Project report

Course teacher

Professor Vincenzo Caglioti

Project supervisors

Professor Giacomo Boracchi

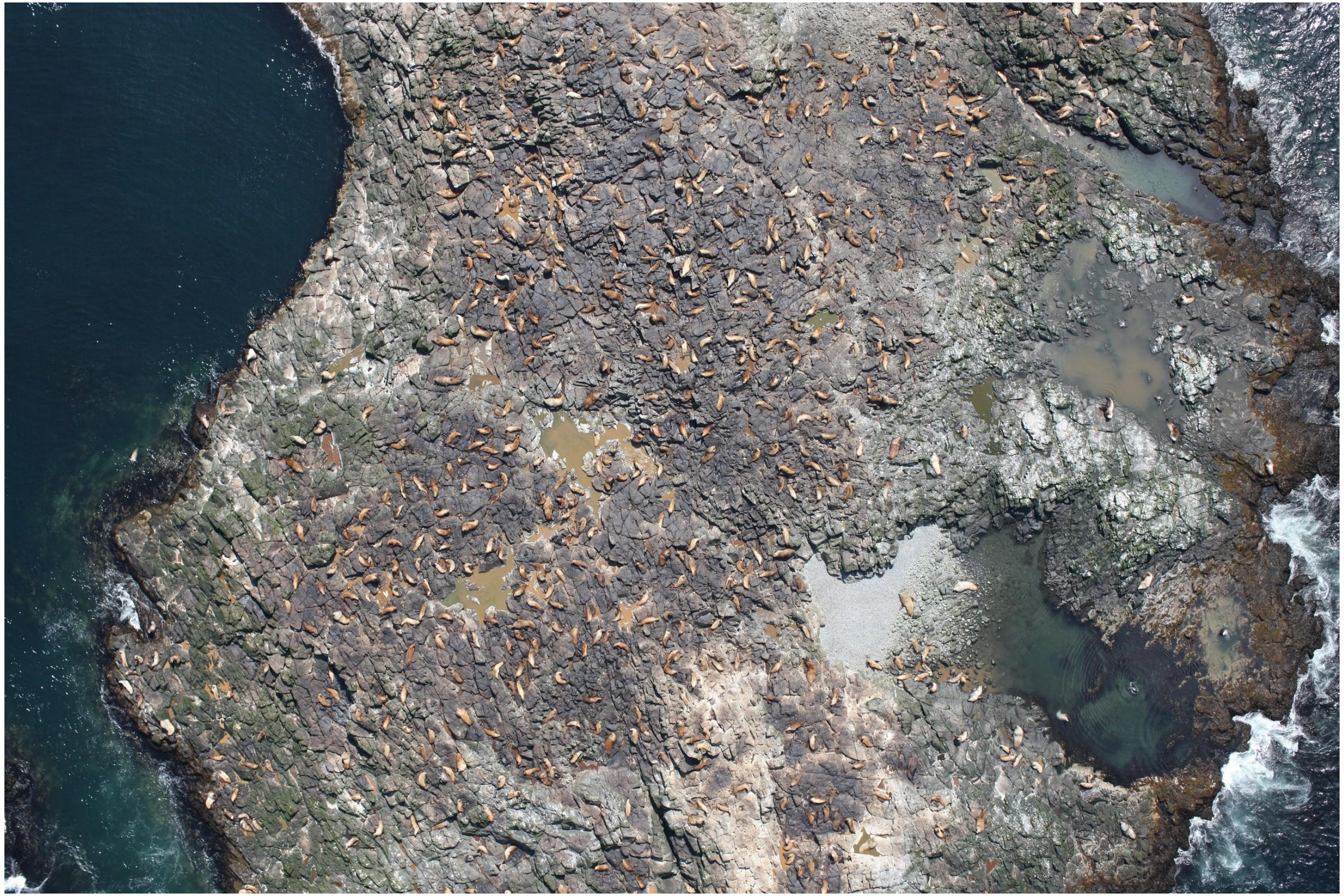
Professor Diego Carrera

Group

Alex Perugini	876359
Vincenzo Scotti	875505
Marco Re	873564

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
- Original Kaggle competition aim was to count the population of sea lions, dividing them into 5 classes: adult male, adult female, subadult male, juvenile and puppy
- The requirement assigned to our group for the project was to provide a classifier able to distinguish between sea lion and background
- Dataset provided by Kaggle, composed of images and relative ground truth



