Project PNM-UCA

Part I Application on sample of videos (24)

Part II Application on all videos(videos_mercantour_23_06_2020)

Part I Application on sample of videos (24)

1st step: Treatment of videos sample

- 24 videos: 12 by day / 12 by night.
- Videos by day: 30 seconds and Videos by night: 20 seconds.
- Videos are split all five thenth seconds into images.
- We obtained 1,223 images of size 1920x1080 pixels.
- 61 images by day video.
- 41 images by night video (except one video shorter than others with 40 images).

Script used *"Projet_PNM/scripts/sample_24_videos/**split_videos.ipynb**",* Images are saved in file "*Projet_PNM/data/sample_24_videos/**picture***"

2nd step: Treatment of images from videos sample

Aim: keep only images with animals
Two methods: manual and automatic.
Manual method:

- 1,223 images sorted manually.
- Classified in 3 categories: empty, one species, >one species.

Images are classified in file "*Projet_PNM /data/ sample_24_videos* /**sample**"

Automatic method:

- For all images by day and night.
- Adjust according to the number of images by video.
- Remove 1st image of each video → brightness problem (fox and ibex videos).
- We scale images, put in black and white and resize images into 224x224 pixels. If we conserved original size the code doesn't work, we put in black and white because in literature they did that.
- Calculated the percentage of variation of pixels between each image and the last one image of a video. To do that we sum the absolute difference of pixels between each image of a video and the last image of the same video (called background). We supposed, the last image doesn't contained animals. We supposed a maximum of variation of pixels correspond to detect an animal. We retained only pictures which have 70% of variation of pixels compared to the last one image of the video (background).

Script used is "*Projet_PNM/scripts/sample_24_videos/***select_picture.ipynb**", images are saved in file "*Projet_PNM /data/sample_24_videos/***picture2**".

The differences between the two methods are in file
"*Projet_PNM /analysis/***manual_vs_auto_img.xlsx**".

3rd step: CNN with VGG16 on manual and automatic sample

- We resize images of size 224x224x3 and scale the images.
- Train (80%) - Validation (10%) - Test (10%)
- I tried VGG16 and VGG19 for manual sample: I obtained approximately the same results with VGG16 and VGG19 so we keep only results of VGG16. I check if there are only images with animals in the test part with the name of images of manual sample. We obtained results for test data with all images and with only images of animals.
- For automatic sample I used only VGG16 and for the test part I considered only pictures with animals. If pictures are empty in test part they are removed.
- We used transfer learning: the weight of the 19 layers are fixed according to the imagenet dataset.
- We define the last three layers with: flatten, fully connected layer + relu activation function and fully connected layer + softmax activation function.
- The modellings are trained only on the 3 last layers.
- The model is compiled with loss function “categorical crossentropy” because the output is categorical. The optimizer is Adam with learning rate 0.0001, I tested different optimizer like SGD or RMSprop and different values of learning rate and I obtained not better results. Adam looks like the better in the literature. The metrics is accuracy because we calculated probabilities.
- The train part is with 100 epochs and we used early stopping with patience of 4 for value of accuracy. The train part stops when the value of accuracy is constant during four epochs.
- For evaluate our models: loss functions/ classification report/ confusion matrix/ evaluate on test data/ prediction on test data.
- Modelling are running 20 times in order to be sure of our results.
- The results are represented with boxplot of loss, accuracy and prediction.
- We tested with automatic sample VGG16 20 times within TRAIN/VALIDATION/TEST parts the same images of the same video. Only 2 species are in 3 videos (loup and lievre-variable). In test part there is only images with animals.
  The repartition of the videos is:

Train (486 images de 14 vidéos, 77 %) :

1. maille56_ecureuil_x1_jour.MP4
2. maille57_martre_x1_nuit.MP4
3. maille86_blaireau_x1_nuit.MP4
4. maille86_sanglier_x1_nuit.MP4
5. maille54_bouquetin-male_x1_jour.MP4
6. maille86_cerf_x2_jour.MP4
7. maille07_chamois_x22_jour.MP4
8. maille55_chevreuil_x1_jour.MP4
9. maille86_biche_x4_jour.MP4
10. maille86_humain_x2_jour.MP4
11. maille22_renard_x1_jour.MP4
12. maille39_lievre-d'Europe_x1_nuit.MP4
13. maille103_loup_x2_jour.MP4
14. maille55_lievre-variable_x1_nuit.MP4

Validation (131 images de 8 vidéos, 21 %) :

1. maille07_chamois_x1_jour.MP4
2. maille24_chevreuil_x1_nuit.MP4
3. maille87_biche_x1_jour.MP4
4. maille55_chien+humain_jour.MP4
5. maille55_renard_x1_nuit.MP4
6. maille41_lievre-d'Europe_x1_nuit.MP4
7. maille55_loup_x2_nuit.MP4
8. maille54_lievre-variable_x1_nuit.MP4

Test (11 images de 2 vidéos, 2 %) :

1. maille41_loup_x1_nuit.MP4
2. maille39_lievre-variable_x1_nuit.MP4

Script used are "*Projet_PNM /scripts/sample_24_videos/***CNN_manual_sample.ipynb**" for CNN on manual sample and "*Projet_PNM /scripts/sample_24_videos/***CNN_auto_sample.ipynb**" for CNN on automatic sample.

