Detection and Classification Algorithms

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Table of Contents

Final Algorithm

The final algorithm will be a combination of the Classification algorithm and the Detection algorithm. The photo is analyzed first by the Detection algorithm which creates a box for each bird. Once it's result is validated, an image is created from each box, and then the Classification algorithm is applied on each one. By defining things this way, we managed to split the work in two independent parts, each one with its problems to solve.

Classification

Researches

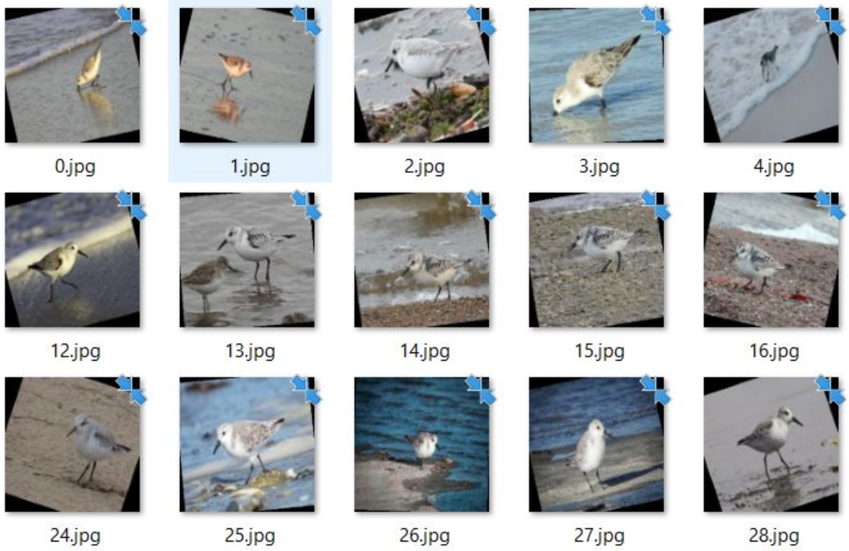
Classification is the simplest part of the Final Algorithm. Convolutional Neural Networks **[1]** are very efficient at classifying images (we have seen in introduction the results of the very deep neural networks). Therefore, we chose to use one of the pre-trained models **[2]** and used it as a base for our neural network.

We used the pre-trained neural network by adding a few layers to it, and freezing the pre-trained part as we trained the neural network to do what we want. In a second part of the training process, it is possible to unfreeze the pre-trained part and train the whole algorithm once again, in order to get better results **[3]**.

There was a problem with the database, which the client couldn't provide us, so we decided to use images found online using a web-scraping algorithm, detailed in the Detection part.

However this method wasn't perfect and we had to make a selection in this first database (we didn't want images with more than one bird, or drawings etc). We came up with a database of 250 images per species, which we then improved using Data Augmentation (artificially creating new images by turning, changing the brightness, etc).

Example of Data Augmentation



Our code

Input images are variable size images collected on the internet, they are converted into RGB images, resized in the shape 224*224*3 and then normalized with a mean and standard deviation vector expected by the pretrained convolutional neural network used.

For the structure of the neural network, we used the pre-trained convolutional neural network Resnet18 and we added a fully connected layer at the output of Resnet18, so that we get an output with as many digits as there are species to be predicted. At the end, we pass the output in the softmax function to get the probabilities, computed by our neural network, that the bird on the input pictures belongs to a certain species.

Since we are dealing with a classification problem, we used the cross-entropy as our loss function, we also added a weight decay (adding the sum of squares of the weights of the neural network) to our loss function in order to avoid overfitting.

We used pytorch DataLoaders to divide our training and validation data into batches of size 32. We did batch training using the optimizer Adam and the learning rate scheduler “OneCycleLR”.
At every epoch, we printed the train and validation accuracy, in order to have an idea of how well our neural network was learning.

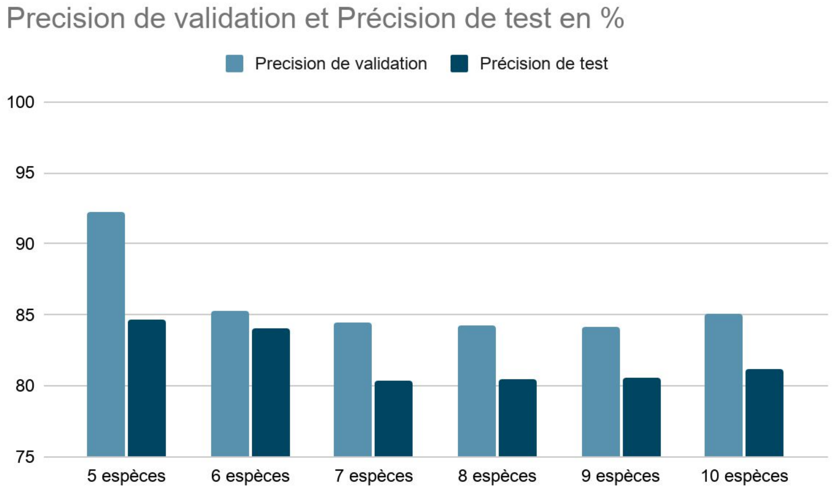
For the evaluation, we used the third part of our dataset, the test dataset (about 40 images per species), and measured the accuracy as well as plotted the confusion matrix. Using a test dataset was very useful, because we would be able to say, if the precision metrics measured on those datasets was representing well, how our algorithm was generalizing his knowledge. Using the validation dataset to do so would’ve biased our evaluation, since we used it to adjust hyperparameters of our algorithm.