Sample image from the dataset

State of the art

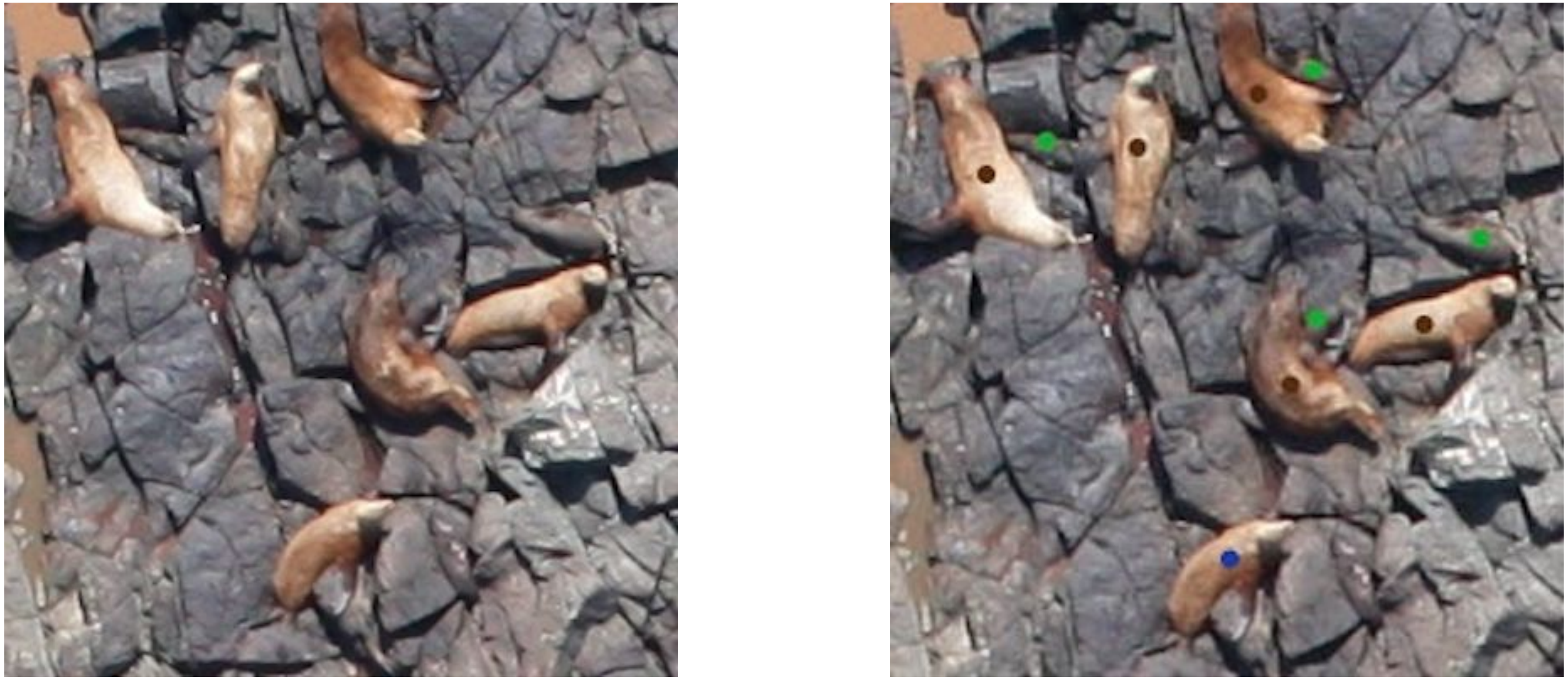
- Image classification is a fundamental problem and at the basis of other computer vision tasks
- Traditionally handcrafted features were first extracted from images using feature descriptors and then used to train the classifier
- In recent years, deep learning exploiting multiple layers of nonlinear information processing have become the main trend
- CNNs are the leading architecture for most image recognition, classification and detection tasks
- Fully CNN exceeded state of the art for semantic segmentation and object detection tasks

Solution approach

- Very large images which can’t be used as a whole to train the network
    - Extracted patches from the dataset and used them to train the classifier
- Each image contains more background patches than sea lions ones, this implies that the dataset is very unbalanced
    - Applied augmentation to sea lions patches to partially overcome this problem, increasing a lot the dimension of the original dataset
- Original problem was to count sea lions in the provided image
    - Extended our work to provide the heatmap of the image and to estimate the number of sea lions in it

Implementation: Dataset creation

- Dataset composed of 947 images
  - First 750 used for training
  - Remaining used for testing
- Extraction performed through the absolute difference between the original image and the ground truth
  - Gathering of coordinates and classes of all sea lions in the image
- Background patches extracted by a sliding window over the image cutting all the patches not intersecting a sea lion one
- 96x96 patches