4th step: Background segmentation (error)

Aim: VGG16 on images without background.

- Selection of backgound images (last images) of each video.
- We applied on manual and automatic sample.
- We resize (224x224x3) and scale image.
- Calculate the absolute difference between each image and the background of each image and we obtained images without background
- We apply VGG16 on these pictures without background like 3rd step.
- The results are not very good because the weights are adjusted on images with background.

Parts I/II/II of script "*Projet_PNM/scripts/sample_24_videos/***background_segmentation.ipynb**".

5th step: Bounding boxes (BB) & Crop images

Aim: Crop images thanks to bounding boxes in order to focus on animal with its background and improve our CNN.

- Firstly, we determine BB. 3 methods are tested: contours, histogram and MegaDetector.
- For methods of contours and histogram, we keep the original size of pictures. The images are scale and put in black and white. Calculate absolute difference of each image and the background (last image) of videos → detecte animal.
- Then for contours method: with OpenCV package we fixed threshold (0.05), if the difference is less than threshold then value = 0 otherwise =1 (*cv2.threshold()*). Then we apply function to remove noise (*cv2.morphologyEx()*), its remove contours of small differences due to brightness for example. Then contours are determined with function *cv2.findContours().* According to these contours we defined BB. We keep only BB on the same video which have at least area value upper 0.3 x maximum area of BB find on the same video.

- Then for histogram method: exactly same process than contours method but rather than works with contours, it works by detect 1, build BB around group of 1 detected. Functions used are *ndimage.label()* et *ndimage.find.objects()* from *scipy* package.
- I tested different values for threshold and also for % of maximum area to retained, the parameters chosen have best results.
- The results with contours method are less performant than histogram method on images from file "*Projet_PNM/data/sample_24_videos/**picture2***". Histogram method is retained, and we extract crop images from file "*Projet_PNM/data/sample_24_videos/**picture***" thanks to this method. The BB are cropped in the real image (not black and white).
- For the last method called "Megadetector", BB are determined according to code of Microsoft. The current model is based on Faster-RCNN with an InceptionResNetv2 base network. Links of their code are in the script of this part. Their code is applied on file "*Projet_PNM/data/sample_24_videos/**picture***" and "*Projet_PNM/data/sample_24_videos/**picture2***". The results are Json files in "*Projet_PNM/data/sample_24_videos/**megadetector***". We cropped real images by coordinates of BB with confidence threshold >=0.9.
- Summary: The best results are obtained with megadetector method, we retained results on file "*Projet_PNM/data/sample_24_videos/**picture***" (they are more images with animals, and they are the background images).
- After determined BB for each video with histogram and megadetector method, we calculate the number of species according to the maximum number of BB find in a same video.

Parts IV and VI of script
"*Projet_PNM/scripts/sample_24_videos/***background_segmentation.ipynb**".
Cropped images from "*Projet_PNM /data/sample_24_videos/**picture***" with histogram method are in file
"*Projet_PNM /data/sample_24_videos/***crop_picture**" and from megadetector method are in file "*Projet_PNM /data/sample_24_videos/***crop_picture_megadetector**".

The results are in file "*Projet_PNM /analysis/**crop_images.xlsx***".

6th step: CNN with VGG16 on crop images

Aim: Improve our CNN.

- Like 3rd step, we applied VGG16 we same parameters on crop images from histogram and megadetector method.
- The results obtained are good, lightly less good than on whole images, but results are similar.
- In order to know if the model learn well, we tested also with not same images from same video in train/val/test. The results are not very good surely due to little number of images.

Parts V of script "*Projet_PNM/scripts/sample_24_videos/***background_segmentation.ipynb**".

The results are in file "*Projet_PNM /analysis/**crop_images.xlsx***".

Part II Application on all videos(videos_mercantour_23_06_2020)

1st step: Formatting data on hard disk

- Files are renamed *malleX* without space between *maille* and the number of the maille *X* and without capital letter.
- Files *maille116, maille117, maille101, maille102* have files with no references in the ods document (no labels), they are moved to a file called « ***echantillon_test*** ».

