch creation was slowed down; puppies are really hard to be identified by the network, with their color and shape can be easily mistaken as background also by human eye.
Given that this training was particularly time consuming for the reasons discussed before, the network was trained for only 30 epochs with a 0.0005 learning rate. The procedure took a lot of time and was not sufficient to learn enough from the data provided.

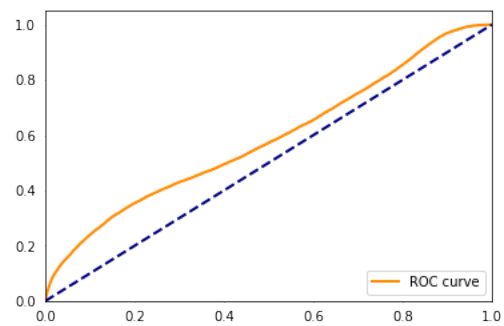


(a) Model accuracy

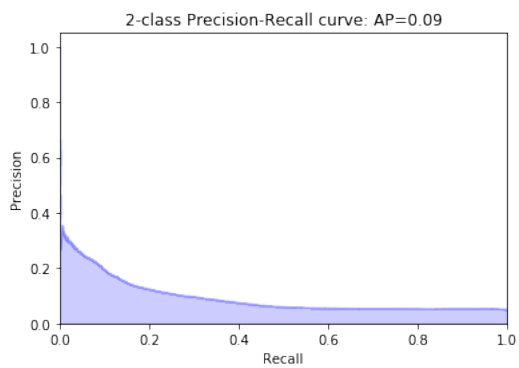


(b) Model loss

Figure 4.2: Loss and accuracy history



(a) *ROC*



(b) Precision-recall curve

Figure 4.3: *ROC* and precision-recall curves

As shown by the following tables the presence of a much higher number of background patches influenced a lot the metrics used to estimate the model performances.

	Loss	*Accuracy*
Train set	0.0533	0.9813
Validation set	0.2689	0.9138

(a) End of training performances

Prediction accuracy	94.91%
AUC	0.5864

(b) Testing results

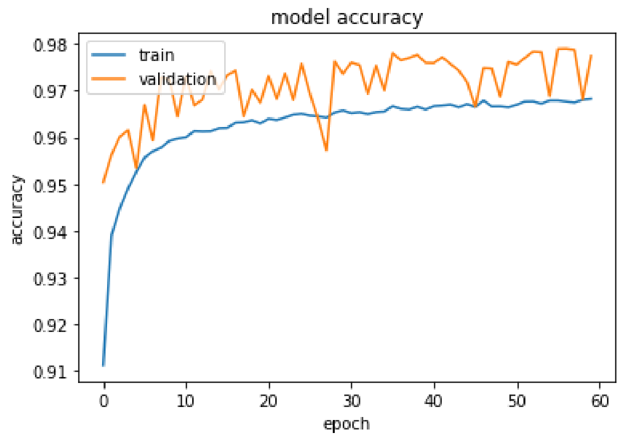
	Precision	*Recall*
Sea lions	0.2948	0.0228
background	0.9516	0.9972

(c) Precision and recall scores

Table 4.2: Second training results

4.5 Third training

The purpose of this training was to solve all the problems that came up in the previous one.
The first adjustment was to change the dataset and reload it from memory every 15/20 epochs and not at each epoch, this choice speeded up the whole training phase allowing us to train for more epochs. The problem related to puppies was solved analyzing where and how they appear in the images, in fact it can be noticed that this particular class is often next to a female sea lion.
Thanks to this information we were able to apply a second important change: remove puppies from the dataset. In this way for the network it's easier to learn and distinguish between all the other classes of sea lions and the background so it's possible to achieve better performances. At the end, in the final result, the number of puppies can be estimated from the number of sea lions detected.
Another remarkable difference with respect to the last training was that we dropped the handcrafted augmentation tool in favor of keras built in one that enabled a much higher throughput in the creation of the training batches.
Performances of this model were measured again using accuracy, that is 97.27%, and *AUC* of the *ROC* curve, which is 99.64%, plus other metrics such as precision and recall respectively 62.53% and 97.68% over the sea lions.