Detail of a sample image and its related ground truth

Implementation: Dataset creation

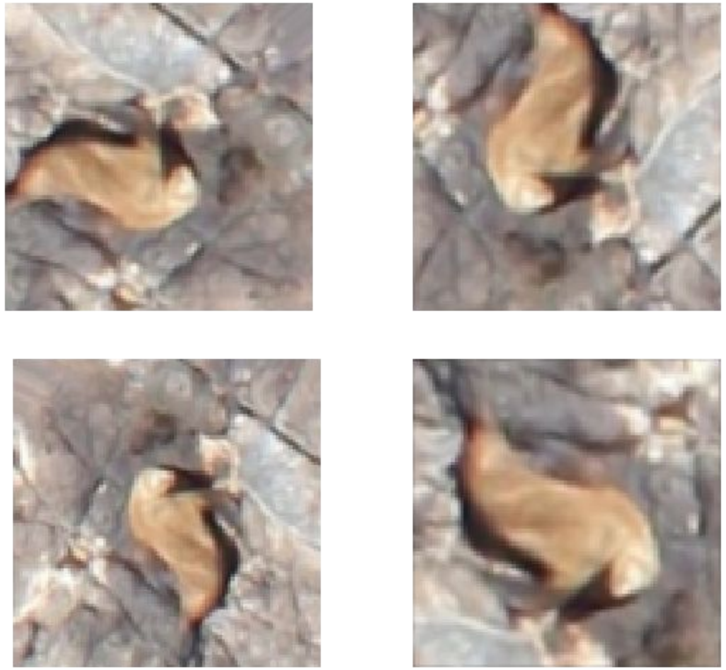
- Image difference and subsequent blob detection for sea lions coordinates
  - train: 40411
  - validation: 9668
  - test: 13539
- Sliding window and deletion of patches containing sea lions from previous section for background coordinates
  - train: 1129863
  - validation: 9668
  - test: 277390

Implementation: Data augmentation

- ➔ Dataset highly unbalanced
    - ◆ too much background patches
    - ◆ not enough sea lions
- ➔ Classifier becomes rotation, scale and position invariant so it allows to achieve more robust classification