Files with no references are called for *maille116*: "2020-04-24" and "2020-05-13"
Files with no references are called for *maille117*: "2020-04-24" and "2020-05-13"
File with no references is called for *maille101*: "2020-05-26"
File with no references is called for *maille102:* "2020-05-17"

- Replacement of ods file by the one according to « nomenclature_PNM.docx » called « releve_mars_piege-photo.ods » (same format for all sheets, for all columns, for all labels, names of videos are similar to their references in files maille etc …)
- Sheets *maille39* et *maille37* in ods file are removed, no files existed with these names. In the ods file *maille72, maille118* more references than images, extra references are removed (same for one reference in *maille103*).
- Notice: The sample of videos **Part I** is included on all videos.
- Add news videos and images given 04/12/2020 by PNM: 4 videos and 1 image are added. Videos are added according to the name of the maille and renamed. The photo is associated to no one maille so we created filed « ***maille00*** », it references all images and videos without maille associated. These new data are referenced in the ods file (we do same thing for images given 15/01/2020).
- 7 files with images (maille00, maille23, maille66, maille 69, maille72, maille117, maille119), total of 2588 images no empty with labels.
- 35 files with videos (maille06, maille07, maille08, maille21, maille22, maille24, maille36, maille38, maille40, maille41, maille50, maille51, maille52, maille53, maille54, maille55, maille56, maille57, maille67, maille68, maille70, maille71, maille73, maille84, maille85, maille86, maille87, maille88, maille100, maille101, maille102, maille103, maille104, maille116, maillle118), total of 1744 videos no empty with labels.
- 5 species: "chat", "gant", "perdrix", "chacal-dore", "lynx" are only in files with pictures, we don't have any videos of these species.
- Only 4 % of videos have multiple species on same video (biche_cerf, chamois_crave, humain_chien, renard_biche, renard_cerf, velo_humain, velo_humain_chien, voiture_humain)

The script used is "*Projet_PNM/scripts/**descriptive_statistics.py***"

2nd step: Transfer data of hard disk to server Azzurra

Time on Azzurra: 4 days (via network CHU).
Transfer of file "videos_mercantour_23_06_2020".

3rd step: Split videos into images

Time on Azzurra: 2h25 (2h to create list of videos paths with animals and 25min to split) on CPU (1 node 40 cores).

- Concerned only videos with animals: 1 744 videos. I didn't work with empty videos.
- Use dopa (package MarcoM) to parallelize my python script (part of split videos into images).
- Obtained 87 839 images.

The script used is "*Projet_PNM/scripts/**videos_to_images.py***" or on Azzurra "*/home/fsimoes/scripts_python/**videos_to_images.py***".
The results are on Azzurra "*/workspace/fsimoes/images*".
The name of job is "*/home/fsimoes/jobs/**job1_vidtoimg.sh***".

DETECTION PART

4th step: Apply MegaDetector on images to determine BB

Aim: get sample of our images with BB with threshold fixed.

A) Apply MegaDetector on our images

Time on Azzurra: I used CPU because for using GPU the queue is too long.
- CPU: 2days 1h 30min (1 node 40 cores)
- GPU: 7h 25min(1 node 16 cores)

- We apply MegaDectector on our 87 839 images in order to obtain json file with BB of each image. All the BB are referenced with all thresholds.
- Like part Using models 2.run_tf_detector_batch.py : https://github.com/microsoft/CameraTraps/blob/master/megadetector.md

Tha bash script used on server is
"*Projet_PNM/scripts/detection/***run_megadetector_server.txt**"
The results are on Azzurra
"*/workspace/fsimoes/detection/megadetector/results_megaBB/**megaBB.json***".
The names of jobs are "*home/fsimoes/jobs/**job2_megaBBtfgpu.sh***" and "*/home/fsimoes/jobs/**job2_megaBBtfcpu.sh***".

B) Images with BB

Time on Azzurra: 6min on CPU (1 node 40 cores)

- The BB with threshold of 90 % are drawn on images.
- Use dopa (package MarcoM) to parallelize my python script.

The script used is "*Projet_PNM/scripts/detection/**imgBBmega.py***" or on Azzurra "*/home/fsimoes/scripts_python/detection/**imgBBmega.py***".

The images with their BB th.90% are on Azzurra "*/workspace/fsimoes/detection/megadetector/imgBB_mega/*".
The name of job is "*/home/fsimoes/jobs/**job3_imgBBmega.sh***".

C) Images with BB to videos

Time on Azzurra: 2h 42min on CPU (1 node 1 core)

- We obtained 1 744 videos with BB.
- The script used is in bash.

The videos with their BB are on Azzurra "*/workspace/fsimoes/detection/megadetector/videoBB_mega/*".
The name of job is "*/home/fsimoes/jobs/**job4_imgBBtovidBB.sh***"

Time on local: 5min to transfer 80 videos (videos are in less quality than originals)

- It is impossible to read video on Azzurra server (ffplay and mplayer don't work).
- From 1 744 videos, I extract manually 80 videos representative of different species in our data.
- I watch the 80 videos locally in order to see if the BB are correct. The BB with threshold of 90% are generally good, very good for big species (humain, voiture, cerf, chevreuil …). Less good when specie is far away or run fast or hide or too small (like some birds: crave, becasse-des-bois). <u>We validate threshold of 90% for BB.</u>

The extract of videos (80) with their BB are on Azzurra "*/workspace/fsimoes/detection/megadetector/videoBB_mega_extract*".