Results

Accuracy for each Version of the Dataset

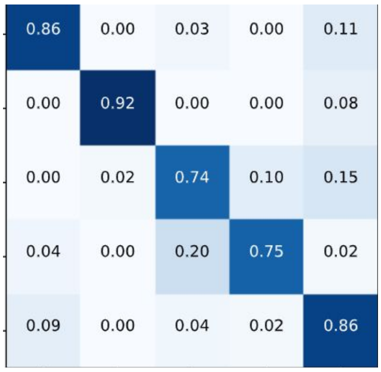
	5 species	5 species Augmented	10 species Augmented	5 species Augmented +unfreezed training
Accuracy	82,3%	84,7%	81%	To do
Training Time(min)	30	33	35	

Accuracy between 5 and 10 species in our Dataset



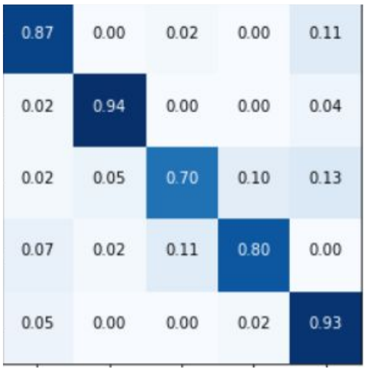
As we can see, the performance of the algorithm doesn't depend on the number of species in our dataset. In fact, the accuracy on the validation set is only changed between 6 and 7 species, because we added a species that was similar to another one. To get a closer look, here is the confusion matrix for 5 species (lines correspond to the ground truth and columns to the predicted birds. The numbers are the proportion of prediction).

Without Data Augmentation



0.86	0.00	0.03	0.00	0.11
0.00	0.92	0.00	0.00	0.08
0.00	0.02	0.74	0.10	0.15
0.04	0.00	0.20	0.75	0.02
0.09	0.00	0.04	0.02	0.86

With Data Augmentation



0.87	0.00	0.02	0.00	0.11
0.02	0.94	0.00	0.00	0.04
0.02	0.05	0.70	0.10	0.13
0.07	0.02	0.11	0.80	0.00
0.05	0.00	0.00	0.02	0.93

There were in fact already two species for which the algorithm had trouble to distinguish, but the augmentation of Data seems to be a solution to the problem.

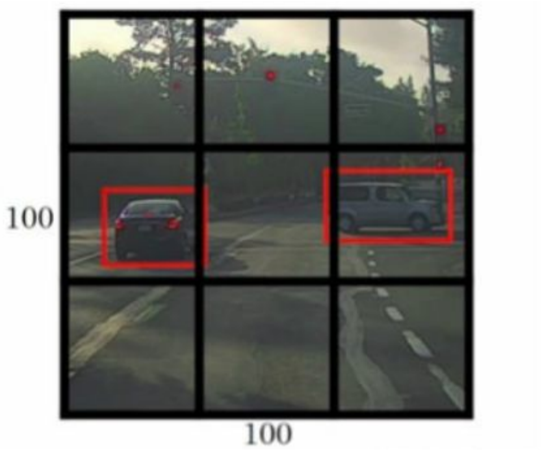
Using YOLO

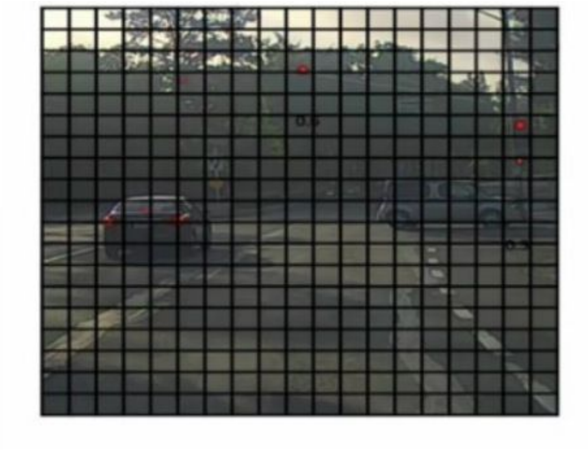
We also tried another method by using yolo, which is mostly known for its use in Object Detection, and use it for Classification.

1. Introduction to the Yolo Algorithm

Because only one CNN network is being used, the first advantage of Yolo is its process speed, which fits to the tasks that imply large amounts of Data.

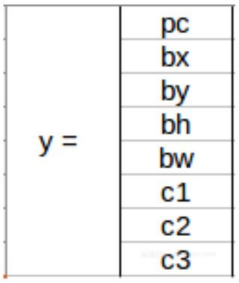
The idea behind Yolo is to divide the image in little squares, and to apply the labels on each square. To simplify, we considered a 3 by 3 grid (on the left), but in reality there are more squares (for example 19*19 on the right).





YOLO algorithm applies a classification to the image and a position treatment to each square in order to get the global box that contains the object and the corresponding class probability.

First we have to transmit the labeled data to the model for the training. Let’s suppose that we work with **3 by 3** grids and that there are a total of 3catégories : species 1, 2 and 3. Consequently, for each square, the label **y** is a 8 dimensions vector.

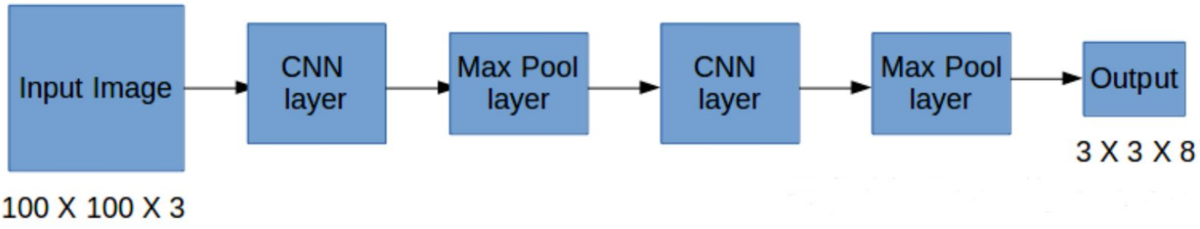


y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

pc tells us if the object is in the square or not. ***bx, by, bh, bw*** specify the box that contains the target in the square. ***c1, c2, c3*** are the categories (species) that we are trying to identify. For example, if the bird detected belongs to the 2nd category, then ***c2*** = 1, ***c1*** = ***c3*** = 0.

There is one 8 dimensions vector per square, so the dimension of the output is **3x3x8**.