- 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

Implementation: Data augmentation samples

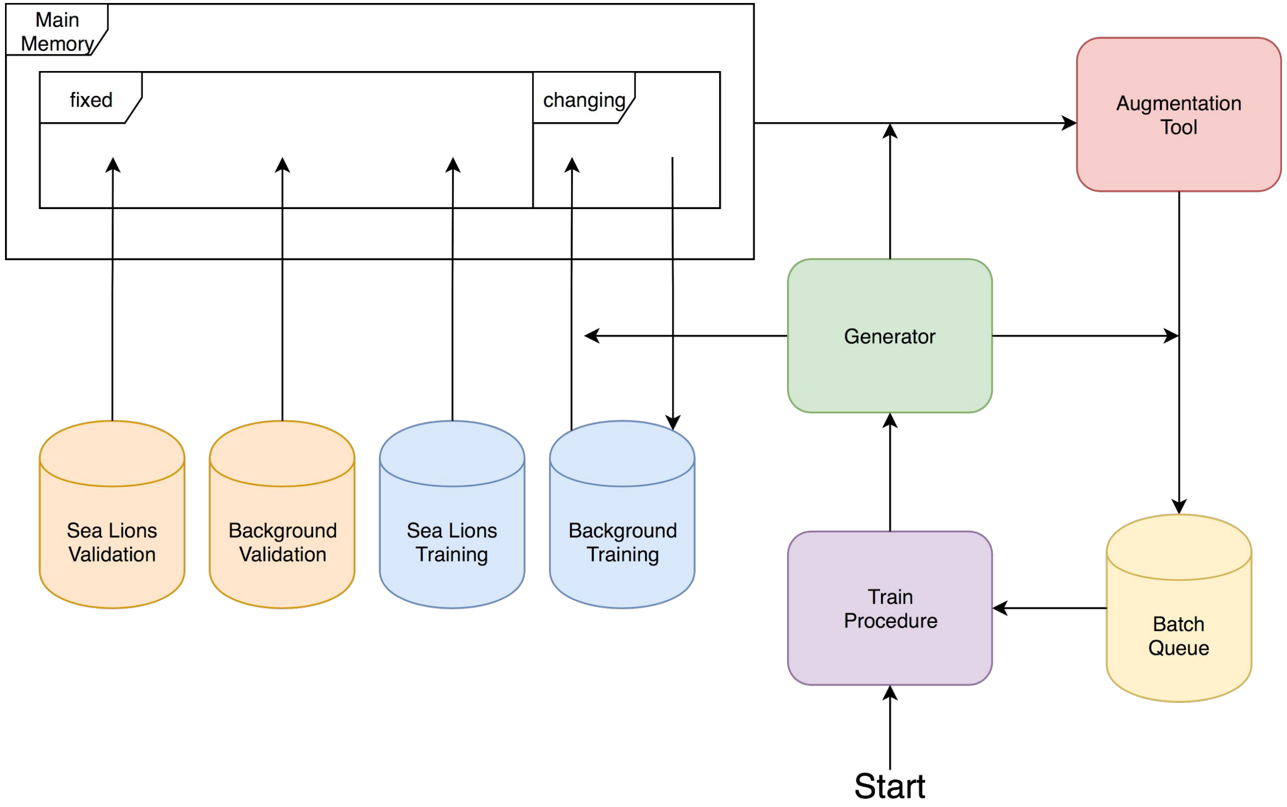


- New sea lions patches generated from original one
- Dataset more balanced

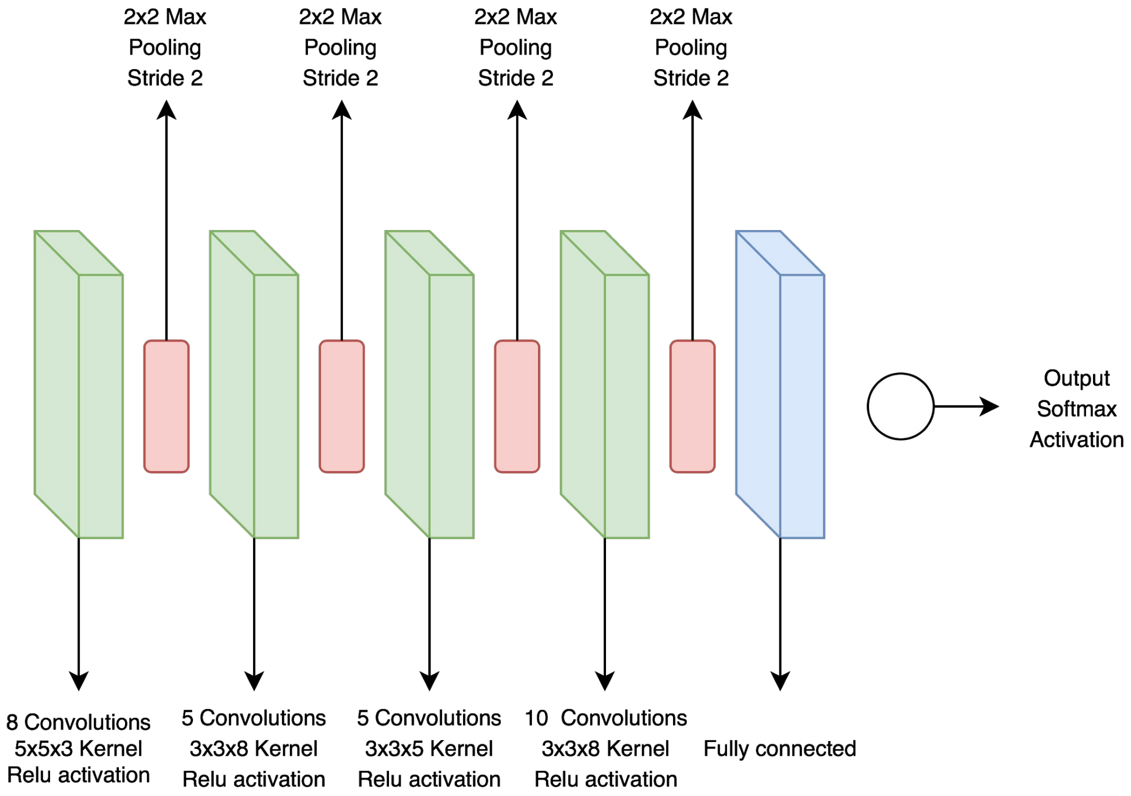
Implementation: Batch generator

- Too little main memory and device memory on the GPU and changing train set led to necessity of efficient memory management
- With batch generator data are moved to the training model only when needed
- Batch generator leverages Keras queue system to avoid waiting for transformations
  - Each batch is generated as the previous are being processed
- Fixed size batch avoids busy memory explosion when moving from 8 bit pixel intensity to normalized float 32 representation
- Portions of the data set for both train and validation that fits in memory are still loaded entirely
  - necessary for sea lions to speed up augmentation
  - background patches are periodically flushed and re-sampled instead

Implementation: Batch generator



Implementation: Model


2x2 Max Pooling Stride 2
2x2 Max Pooling Stride 2
2x2 Max Pooling Stride 2
2x2 Max Pooling Stride 2
Output Softmax Activation
8 Convolutions 5x5x3 Kernel Relu activation
5 Convolutions 3x3x8 Kernel Relu activation
5 Convolutions 3x3x5 Kernel Relu activation
10 Convolutions 3x3x8 Kernel Relu activation
Fully connected

Implementation: Model

- 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

Implementation: Object detection

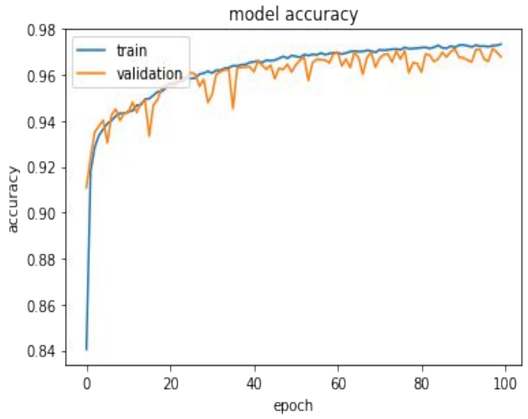
- Modified the CNN model to get the fully convolutional one
- Final layer of the network modified into a convolutional layer using 2 filters and a 4x4 kernel, softmax activation function
- Same weights and connections of the previous network
- Dropped constraints on the input shape to use the whole image as input
- Output composed of two heatmaps, highlighting respectively pixels belonging to the sea lion and the background classes according to the probabilities predicted by the classifier
- Cubic interpolation to reconstruct image size heatmap

