MASTER OF SCIENCE
IN ENGINEERING

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

MLBD
Crapauduc

Martin Spoto
martin.spoto@master.hes-so.ch

Romain Silvestri
romain.silvestri@master.hes-so.ch

June 28, 2021

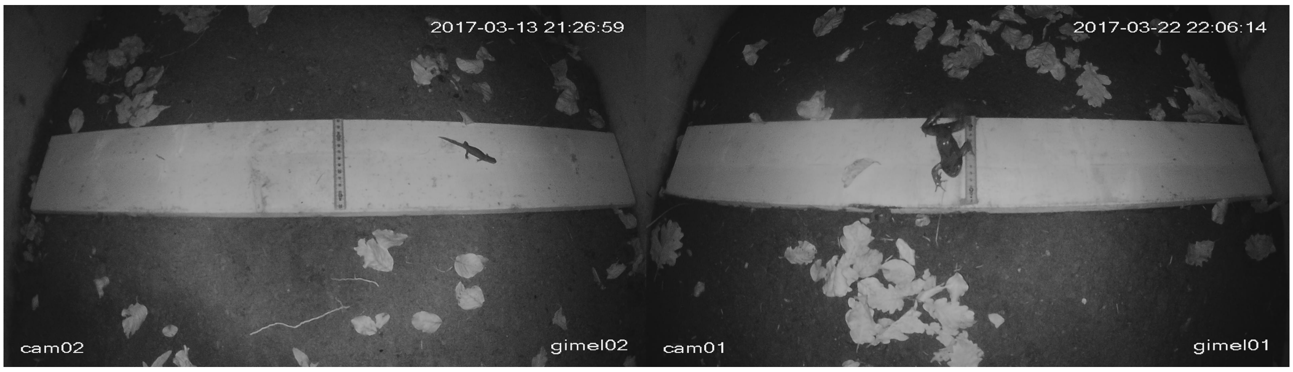
1 Context and objectives

In Gimel (VD), a tunnel was built to allow toads, frogs, newts and other animals to cross the road safely. A camera was placed inside the tunnel to see which animals were using it. The camera had an automatic trigger to take shots only when something moved inside the tunnel. This project is based on those images.
Most of the images have no label: we do not know if an animal is present, what animal it is or where it is on the image. The main objective of this project is to implement a solution capable of answering theses questions automatically, learning from a few hand-labelled examples. We can split this objective into several smaller ones :

- Choose which animals we want to look for, and select the images accordingly: images are mostly false positives containing nothing but leaves, so we need to filter them. We chose to keep only those containing newts, toads and frogs.
- Label the images: as said earlier, most of the images have absolutely no labels, and those that do only have the information of whether it contains a toad or a newt. We need to manually label them, giving the animal type and a bounding box for each of them.
- Machine learning: we need to find and experiment a machine learning technique able to do what we want.
- Test: we need to test the machine learning model to see if it produces coherent results

2 Database description

As stated above, the data in this project are pictures of newts, frogs and toads taken by a camera. A part of this data has been labelled in a previous project but most of the pictures are not. There are hundreds of GB of unlabelled data that can't be used for training.. Furthermore, the label of a picture only state what kind of animal is in it. No indication of the location of it is added. The "boxes" showing exactly the location had to be manually added during the first part of the project using an external tool [6]. Manually labelling all the available data was out of question since it would require a tremendous amount of work so only 100 pictures were treated and used as training set. The data are available on a server at the HEIG-VD and can be accessed through *ssh*. Given their very large size, the data entirety of the data were not downloaded locally to work with; only the relevant picture for one camera were taken. The two following pictures show the data we are using.



(a) Example of a newt (b) Example of a frog

Figure 1: Dataset extract

3 Data pre-processing & feature extraction

Every picture from the initial dataset was first cropped to keep only the relevant part (the wooden plank). This part is the central piece of every picture and should allow the object recognition to have the best results since the contrast between the white of the plank and the black of the animals makes them easily recognisable. Outside of this plank, the animals are difficult to spot even for a human observer since they don't stand out on the dirt.

Once every image were cropped, they were then split into squares along the horizontal. This step was necessary to have squared images again since the previous crop transformed them into rectangles which should be less efficient when fed to the model. The trick used in this step to avoid cutting an animal between two images is to add a margin to each image: each square start in the middle of the previous one so any animal caught between two should appear complete in at least one.

The last step of the pre-processing was to adapt the bounding boxes so that they match the cropped images. In order to keep only relevant images, we then filtered the cropped images by the percentage of bounding box present on them: any image with less than 25% of the original box is dropped. This allows to easily filter very small boxes that would only contain a small fraction of a newt or a frog and thus be hard to learn from by the model.