D) Treatment of images with BB with multiple labels

Time on Azzurra: 9min on CPU (1 node 1 core)

- **Reminder:** Only 4 % of all videos (69) have multiple species on same video (biche_cerf, chamois_crave, humain_chien, renard_biche, renard_cerf, velo_humain, velo_humain_chien, voiture_humain).
- 2 355 images are associated to multiple labels and have BB with threshold of 90%.
- There are 4 698 BB associated to these 2 355 images.
- Firstly, I create a python dictionary of images with multiple labels and threshold of BB of 90%, I have the link of the image, the BB coordinates, the label and I attribute an id to each BB. Then, each image is exported with their BB and their id. (one image → one BB threshold 90% → one id).

The script used is "*Projet_PNM/scripts/detection/**ident_mutilabel.py***" or on Azzurra "*/home/fsimoes/scripts_python/detection/**ident_multilabel.py***".
The images with their BB th.90% and id associated are on Azzurra "*/workspace/fsimoes/detection/megadetector/rename_multilabels/idBB/*".
The python dictionary is on Azzurra
"*/workspace/fsimoes/detection/megadetector/rename_multilabels/**dict_2label.json***".

The name of job is "*/home/fsimoes/jobs/**job5_BBmultilabel.sh***".

Time on Azzurra: 6min on CPU (1 node 1 core)

- From these images I created a video with ffmpeg with fps=0.5 according to order of id (id1, id2, id3, etc …).
- Video during 2h36min. I watched it and note in excel file/csv file the name of the id and the label associated (one image → one BB threshold 90% → one id).

The video is on Azzurra
"*workspace/fsimoes/detection/megadetector/rename_multilabels/idBB/**vid_idBB2.MP4***
"
The csv file is on Azzurra
"/workspace/fsimoes/detection/megadetector/rename_multilabels/**BBmultilabels.csv**"
and in "*Projet_PNM/data/***BBmultilabels.csv**
The name of job is "*home/fsimoes/jobs/**job6_vidBBmultilabel.sh***".

E) Creation of final sample

Time on Azzurra: 1h15min on CPU (1 node 1 core)

- For the images with multiple labels, the label associated to each id is replaced by the new label associated in csv file (ex: humain_chien → chien).
- If the new label corresponds to "vide" then BB is removed from python dictionary.
- A new python dictionary is created, image with one label is add to the previous dictionary. Finally, we obtained a python dictionary with one image, one coordinate of BB with threshold of 90% and one label associated to this BB (one image → one BB threshold 90% → one label).
- We have 38 158 images with BB with threshold of 90%, 35 803 images are associated to one label and 2 355 images was associated to multiple labels.
- The number of BB th.90% associated to multi label equal to 4 698 when I removed BB associated to label "vide", I obtained 4 626 BB.
- Finally, our final sample, after corrected multiple labels to one label thanks to my csv file, we have 38 158 images associated to 53 203 BB th.90%.
  Number of BB by species:

1. humain : 18 327
2. chamois : 7 823
3. chevreuil : 6 142
4. biche : 5 943
5. cerf : 4 436
6. renard : 4 288
7. blaireau : 1 216
8. loup : 953
9. lievre : 732
10. chien : 715
11. sanglier : 712
12. velo : 709
13. bouquetin : 474

14. martre : 186
15. voiture : 97
16. lievre-variable : 79
17. pigeon : 60
18. ecureuil : 60
19. merle : 58
20. geai-des-chenes : 47
21. epervier : 47
22. grive : 39
23. chouette-tengmalm : 27
24. becasse-des-bois : 18
25. chouette : 7
26. genette : 4
27. crave : 3
28. pinson-des-arbres : 1

Missing species: belette, lezard, papillon.

The script used is "*Projet_PNM/scripts/detection/**sampleimgBB_mega.py***" or on Azzurra "*/home/fsimoes/scripts_python/detection/ **sampleimgBB_mega.py***".
The images (previously associated to multiple labels) with their BB th.90% and new labels associated are on Azzurra "*/workspace/fsimoes/detection/megadetector/rename_multilabels/newlabels/*".
The python dictionary of final sample BB (one image → one BB threshold 90% → one label) is on Azzurra "*/workspace/fsimoes/detection/megadetector /**sample_imgBB.json***" **or on github** "*Projet_PNM/data/**sample_imgBB.json***".
The name of job is "*home/fsimoes/jobs/**job7_sampleimgBB.sh***".

Time on Azzurra: 7min on CPU (1 node 1 core) (A) and 4 sec (1 node 1 core) for (B).

- To check if the new labels associated to multiple labels are correct, I created video from these images. I used ffmpeg with fps=2 (A). After watch part of the video, the news labels are validated.
- I extract an example of images (each species) from the finale sample, in order to create video of sample with BB th.90% and one label associated (with python code). I used ffmpeg for create video with fps=1 (B).