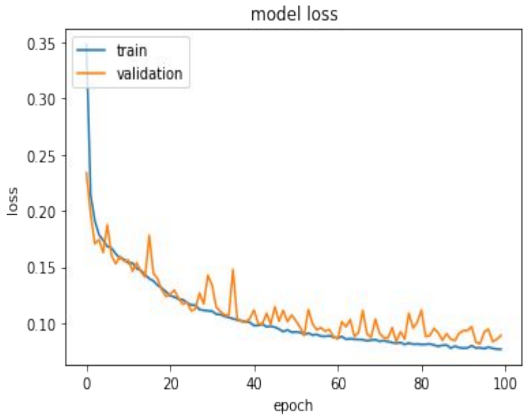
Experimental activity and results: Development tools

- Jupyter notebooks
  - Allowed to write Python code and run it section by section
- Keras
  - High level neural network API with TensorFlow as backend
- Dataframes
  - Simplified a lot the management of all the informations about the data set

Experimental activity and results: Balanced dataset

- As simple as possible
  - No augmentation
  - Balanced train set and test set
- Training
  - 100 epochs
  - Learning rate 0.001





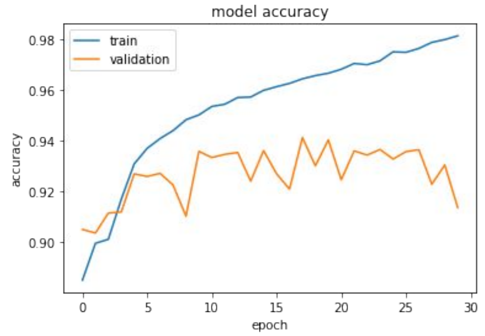
Experimental activity and results: Balanced dataset

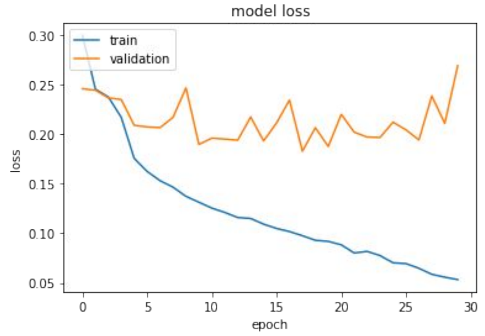
	Loss	Accuracy
Train set	0.0768	0.9733
Validation set	0.0891	0.9679

Prediction accuracy	96.60%
AUC	0.9929

Experimental activity and results: Batch generator and augmentation

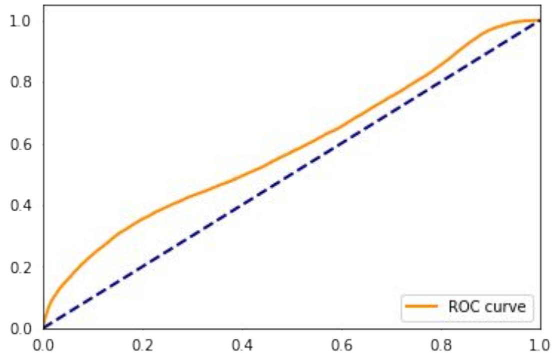
- Attempt to use entire data set
  - Use of a custom generator to have balanced data set at each epoch of the training by subsampling randomly the background patches at runtime
  - Efficient memory management through custom generator
- Augmentation
  - Handcrafted augmentation tool
- Training
  - 30 epochs
  - Learning rate 0.0005





Experimental activity and results: Batch generator and augmentation

- Poor results
  - Changing dataset at each epoch is time consuming
  - slow augmentation tool
  - hard to identify/distinguish puppies
- Near random classifier performances