If we assume that the input image has a size of 100x100x3, the model will be formed as follows

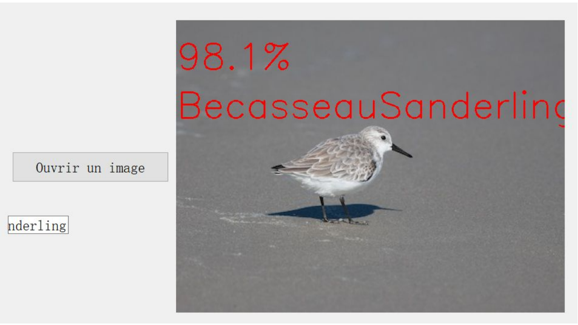


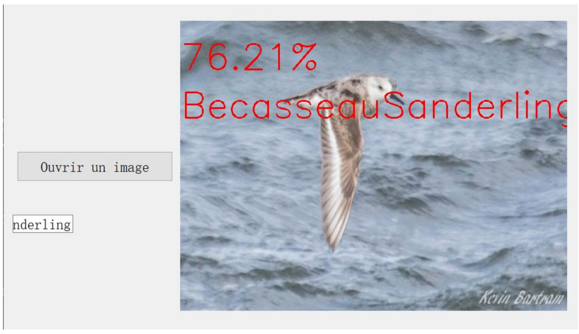
2. Process

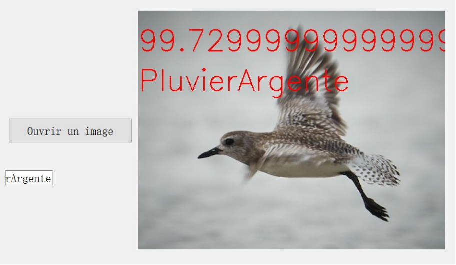
We used images from 10 different species for the training. These Data are the same that we used for the other algorithm of classification, so go to the last sections to get more information about how we acquired them etc.

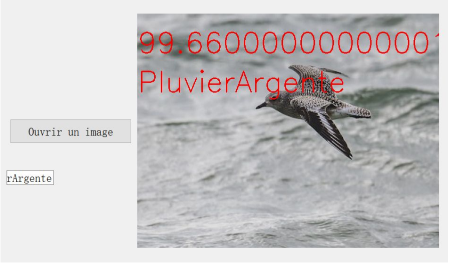
3. Results and analysis

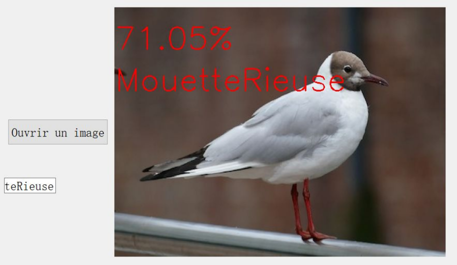
Some results are selected at random as follows:

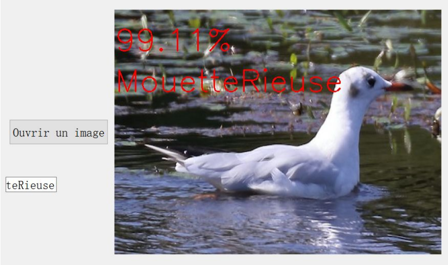


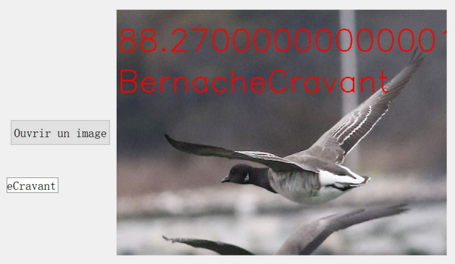


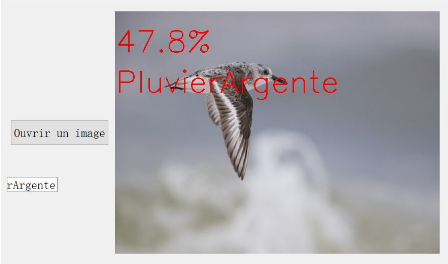






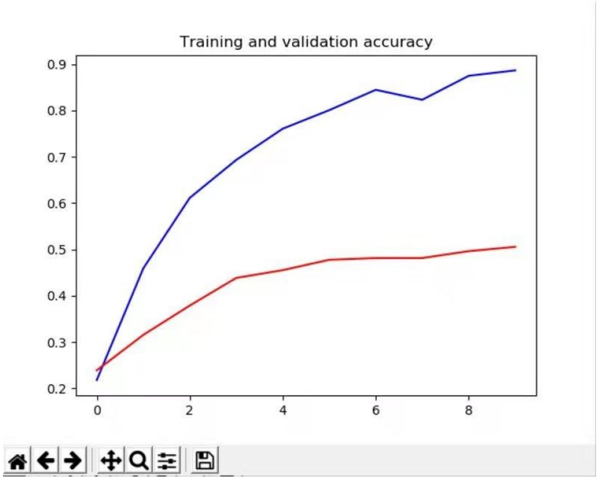


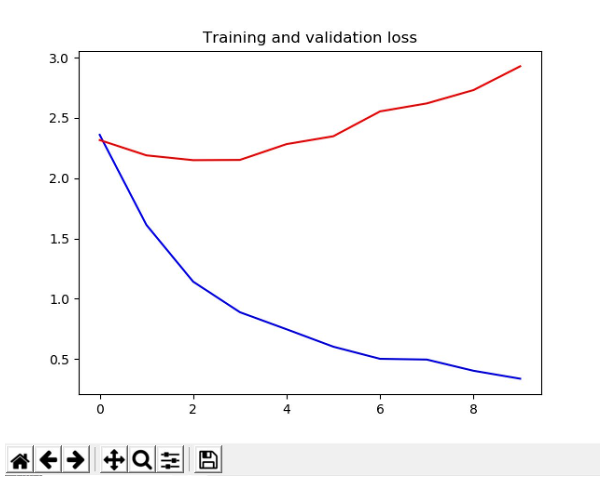

88.27000000000001
BernacheCravant


47.8%
PluvierArgente

The images above show that the accuracy of the classification is relatively high. For most of the images, the accuracy of classification reaches 85% and almost all the images have an accuracy above 70%. However, it's far from perfect, and as we can see in the last image, the algorithm can be wrong and identify Bécasseau Sanderling as a Pluvier Argenté.