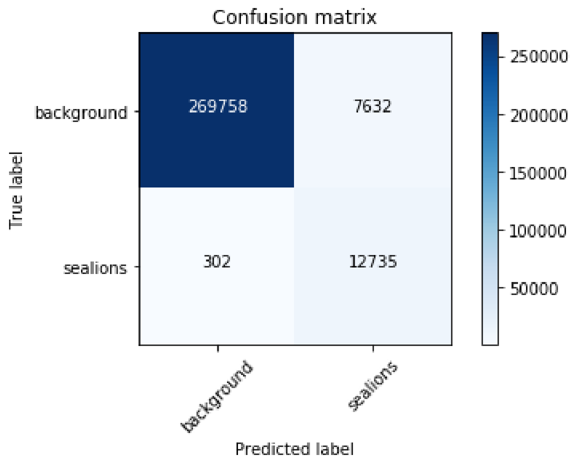


(a) Model accuracy

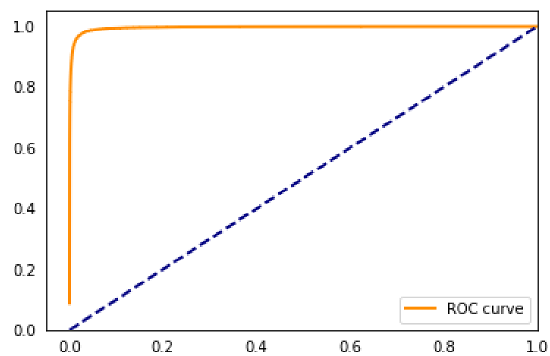
(b) Model loss

Figure 4.4: Loss and accuracy history

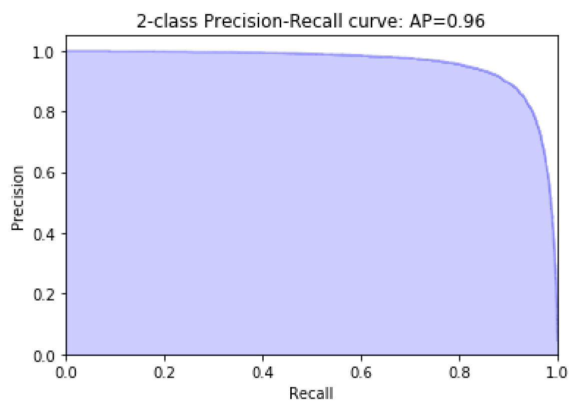
During this phase we set the learning rate to 0.0005 and we trained for 60 epochs.