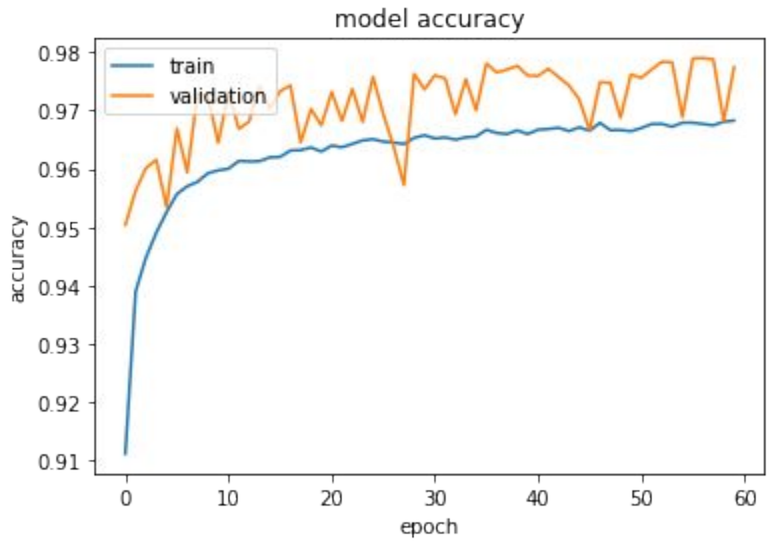
Prediction accuracy	94.91%
AUC	0.5864

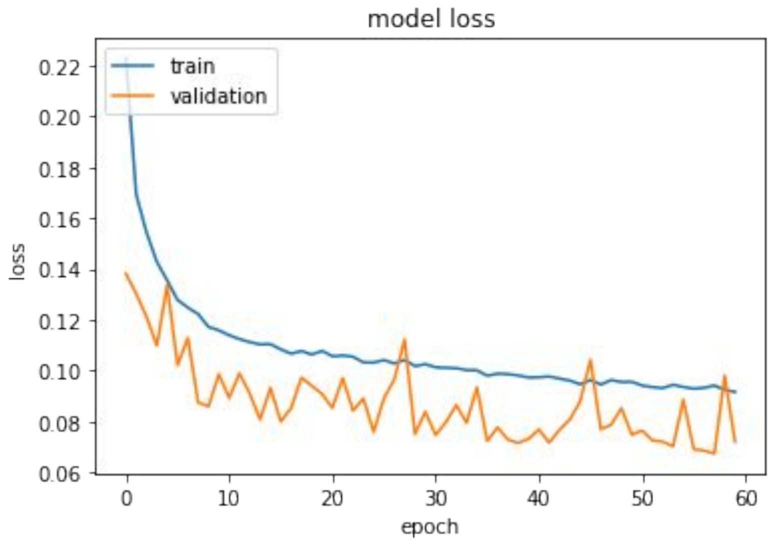
	Precision	Recall
Sea lions	0.2948	0.0288
Background	0.9516	0.9972

Experimental activity and results: Keras augmentation and puppies exclusion

- Dataset reloaded from memory every 15 epochs
  - Speeded up the training
- Puppies removed from the dataset
  - For the network it’s easier to learn
  - Number of puppies can be estimated from the number of female sea lions detected
- Augmentation
  - Kears built-in one
  - Higher throughput
- Training
  - 60 epochs
  - Learning rate 0.0005

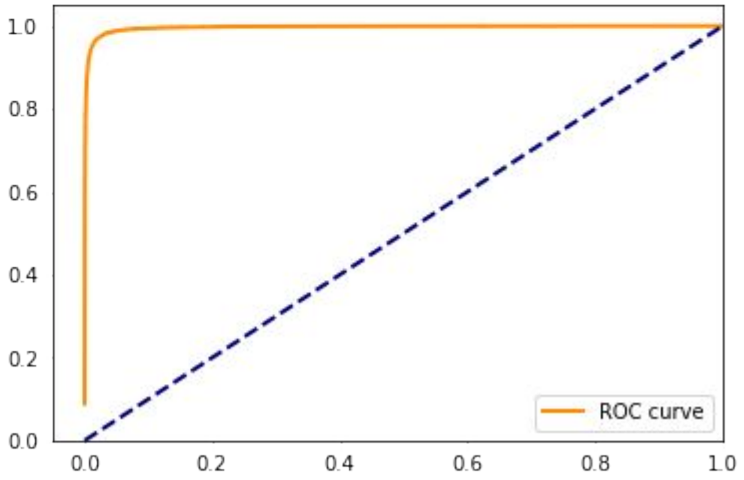
Experimental activity and results: Keras augmentation and puppies exclusion





Experimental activity and results: Keras augmentation and puppies exclusion

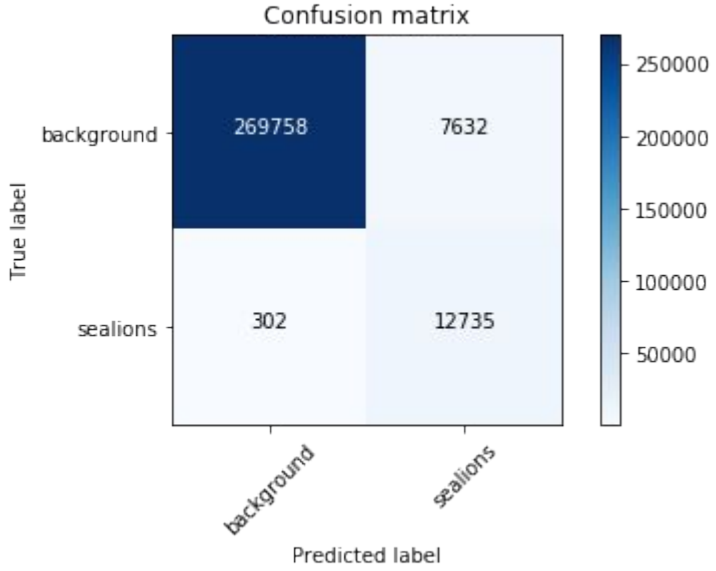
- Very good performances
- ROC almost covers all the area
- AUC higher than the first one