Here are the evolution of ***the training and validation accuracy*** and ***the training and validation loss*** during our training:





The blue line represents ***training*** and the red line represents ***validation***.

Detection

Researches

Detection was more difficult than the Classification part, because we need to show where the bird is in a given picture, whereas the classification algorithm only has five information about what the photo is representing. The number of possible positions for a bird is very high, and that is why finding it is way more difficult than classification. Finding a database is also harder in many cases. However, a classical dataset, COCO **[4]** features around birds. This dataset was used to train neural networks to detect objects, animals or people but this database isn't close enough to the data we would work with in the end (not enough birds on the pictures, not all the birds were labelled, not the same species etc). However, this database can be really useful because it features an enormous number of photos (around 100k), and around 3000 photos of birds.

Because of the time it would require to create our own Dataset for Detection (labels are more complicated than the ones used for the Classification), we chose to focus on the neural network, even if we created a tool to produce a Dataset in the future. This tool is made to collect images online only by pressing a button, instead of Ctrl+Save+Choose the Location, which makes it way faster:

Web-Scraping

Web scraping consists in automatically downloading images from the internet. In order to do web scraping, we used the html code of the page. Thanks to a library called selenium, we managed to interact with the page elements using python. We posted a tutorial in our github.

Back to the researches about Detection

The challenge of this task is to know where are birds in the photo without taking every possibility into account because this would be too complicated. We have found two ways in the litterature to do this kind of operation:

The first option is to extract useful information from the data using libraries such as OpenCV, which offers tools based on image processing in order to limit the number of regions that we will have to look at. This tool doesn't need any data to train. Once this step is done, the data is fed to the neuronal network. The neuronal network has to be designed in order to process the different sizes given to them.

This kind of method is easier to train, because we don’t need data in order to find possible regions (we need them after). Depending on the algorithm, the training time can be long, and the inference time (application of the algorithm once it is trained) is always longer than the other option. Some example for feature extraction **[5]** :

- Using the edges (for example with the contours provided by opencv **[6]**)
- Clustering methods (for example K-Means clustering algorithm)
- Selective Search **[7]** (used in rcnn and fast-rcnn **[8]** for example)
- Artificial Neural Network based segmentation...

The second option is to use tricks and the power of the neuronal network, to reduce the number of regions that we have to consider. Here, we don’t try to extract useful information from the data, but to modify these data to make it easier to work on.

For example, in one of the algorithms, in order to consider less regions and to be able to feed it to a neuronal network, we use anchors. Anchor is a set of different box sizes that we apply to different parts of the image. The idea behind this option is to link the anchors to a neuronal network, in order to predict if this region can be associated with an object. The neuronal network also adjusts the box, in order to fit the object better.

After that, the regions that look the same are removed, using the concept of IoU (intersection over union).

We have chosen to focus on the second option, since we thought it could offer better results, although they are harder to implement **[9]** (where the author compared some algorithms (rcnn, fast-rcnn, faster rcnn) for monitoring island birds).
There are different possibilities to implement such an algorithm, and can be divided into one staged detector and two stage detectors.

One stage detector is often faster, but less accurate. The principle of these algorithms is to use a CNN network and change a few things in order to detect objects. This kind of algorithm often looks at several parts of the image, and tries to fit boxes in that part. How this process is done is what differentiate the algorithms between them. This kind of algorithm doesn't feature any region proposal step, and that is why they are faster. The Yolo algorithm **[10]** was one of the first neuronal networks we stumbled into when we did some research. However, we found out that another algorithm, single shot detector (SSD) seems to give better results. We used it with some success with the resnet50 CNN **[11]**.

Two stage detectors, on the other hand, have proven to be harder to implement. It features two networks, who are divided by a ROI pooling layer. For example, in Faster-rcnn **[12]**, the region proposal network proposes candidate bounding box, and the second part extracts features with ROI pooling operation for each box and performs classification as well as bounding box regression. The two parts share the same convolutional features (explaining why they can’t be seen as two separate networks). This was used in **[13]** with good accuracy, for example. The authors manage to have a similar object detection score than human specialists, and have shown that the use of both specialist and IA allows the specialist to be more time efficient and more reliable (for bird detection on image).

More details about the state of the arts object detection algorithm and the differences between one and two stage detectors can be found in **[14]**.

Our code

In order to implement this algorithm, we used tensorflow because it features an API of object detection. We aimed at training a network for the algorithms proposed by this library. In order to do that, we found a website **[15]** that explains the process of installing tensorflow and the API and training the network. In order to perform inference, we also used a jupyter notebook because it allows us to plot images. We implemented the ssd-resnet50 algorithm because the faster-rcnn algorithm couldn't run on our computer.

I. Before launching the algorithm

In order to use the image in the algorithm, we needed to have the information about where there are located and what is each object. For that, we used a .pbtxt file. There is also a class file included (.label). It must begin at 1, but for the reason that we'll explain later we've put 16 classes in that file. The .config file is used to set up the algorithm. When the image is fed to the algorithm, it is resized (when can see the final size in the .config file).

II. Training.

In order to train our network more efficiently and get better results, we used a pretrained network. To do so, we needed to import the weights corresponding to our model. We can find these weights in the zoo of the API object detection.

Once the model is downloaded, we set up the configuration file, and launch the training with a file (.config) provided in this API (see tuto). This file allows us to configure the different learning values as well as some other parameters. It also features automatic data augmentation (but adding more data augmentation is possible).