The images of our sample are on Azzurra "*/workspace/fsimoes/detection/megadetector/exvid_sample_imgBB/*".
The video to check new labels is on Azzurra: "*/workspace/fsimoes/detection/megadetector/rename_multilabels/newlabels/**vid_newlab.MP4***".
The video of extract of our sample is on Azzurra: "*/workspace/fsimoes/detection/megadetector/ exvid_sample_imgBB /**vid_sampleBB.MP4***" and in "*Projet_PNM/data/.*

The name of jobs for created videos in order to check news labels "*home/fsimoes/jobs/**job8_checkvidBBmultilabel.sh***" and for created video of example of our sample "*home/fsimoes/jobs/**job7_sampleimgBB.sh***".

5th step: Training our own model on our classes

A) Installation of TensorFlow 1 Object Detection API on Azzurra

Installed version of **TFOD API 1.12.0,** inspired by:
https://github.com/microsoft/CameraTraps/tree/master/detection

For the installation I followed this tutorial: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/tensorflow-1.14/install.html

Use of **python 3.6** and I create two environments conda, one for CPU, one for GPU.

############## INSTALLATION PYTHON + TF GPU

conda create -n pyt36tf112gpu python=3.6
conda activate pyt36tf112gpu
pip install tensorflow-gpu==1.12
pip install pillow lxml jupyter matplotlib cython opencv-python humanfriendly tqdm jsonpickle statistics requests pandas

############### INSTALLATION PYTHON + TF CPU

conda create -n pyt36tf112cpu python=3.6
conda activate pyt36tf112cpu
pip install tensorflow==1.12
pip install pillow lxml jupyter matplotlib cython opencv-python humanfriendly tqdm jsonpickle statistics requests pandas

############### DOWNLOAD TF MODELS

cd /workspace/fsimoes/detection/mymegadetector/tensorflow1
 # git clone https://github.com/tensorflow/models.git doesn't works missing files
 # Download: https://github.com/tensorflow/models/tree/r1.13.0 zip file then transfered with rsync
 # obtained new file "models": /workspace/fsimoes/detection/mymegadetector/tensorflow1/models

 # To modificate line 259 add: list(range(boundaries[0]))
#https://github.com/tensorflow/models/blob/master/research/object_detection/utils/learning_schedules.py
#nano
/workspace/fsimoes/detection/mymegadetector/tensorflow1/models/research/object_detection/utils/learning_schedules.py

############ COCO API INSTALLATION

pip install pycocotools #for pyt36tf112cpu and pyt36tf112gpu

########### PROTOBUF INSTALLATION/COMPILATION

 # Locally on mac: brew install protobuf
 # Otherwise

 # Download: https://github.com/protocolbuffers/protobuf/releases --> protoc-3.15.6-linux-x86_64
cd /workspace/fsimoes/detection/mymegadetector/protobuf
 # rsync protoc-3.15.6-linux-x86_64 in file protobuf
cd /workspace/fsimoes/detection/mymegadetector/tensorflow1/models/research/
export PATH=$PATH:/workspace/fsimoes/detection/mymegadetector/protobuf/bin
protoc object_detection/protos/.proto --python_out=.*

########## INSTALL OBJECT DETECTION

cd /workspace/fsimoes/detection/mymegadetector/tensorflow1/models/research/
pip install .

B) Training custom object detector

The python dictionary of final sample BB (one image → one BB threshold 90% → one label) is on Azzurra "*/workspace/fsimoes/detection/megadetector/**sample_imgBB.json***" **or on github** "*Projet_PNM/data/**sample_imgBB.json***"
The images are on Azzurra "*/workspace/fsimoes/images*".

Tutorial: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/tensorflow-1.14/training.html

########## *FILES*
cd /workspace/fsimoes/detection/mymegadetector/tensorflow1/

```
tensorflow1
|– models
|   |– official
|   |– research
|   |– samples
|   └– tutorials
└– tf1workspace
   |– training_demo
   └– training_uniqueimg
```

```
training_demo
|– annotations
|– images
|   |– test
|   └– train
|– pre-trained-models
|– training1
|– training2
└– readme.txt
```

```
training_uniqueimg
|– annotations
|– images
|   |– test
|   |–train
|   └– val
|– pre-trained-models
|– training1
|– training2
```

├─ training3
└─ readme.txt

Notice: tf1workspace contains all models I tried.

training_demo:

- *1st test I did.*
- The label map contains 25 species until “chouette”.
- Images are separated in 2 parts: train (45 556 BB - 80 %) and test (10 639 BB - 20%). Images are randomly affected in each part and could be in both parts, train and test according to BB. Only images with species until chouette.
  The script used are on Azzurra */home/fsimoes/scripts_python/detection/**train-test_mymega.py** or on Git Projet_PNM/scripts/detection/**train-test_mymega.py***
- The script used to transform csv file to record file are change according to labels selected. The script used are on Azzurra */home/fsimoes/scripts_python/detection/**generate_tfrecord_train-test_mymega.py** or on Git Projet_PNM/scripts/detection/**generate_tfrecord_train-test_mymega.py***