4 ML techniques

4.1 Transfer learning

In this project, we used already existing models on our data to do object detection. The existing models were already trained using the *COCO 2017* pictures so less training data were required on our side. By fully utilising the transfer learning, we were able to adapt the model to our problem without requiring an extensive dataset of labelled images. Only 100 images were used to train and refine the model (50 newts and 50 frogs). Of course, using a higher amount of training data would greatly increase the performance of the model but the task of manually adding boxes to the images requires a huge amount of time.

Different models are available for this problem [5]. We tried a couple of models from this list to see their effectiveness and which one is better. We managed to make our pipeline work with three types of model: *SSD MobileNet*, *EfficientDet* and *SSD ResNet50*. The results for each of those will be discussed in the following section. Some experiments were done on the other models but we couldn't make them work because of version compatibility problems.

The pipeline we used to label images come from this example [1]. This is available at the same place as the models and is done by the same people. In this example, they are trying to label plastic ducks. With this notebook as a starting point, we did the necessary modifications to be able to recognise newts and frogs.

4.2 Object Detection

While image classification's models try to predict the object in an image, object detection's ones try to also show where the predicted object is in the given image [2]. This project focused more on the latter by trying to draw boxes around predicted animals. The models used also add a score to the predicted box showing the estimated accuracy of the prediction. There are several possible methods to do object detection like R-CNN, YOLO and SSD [3]. The main method we used in this project is SSD which tries to locate objects in an image in a single shot (unlike R-CNN for example that takes two: one to identify the interesting regions and another to investigate those regions). This algorithm uses an image classification network as the first part of the neural network and then other convolutional layers to add the localisation boxes. This layer uses a set of default boxes with different aspect ratios and apply them to the previously created feature maps. The best boxes are selected by calculating scores for each category [4].

5 Experiments and Results

The main problem that occurred during the evaluation of our result is the lack of metrics to formally see if a model is better than another. Since the data used were not all labelled and that, even when labelled, only the category was given, not the location of the animal, it was hard to really evaluate how a given model performs. Hence, the following results are solely based on human observation of what seems to work better.

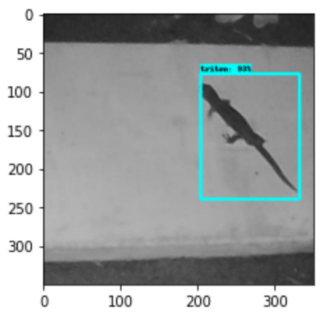
5.1 Models

Different models were tried during this project such as Faster R-CNN, SSD, EfficientDet and CenterNet. But only the following three gave usable results: *SSD MobileNet*, *EfficientDet* and *SSD ResNet50*. The others were not compatible with the pipeline we were using thus making them unusable in our case. From the three working ones, the results were diverging:

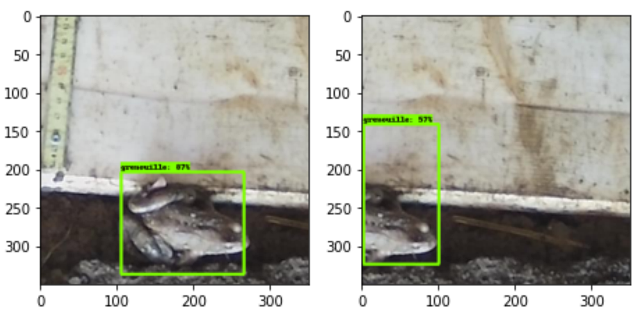
- The model *SSD ResNet50* seemed to be able to correctly locate both newts and frogs in a small number of cases, labelling the vast majority of the data as empty despite the presence of animals.

- The model *EfficientDet* seemed to have troubles with the training since the loss is not reducing with the iteration. During the first try with this model, some results were given where empty images received labels but with every other execution, not a single image was labelled by the model making us think that there may be a problem with the parameters.

- The model *SSD MobileNet* was giving good results for the localisation of newts but was worse for the frogs, often labelling a frog as a newt.