If we have less classes than the original model (we have 1 against 90 for the original model), the algorithm does not load the weight of the output corresponding to these classes. However, we didn't manage to choose the class that will be conserved, which is a shame because the pre-trained model features a class corresponding to birds (class 16). It's why we've decided to include 15 fake classes, and to use only the 16th.*

However, by looking deeper on the programs, it seems that training for ssd-resnet50 in tensorflow API object detection either restores the classification parts, or the parts responsible for the region. I think that when I trained my ssd-resnet50, the classification part was not restored.

By doing so, we've managed to train the SSD-Resnet with moderate success (see results). However, It seems like all the weights are not restored. we have tried to fix this issue in order to get better results, without success.

The training took a long time (around 10h), but restoring all the weight should improve training time.

III. Inference.

We’ve reused a notebook provided in the API object detection in order to perform inference. It requires the labels of the classes, and the trained model. However, this algorithm can’t be used to load a model into tensorflow (which could be useful to transform the model) because it is not fully loaded when we perform inference. This notebook can use downloaded models from the API detection object zoo as well as exported trained models.

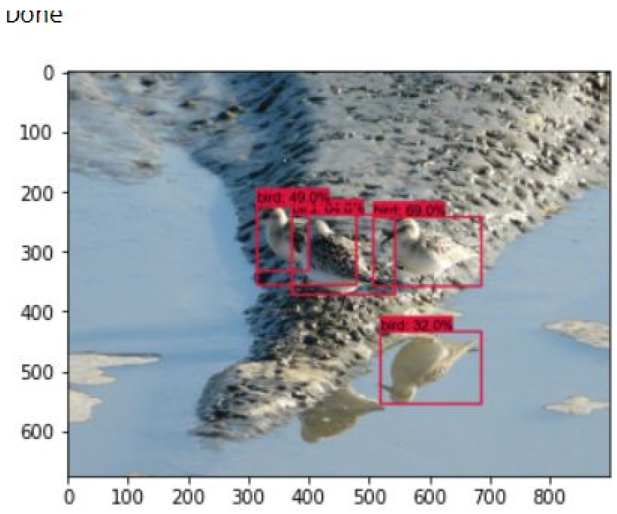
Results

Part 1: application of pre-trained ssd-resnet50 to ebirds image

The purpose of this part is to assess the feasibility of bird detection. In particular, there is the question of detecting birds that camouflage well in the landscape, as well as birds that are close to each other.

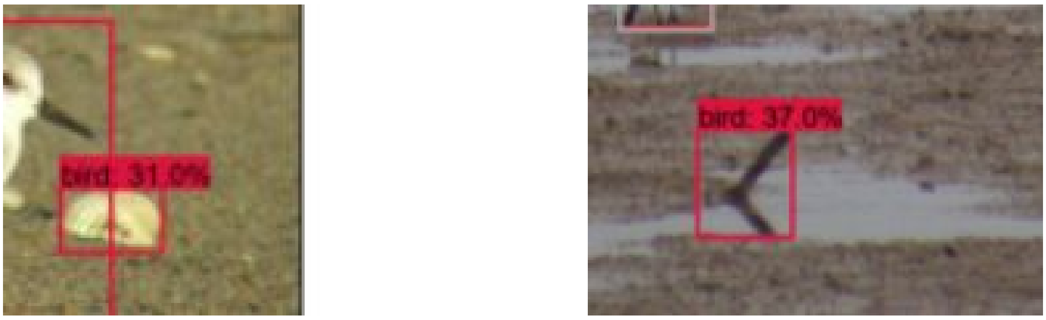
Below are somes of our results. In order to be shown, the pourcentage associated with the bounding box must be over 30%.

1) False detection:

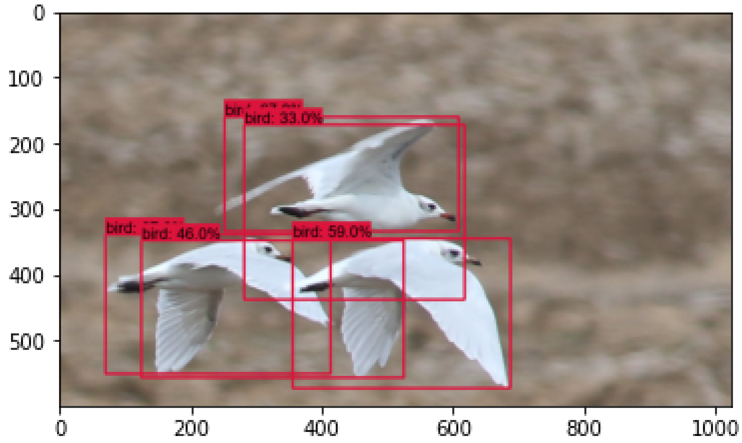


```
Done
```

false detection: reflection

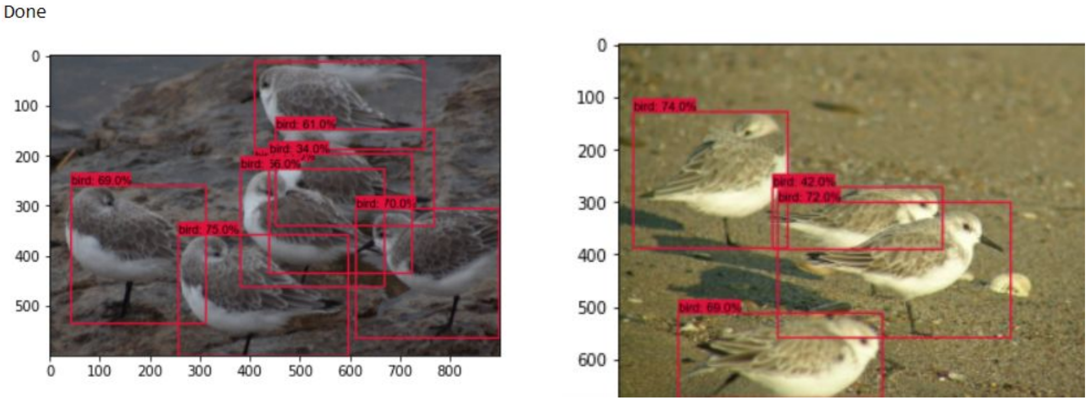


false detection : “similar” element

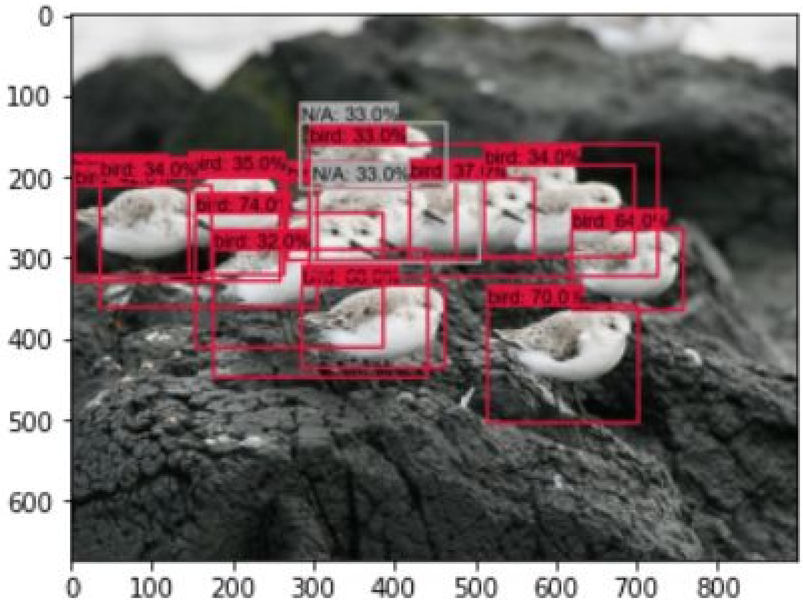


Bird too big and detected as multiple

2) packed birds



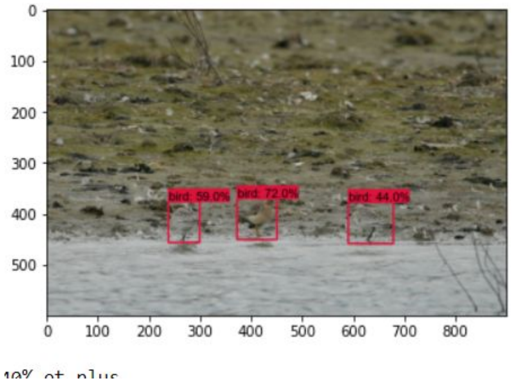
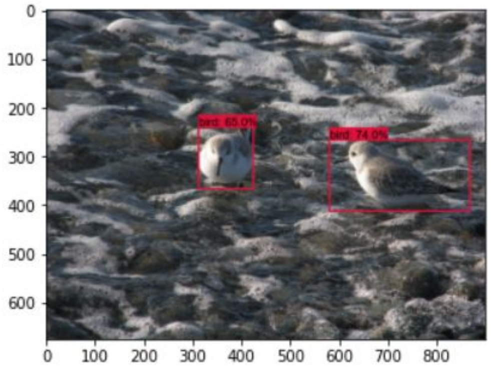
Moderately packed


Really packed

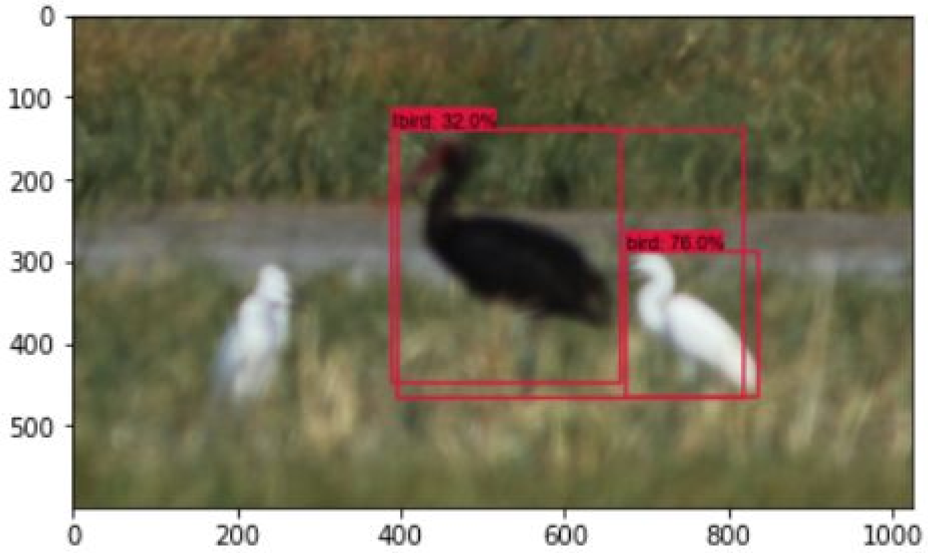
In some cases, the pre-trained resnet will propose too many regions, counting two or three times the same bird.

3) environnement

Species: Sanderling

	
Flatland	**water**

4) Photo sharpness



Result on blurred photo

5) Conclusion

To conclude, we have seen that this neuronal network can allow us to detect birds. The neuronal network manages to detect birds even in cases where they seem similar to the background. The neuronal network also manages to count birds when they overlap. In cases where the birds are a bit packed together, the neuronal network still manages to separate them. Even on blurred photos, the network manages to detect the birds.

However, in all these cases, the detection is more difficult, which means that the network is not confident enough to differentiate these detection with fake detection. Indeed, when the bounding box is around 30% to 40% confidence, they are in a situation where a bounding box does not correspond to a bird and some situation where it does. Increasing the percentage necessary to consider a bounding box as a bird will then lead us to less fake bounding box, but will also mean that we will not detect some birds.

In the second part, we will see if better training can help us to solve this issue.