- training1 =
  modificate /home/fsimoes/scripts_python/detection/**model_main_megadetector.py**
  → save_checkpoints_steps = 200
  modificate *mymega.config*
  → *num_classes: 25*
  → num_steps: 2000
  → eval_config{} : num_visualizations: 30, min_score_threshold = 0.5
  modificate *job12cpu_training_demo1.sh*
  → sample_1_of_n_examples = 10
  Running 6h cpu around 800 steps
- training2 =
  modificate /home/fsimoes/scripts_python/detection/**model_main_megadetector.py**
  → save_checkpoints_steps = 200 steps
  modificate *mymega.config*
  → *num_classes: 25*
  → num_steps : 3000
  → eval_config{} : num_visualizations: 30, min_score_threshold = 0.5
  → shuffle = false
  modificate *job12cpu_training_demo2.sh*
  → sample_1_of_n_examples remove
  Running 11h cpu around 400 steps

training_uniqueimg:

- 2nd test I did.
- The label map contains 13 species until “bouquetin”.
- Images are separated in 3 parts: train (29976 img – 42084 BB - 80 %), validation (3705 img - 5184 BB - 10%), test (3743 img – 5202 BB - 10%).
  Images are <u>not</u> randomly affected in each part, an image could not be in different parts, the images are unique for each part. Train-val-test according to images.

The script used are on Azzurra */home/fsimoes/scripts_python/detection/**train-val-test_mymega_uniqueimg.py** or on Git Projet_PNM/scripts/detection/**train-val-test_mymega_uniqueimg.py***

- The script used to transform csv file to record file are change according to labels selected. The script used are on Azzurra */home/fsimoes/scripts_python/detection/**generate_tfrecord_train-val-test_mymega_uniqueimg.py** or on Git Projet_PNM/scripts/detection/**generate_tfrecord_train-val-test_mymega_uniqueimg.py***

- training1 =
  modificate /home/fsimoes/scripts_python/detection/**model_main_megadetector.py**
  → save_checkpoints_steps = 200
  modificate *mymega.config*
  → *num_classes: 13*
  → num_steps: 2000
  → eval_config{} : num_visualizations: 30, min_score_threshold = 0.5
  → shuffle = false
  modificate *job12cpu_training_uniqueimg1.sh*
  → sample_1_of_n_examples = 10
  Running 6h cpu around 2000 steps (save problem, I don't know why)
- training2 =
  modificate /home/fsimoes/scripts_python/detection/**model_main_megadetector.py**
  → save_checkpoints_steps = 200
  modificate *mymega.config*
  → *num_classes: 13*
  → image_resizer {
  fixed_shape_resizer {
  height : 1080
  width : 1920
  }
  → num_steps: 2000
  → remove keep_checkpoint_every_n_hours: 1.0
  → eval_config{} : num_visualizations: 40, min_score_threshold = 0.5
  → shuffle = false
  modificate *job12cpu_training_uniqueimg2.sh*
  → sample_1_of_n_examples = 10
  Running 6h cpu around 600 steps (batch_size doesn't work for 32/30/40/50 I don't know why)
- training3 =
  modificate /home/fsimoes/scripts_python/detection/**model_main_megadetector.py**
  → save_checkpoints_steps = 200
  modificate *mymega.config*
  → *num_classes: 13*
  → num_steps: 4000
  → remove keep_checkpoint_every_n_hours: 1.0
  → eval_config{} : num_visualizations: 30, min_score_threshold = 0.5
  → shuffle = false

modificate *job12cpu_training_uniqueimg3.sh*
→ sample_1_of_n_examples = 10
Running 12h cpu 4000 steps

The repartition of species in train-val-test for training_demo and training_uniqueimg is in excel file *Projet_PNM/analysis/**species_repartition.xlsx*** **.**

To do each time to train a model:

########### TRANSFERT ORIGINAL MEGADETECTOR MODEL

 # use rsync for transferred "MegaDetector v4.1, 2020.04.27":
https://github.com/microsoft/CameraTraps/blob/master/megadetector.md
 # transfert original model in file pre-trained-models

########### CREATE LABEL MAP

 # in annotations
nano label_map.pbtxt

########### PARTITIONING THE IMAGES

scripts classify images (train-validation-test) in images and create csv files save in annotations for each part which contains names of images, labels, BB coordinates of MegaDetector (carefull to format see description in their website github), width, height and real coordinates of BB.

 # script in /home/fsimoes/scripts_python/detection/
 # training_demo: train-test_mymega.py
 # training_uniqueimg: train-val-test_mymega_uniqueimg.py
 # notice for run these scripts you have to use recent version of python not 3.6

sbatch job10 (dopa, 40 coeurs 1 nodes 3min, name according to python script used)

########### CREATING TF RECORDS (CONVERT CSV TO RECORD)

scripts transform csv files save in annotations to record files in annotations.

#script in /home/fsimoes/scripts_python/detection/
 # training_demo: generate_tfrecord_train-test_mymega.py
 # training_uniqueimg: generate_tfrecord_train-val-test_mymega_uniqueimg.py

sbatch job11 (name according to python script used, 30 min)

########### CONFIGURING TRAINING PIPELINE

This file allows to configure the parameters of our model. You have to define number of classes, links to images and ckpt file.