(a) Confusion matrix



(b) *ROC*



(c) Precision-recall curve

Figure 4.5: Confusion matrix, *ROC* and precision-recall curves

Prediction accuracy	97.27%
AUC	0.9964

(a) Testing results

	Precision	*Recall*
Sea lions	0.6253	0.9768
background	0.9989	0.9275

(b) Precision and recall scores

Table 4.3: Third training results

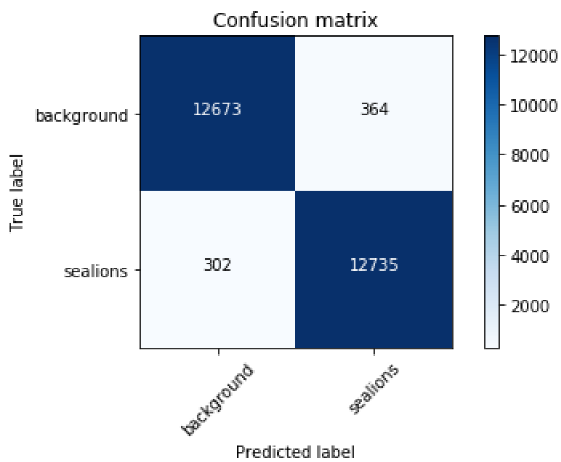


Figure 4.6: Confusion matrix with the balanced test set

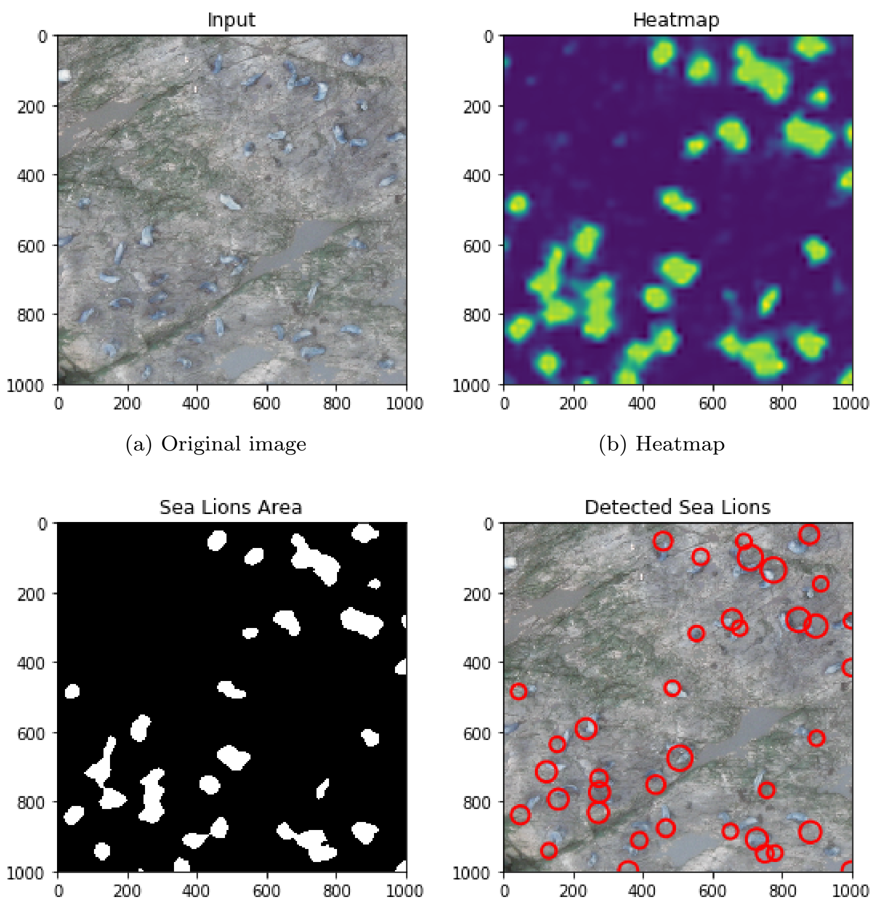
	Precision	*Recall*
Sea lions	0.9722	0.9768
background	0.9767	0.9721

Table 4.4: Precision and recall scores on a balanced test set

4.6 Semantic segmentation and sea lions counting

At this point we took the last trained network and, as explained before, replaced the last fully connected layer with a convolutional one maintaining all the weights. With this reshaped network we are able to evaluate an entire image and obtain a heatmap highlighting zone where is more probable to have a sea

lion.
Then we decided to apply a threshold on the heatmap to create a grayscale image with white zones corresponding to sea lions. On this new image it was possible to apply blob detection to have a rough count of how many sea lions are present in the considered image, the obtained results are the following. . .



(a) Original image (b) Heatmap

(c) Blobs (d) Detected sea lions

Figure 4.7: Sea lions detection

Chapter 5

Conclusions

The work done for this project produced a convolutional neural network able to achieve good performance on the classification of sea lions over background. In particular we want to point out how the two major enhancements we did to improve the performance of the network, i.e. data augmentation and the management of the dataset during the training, have actually set out the condition for a good classification.
The first approach to augmentation didn't produce the expected results. This was mainly caused by the high variety of data introduced by it and the complexity for the network to learn it. In fact, the high computational cost of the training didn't allow us to run an high number of epochs. Thus, the benefits of augmentation were overcome by its disadvantages. After we introduced generator and by it optimized the memory usage, we found out that the augmentation provided the expected improvements.
There are still some open problems linked to this project.
First of all we have overcome the recognition of puppies due to the intrinsic difficulty of the task. Further works could reintroduce them with a correct classification.
Our task was to produce a binary classifier which could distinguish between a sea lion and the background. An extension to this could a multiclass classifier which can effectively recognise among the classes of the original competition.
Another approach that could produce an improvement in the results is to exploit transfer learning from models trained with imageNet, for example, since they provide optimal performances in low level feature extraction.
The last thing we imagine could work better is a direct object recognition approach without going through the classification task, however this would require a much more detailed ground truth.