Part2: Using AU birds Dataset

Link to dataset: https://www.kaggle.com/aubird/au-bird-scene-photo-collection/version/1

a. Purpose

but de cette partie : évaluer la faisabilité d'entraîner un réseau de neurones. Utilisation d'un dataset avec des oiseaux volants pour avoir beaucoup de photos. ici, utilisation de ssd-resnet50.

b. Evaluation of the results

The evaluation of the performance of the model can be given by the AP and AR (Average Precision and Average Recall) **[16]**. This metric is the mean of true positive divided by the sum of the true positive added with false positive (the number of true positive change in function of the tolerance that we give for a box to be a true positive). AP or average precision is a metric about how certain we are that a given result of the algorithm corresponds to the reality it is calculated by dividing the true positive by the sum of true positive and false positive. AR or average recall is a metric about how certain we are that a given bird will be detected. It is calculated by dividing the true positive by the sum of true positive and false negative.

Accuracy for each Version of the Resnet

	Pre-trained Resnet	Resnet 14k Steps	Resnet 21k Steps
Average Precision (AP)	50,0 %	37,2 %	46,4 %
Average Recall (AR)	48,5 %	37,1 %	43,0 %

As we can see, the performance of the algorithm doesn't exceed the pre-trained resnet accuracy. This is explained by the fact that in API object detection, all the neuronal network is not restored. This result also hides the fact that the obtained result greatly varies in function of the condition of the birds (in some conditions, they are always detected, and in some conditions, never detected).

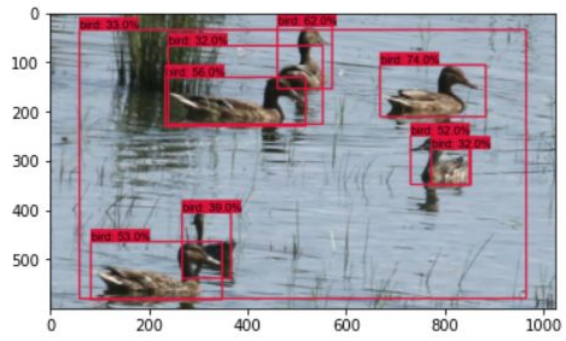
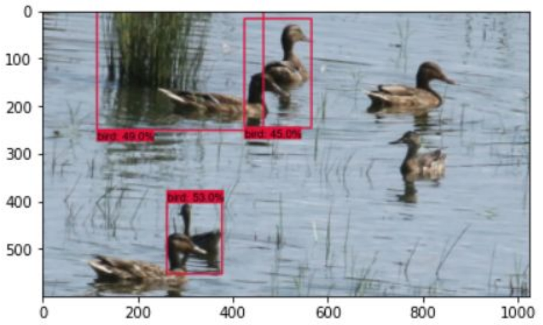
However, this result proves that we are able to train ssd-resnet50 to detect birds.

C. Detailed review of the results

Below are somes of our results. In order to be shown, the pourcentage associated with the bounding box must be over 30%.

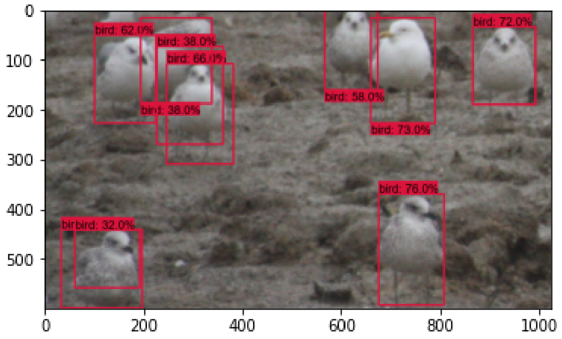
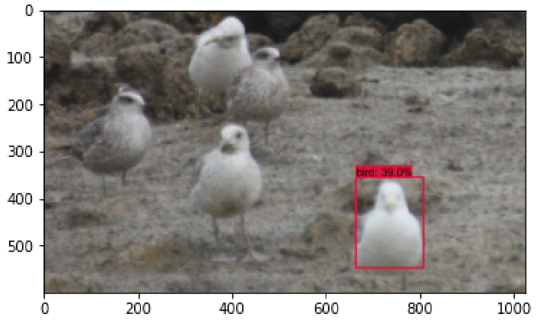
1) Better pre-training

Water environment:

 bird: 33.0% bird: 62.0% bird: 32.0% bird: 56.0% bird: 74.0% bird: 52.0% bird: 32.0% bird: 39.0% bird: 53.0% 0 100 200 300 400 500 0 200 400 600 800 1000	 bird: 49.0% bird: 45.0% bird: 53.0% 0 100 200 300 400 500 0 200 400 600 800 1000
birds detection by the Pre-trained SSD-Resnet	**Birds detection by the Trained SSD-Resnet**

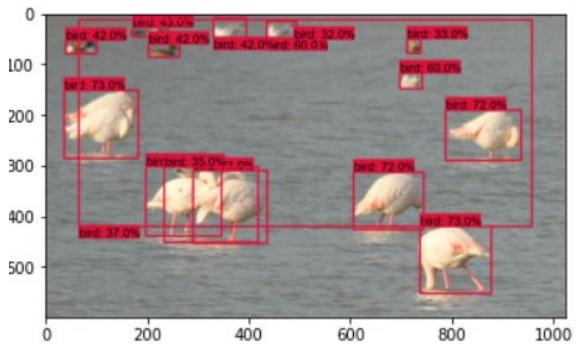
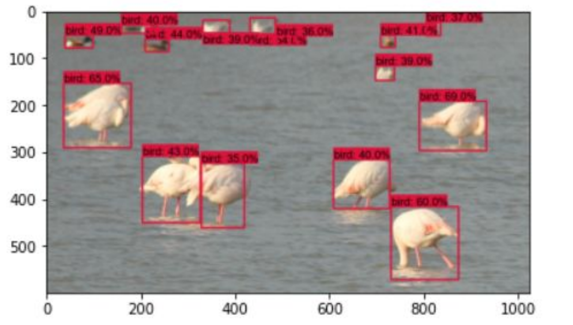
Number of similar photo in the train dataset : 2 (10 cases)