	Precision	Recall
Sea lions	0.6253	0.9768
Background	0.9989	0.9275

Prediction accuracy	97.27%
AUC	0.9964

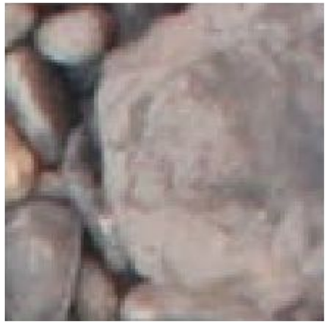
Experimental activity and results: Keras augmentation and puppies exclusion

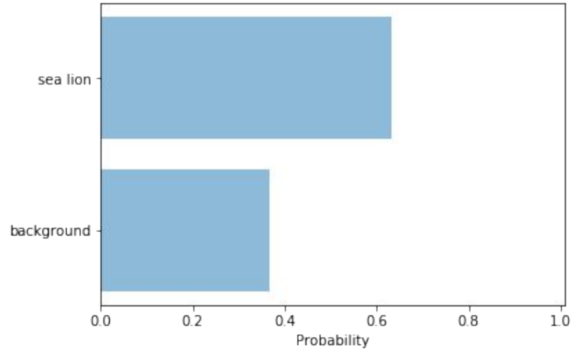


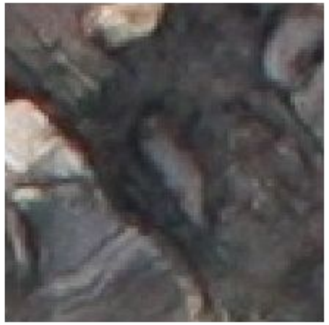
	Precision	Recall
Sea lions	0.6253	0.9768
Background	0.9989	0.9275

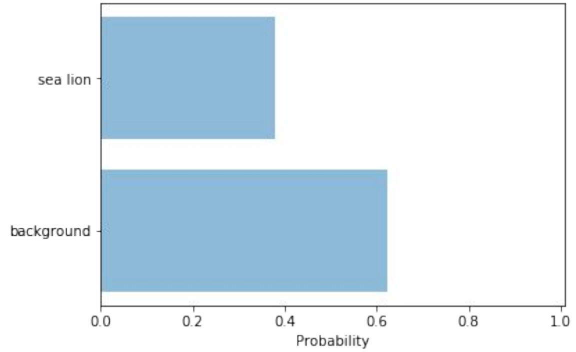
Experimental activity and results: Misclassified patches

- Reasonable mistakes
- Some rocks can appear as sea lions due to their shape and size
- Some sea lions are really hard to see and detect