*#Copy of **md_v4.1.0.config** in file pre-trained-models to training*
#Called it mymega.config
nano mymega.config

########### TRAIN MODEL

 # Download model_main of Megadetector in script in /home/fsimoes/scripts_python/detection/
 # https://github.com/microsoft/CameraTraps/blob/master/detection/detector_training/model_main.py
 # /home/fsimoes/scripts_python/detection/model_main_megadetector.py
*# Projet_PNM/scripts/detection/**model_main_megadetector.py***
 # arrange some parts, change value of checkpoint for example if you need

 # training_demo: job12cpu_training_demo
 # training_uniqueimg: job12cpu_training_uniqueimg
 # to change parameters according to which model you want to train

C) Visualize results via TensorBoard

After training our models we can visualize results (mAP, loss function, AR, evaluation).
It is not possible to use TensorBoard on Azzurra, so I used it locally after imported results from server Azzurra.

On terminal
conda activate pyt36tf112cpu
Tensorboard – logdir= link_to_model_train --host localhost

6th step: Export our trained model PNMdetector

Tutorial: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/tensorflow-1.14/training.html#exporting-a-trained-inference-graph

We exported best model among all models tested. The best model (at the moment) is training_uniqueimg training1. We decided to save this model after 2000 steps (ckpt-2000).

The model saved is on Azzurra:
"*/workspace/fsimoes/detection/*mymegadetector/tensorflow1/tf1workspace/training_uniqueimg/training1/**export_model/output_inference_graph_2000.pb**"
The script used is "*Projet_PNM/scripts/detection/**export_inference_graph.py***" or on Azzurra "*/home/fsimoes/scripts_python/detection/**export_inference_graph.py***".
The name of job is "*home/fsimoes/jobs/**job13_export_tu1.sh***" (1 min).

COUNTING PART

7th step: Apply our own PNMdetector and counts animal

In this part, it describes only tests of count species and watch if PNMdetector works fine. If you want to count species and apply our model see next section.

Firstly, 10 videos are selected representative of each species detected thanks to PNMdetector. The exported model is applied on these videos and finally species are count by videos.

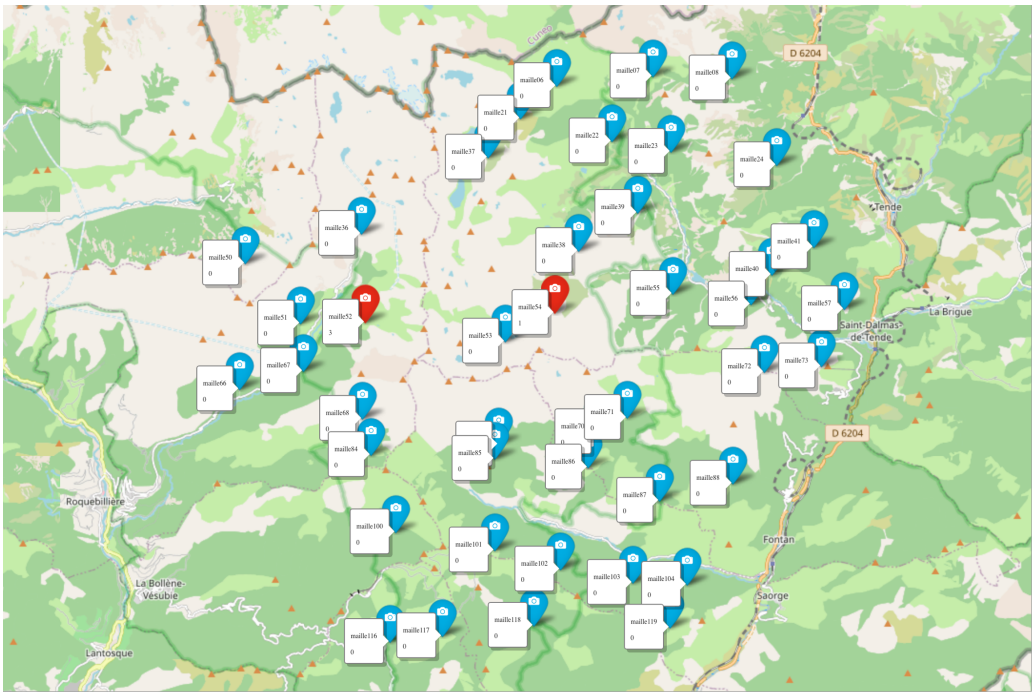
To obtain good detection for all species, a threshold of 0.6 is fixed. When we tried 0.5, lot of false detections were retained whereas when it was fixed to 0.7 few true detections were retained. Therefore, the threshold 0.6 seems to be the best option. So, if detection_scores are < 0.6, we consider images are empty (detection_classes = 0 = "vide"). Firstly, we calculate number of species detected by video and the number of frames where species were detected. Secondly, we count total number of frames on videos. If a specie represents less than 10 % of image, then it is a false detection, we remove this specie. Finally, we obtained csv file which summary number of species by video (this method is upade in further section).