Seagulls on earth:

 bird: 62.0% bird: 38.0% bird: 66.0% bird: 38.0% bird: 58.0% bird: 73.0% bird: 72.0% bird: 76.0% birbird: 32.0% 0 100 200 300 400 500 0 200 400 600 800 1000	 bird: 39.0% 0 100 200 300 400 500 0 200 400 600 800 1000
birds detection by the Pre-trained SSD-Resnet	**Birds detection by the Trained SSD-Resnet**

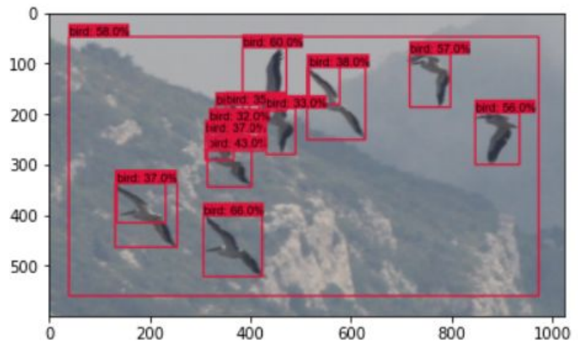
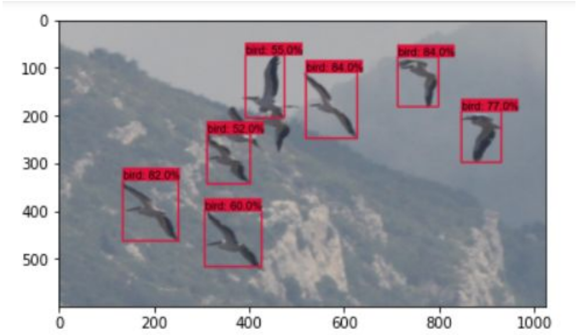
Numbers of similar photo in the train dataSet : 10-20

2) Equivalent training

 0 100 200 300 400 500 0 200 400 600 800 1000 bird: 42.0% bird: 42.0% bird: 42.0% bird: 60.0% bird: 32.0% bird: 33.0% bird: 60.0% bird: 73.0% bird: 72.0% bird: 35.0% bird: 72.0% bird: 37.0% bird: 73.0%	 0 100 200 300 400 500 0 200 400 600 800 1000 bird: 40.0% bird: 49.0% bird: 44.0% bird: 39.0% bird: 36.0% bird: 37.0% bird: 41.0% bird: 39.0% bird: 65.0% bird: 69.0% bird: 43.0% bird: 35.0% bird: 40.0% bird: 60.0%
Birds detection by the Pre-trained SSD-Resnet	**Birds detection by the Trained SSD-Resnet**

Numbers of similar photo in dataSet : 10-20
- 7 pictures for the flamingo (30 birds)
- 3 pictures for the birds far away (31 birds)

3) In the air

 0 100 200 300 400 500 0 200 400 600 800 1000 bird: 58.0% bird: 60.0% bird: 38.0% bird: 57.0% bird: 33.0% bird: 32.0% bird: 37.0% bird: 43.0% bird: 56.0% bird: 37.0% bird: 66.0%	 0 100 200 300 400 500 0 200 400 600 800 1000 bird: 55.0% bird: 84.0% bird: 84.0% bird: 77.0% bird: 52.0% bird: 82.0% bird: 60.0%
birds detection by the Pre-trained ssd-Resnet	**Birds detection by the Trained ssd-Resnet**

Numbers of similar photo in dataSet : 10-20 :
A lot of similar flying birds photo (more than 400 showing flying birds) and very similar photos

D. Results interpretation

Both resnets are able to detect birds, as shown in these two photos above. However, these results are variable, in some cases the trained ssd-resnet50 performs well and in some other cases it performs badly.

We can identify that in cases where the number of pictures are sufficient, the training seems to work.

In particular, the percentage of confidence that a bounding box is a bird increases, allowing for better slicing in more complicated cases. There also seems to be fewer birds detected twice.

However, we note that in the case of gulls, the training did not work. This may be related to the fact that gulls are too different from other images of birds in flight, or that the task is made more difficult by the environment in which they are found, which is less present in the images. We notice moreover that the trained algorithm does not give any result for birds that are close to each other. However, the database presents few instances of this case, and this may be a reason for this poor performance. The use of another neural network more powerful (faster rcnn) could also benefit these cases.

Bibliographie

Number	Subject	Link
1	Convolutional Neural Network	Course: https://cs231n.github.io/convolutional-networks/ Video: https://www.youtube.com/watch?v=bNb2fEVKeEo&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=5
2	Pre-trained models	https://pytorch.org/docs/stable/torchvision/models.html
3	Unfreezing the pre-trained model	https://keras.io/guides/transfer_learning/
4	Coco	https://cocodataset.org/#explore
5	Preprocessing methods	https://ijcsmc.com/docs/papers/May2014/V3I5201499a84.pdf
6	Using contours for bird detection with Opencv	https://github.com/MichaelTeti/BirdDetectorCNN?fbclid=IwAR3p8pNJj6tnNT2yc371RtXm7NAmP4-SHc89HdYF3qjGmgr_hkyCz5hLrc8
7	Selective Search	https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf

8	Fast RCNN	https://arxiv.org/abs/1504.08083
9	Multi-Background Island Bird Detection Based on Faster R-CNN	https://www.tandfonline.com/doi/abs/10.1080/01969722.2020.1827799
10	YOLO	https://arxiv.org/abs/1506.02640
11	SSD	https://arxiv.org/abs/1512.02325
12	Faster-RCNN	https://arxiv.org/abs/1506.01497
13	Automated Bird Counting with Deep Learning for Regional Bird Distribution Mapping	https://www.mdpi.com/2076-2615/10/7/1207
14	A Survey of Deep Learning-based Object Detection	https://arxiv.org/abs/1907.09408
15	Tuto for implementing object detection network from the tensorflow api	https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html
16	Average precision and average recall	https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173