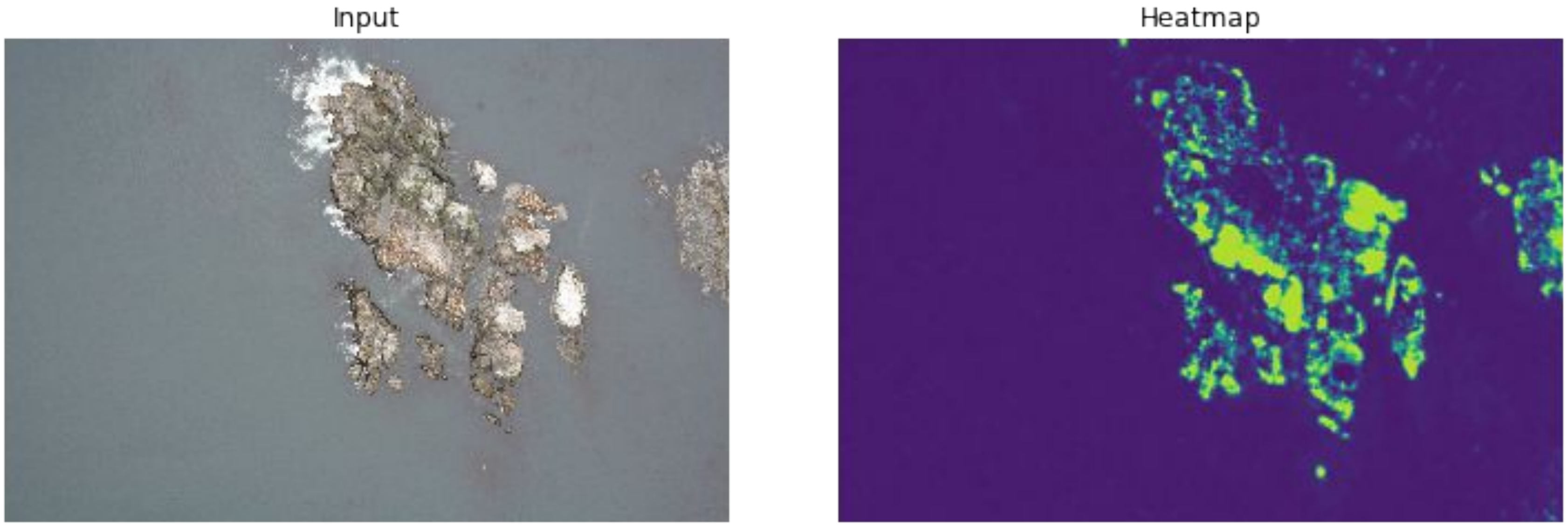


Experimental activity and results: Object detection and sea lions counting

- Network reshaping into Fully CNN
    - Final layer from flat to convolutional
    - Entire image evaluation
- Heatmaps
- Detection
    - Heatmap thresholding
    - Blob detection
    - Counting

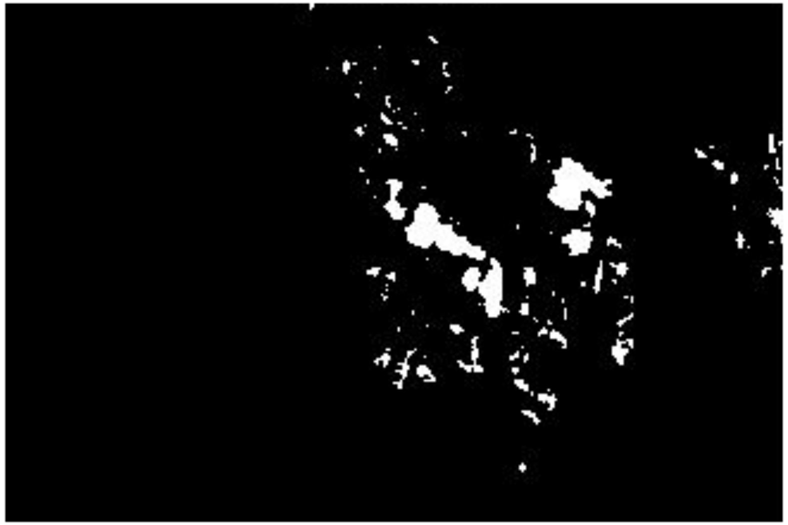
	True	Predicted
Image 755	212	233
Image 771	149	141
Image 773	391	411
Image 907	93	123

Experimental activity and results: Object detection and sea lions counting

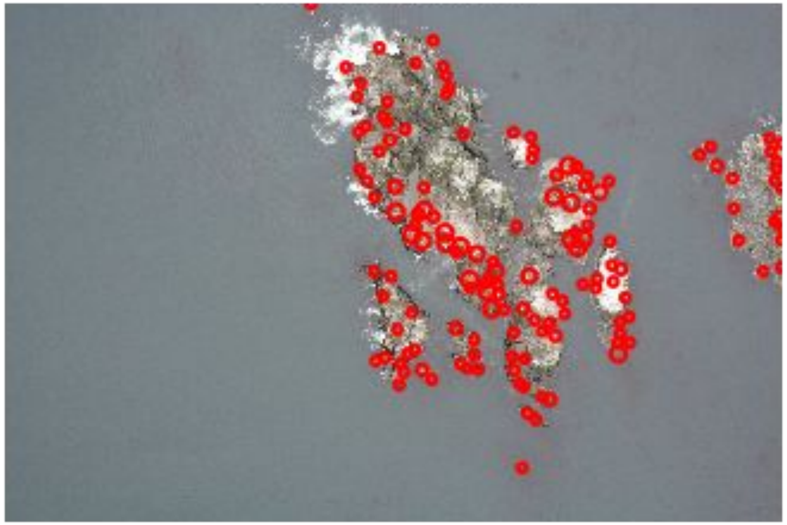


Experimental activity and results: Object detection and sea lions counting

Sea Lions Area



Detected Sea Lions



Experimental activity and results: Object detection and sea lions counting

- Non-optimal solution
  - Many sea lions are not identified
  - Many background areas (often rocks) are counted as sea lion
  - Blobs are not correctly positioned even in the correctly identified instances
  - Predicted number is an overestimate
  - True and predicted numbers are similar but puppies are not considered in the prediction

Conclusions

- Final network has good performance on the classification of sea lions over background
- Two major enhancements that allowed this high performances
  - Augmentation
  - Management of the data set during the training
- Augmentation
  - At first made results worse
  - Too high variety introduced in the data set with respect to the training we could do
  - After memory handling optimization, provided expected improvements
- Management of the data set during the training
  - Allowed to speed up the training a lot by optimizing the memory usage
  - Allowed the network to deal with the augmented data set, increasing robustness of the classification

Open problems

- Reintroduction of the puppies in the data set
- Extension from binary classifier to multiclass classifier
- Transfer learning to improve performances
- Object recognition approach without going through the classification task