Globally, PNMdetector seems to work fine for detection but for classification results are not very good, it seems to work very well only for big species like humain (around detection_scores > 90%), chamois, chevreuil and renard.

The script used are on Azzurra
*/home/fsimoes/scripts_python/counting/**mymegadetector_count.py*** *or on Git*
*Projet_PNM/scripts/counting/**mymegadetector_count.py***
The results are saved on Azzurra */workspace/fsimoes/counting/**final_number_species2.csv***
and visualization results *are /workspace/fsimoes/counting/**img_BB_mymeg/***
The name of job is "*/home/fsimoes/jobs/**job14_count.sh***".

Secondly, maps are realized to show repartition of species at fixed date. The number of chamois, chevreuil and humain is calculated for each maille to a fixed date (date de relevé 2020-03-10). It was a mistake, for the further steps we work with real dates of videos. These maps are done for species best classified. PNM give us excel file which references the localisation of each camera (*/workspace/fsimoes/videos_mercantour_23_06_2020 **/coordonnees_pieges-photos_2020_4.ods***).

Here example of map for Chamois (so with wrong dates and not last version of couting …) :



The script used are on Azzurra
*/home/fsimoes/scripts_python/counting/**mymegadetector_countmapchamois.py** or on Git Projet_PNM/scripts/counting/**mymegadetector_countmapchamois.py***
The script used are on Azzurra
*/home/fsimoes/scripts_python/counting/**mymegadetector_countmapchevreuils.py** or on Git Projet_PNM/scripts/counting/**mymegadetector_countmapchevreuil.py***
The script used are on Azzurra
*/home/fsimoes/scripts_python/counting/**mymegadetector_countmaphumain.py** or on Git Projet_PNM/scripts/counting/**mymegadetector_countmaphumain.py***

The results are saved on Azzurra:
*/workspace/fsimoes/counting/**final_number_species_humain_2020-03-10.csv***
*/workspace/fsimoes/counting/**final_number_species_chamois_2020-03-10.csv***
*/workspace/fsimoes/counting/**final_number_species_chevreuil_2020-03-10.csv***

The name of job are Azzurra
"*/home/fsimoes/jobs/**job14_count_chamois.sh***",
"*/home/fsimoes/jobs/**job14_count_chevreuil.sh***",
"*/home/fsimoes/jobs/**job14_count_humain.sh***".

The script used *on Git Projet_PNM/scripts/counting/**map.R***

SUMMARY RESULTS

PNMdetector V1.0

To train model:

- If you use new images, go to file *Projet_PNM/**annotated_pnmdetector***
  1st: Split your videos into images
  2nd: Annotated your images
- Then whatever you use new images or not go to file
  Projet_PNM/**train_pnmdetector**

The current model is based on MegaDetector model V4.1. (https://github.com/microsoft/CameraTraps/blob/master/megadetector.md), based on Faster-RCNN with an InceptionResNetv2 base network and a Softmax layer. 13 species are retained: humain, chamois, chevreuil, biche, cerf, renard, blaireau, loup, lievre, chien, sanglier, velo and bouquetin. The model is trained on 53,203 bounding boxes correspond to 38,158 images. Images are different in train-validation-test part.

To apply model:

- Go to file Projet_PNM/**apply_pnmdetector**

At the moment, it is only applicable on files which contains video. The model applied is the last model trained in the last section.

Method to count species :
To obtain good detection for all species, a threshold of 0.6 is fixed. When we tried 0.5, lot of false detections were retained whereas when it was fixed to 0.7 few true detections were retained. Therefore, the threshold 0.6 seems to be the best option. So, if detection_scores are < 0.6, we consider images are empty (detection_classes = 0 = “vide”). Firstly, we calculate number of species detected by video and the number of frames where species were detected. Secondly, we count total number of frames on videos. If a specie represents less than 10 % of image, then it is a false detection, we remove this specie. Concerning videos, if 100 % of frames are detection_classes=“vide” then videos are named “vide”. We have create a CSV file summary, listing every species’ detections on videos, for each video we classify that way : nom, maille, date, heure, espece, nombre d’espèce.

Globally, PNMdetector seems to work fine for detection but for classification, results are not very good, it seems to work very well only for big species like humain (around detection_scores > 90%), chamois, chevreuil and renard.

To realise map of repartition of species you can use ods file with localisation of camera by maille: */workspace/fsimoes/videos_mercantour_23_06_2020/**coordonnees_pieges-photos_2020_4.ods***

Improvement axes:

To train model:

- Change configuration of model: pnmdetector.config (replace softmax, batch size, steps, metric_set, data augmentation, size images, optimizer).
- Change classes (add news classes or create classes which gather multiple species).
- Train with more images.
- Images train-validation-test from different video or maille.

To apply model:

- Possibility to apply PNMdetector on files which contain only images.
- Method of counting (when two species are not present at same moment, indicate false detection, change threshold fixed).