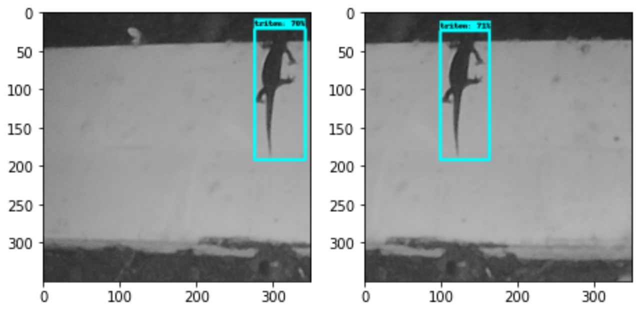


(a) Example of a newt

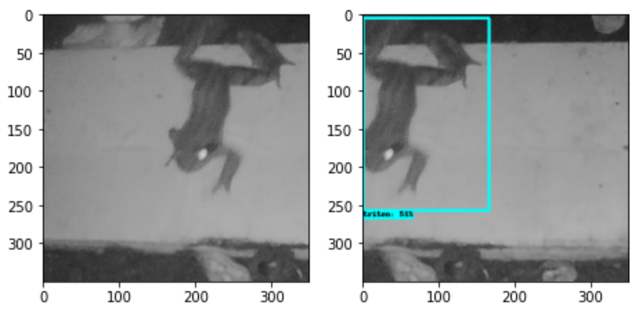


(b) Example of a frog

Figure 2: Resnet results



(a) Example of a newt

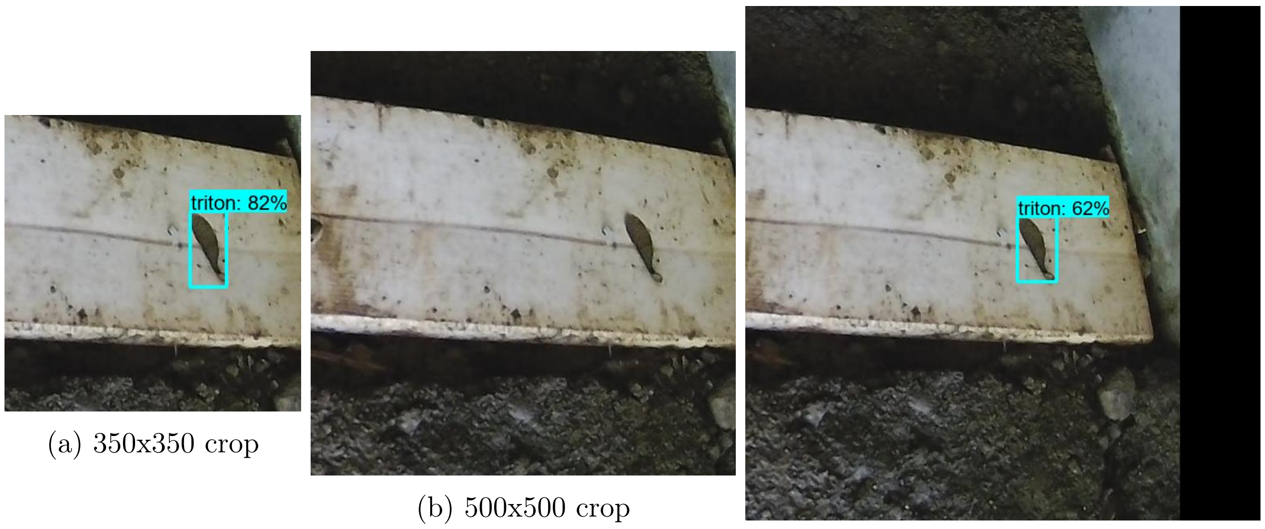


(b) Example of a frog

Figure 3: MobileNet results

5.2 Image size

As explained in the section 3, we had to crop images to keep only relevant parts. We thus experimented with different crop sizes. Our first idea was to use a 350 by 350 pixels crop. This first size was based on the size of the plank: each crop would have sufficient height to cover it vertically, as you can see on the previous figures (e.g. 1a).
This initial crop gave decent results, but there was an issue: the model used has inputs of size 640x640, so the crop has to be up-scaled, and we "waste" potential information by giving it a smaller image. For this reason, we then tried using crops of size 500x500 and 640x640 (full model size).
As we can see on the figure 4, the higher size helps with false positives, either completely removing them (4b) or at least reducing the confidence of the output significantly (4c).



(a) 350x350 crop

(b) 500x500 crop

(c) 640x640 crop

Figure 4: Crops sizes example

5.3 Data Augmentation

Our last experiment was to use data augmentation to compensate for the lack of training samples. We used techniques to rotate both the image and the bounding boxes of the training samples by angles of ±90°. Surprisingly, the results were not better, and some were even worse, though we saw some improvements for newts that were perpendicular to the plank, as there was no example of those in the training set.

6 Analysis and Conclusions

As stated in the previous section, we don't have a former protocol/metric to evaluate the performances of the different solutions. We relied on visual observation to see if a result "looked" better than another one. Because of this, the results we obtain during this project have little scientific value and can't be used to draw real conclusions on this matter. The main issue with creating our own metric was finding a way to evaluate the "correctness" of a box since there is no ground truth. Furthermore, most images do not have a label so even just evaluating if the animal was correctly identified would be a challenge. The first thing to improve would be to use labelled data for both the training and the testing which we didn't do because the amount of data available was small so reducing it further by saving a part for the testing would leave almost nothing for the training, most likely reducing the performance. The second step would be to find a metric of some sort to evaluate how far of the ground truth the predicted box is. A solution of some kind could be to compute the average difference between the two box for each edge but that may not be a relevant metric.

It is interesting to see that by using transfer learning, it is possible to use and train a model using a very small set of training data and achieve seemingly decent results which could without a doubt be improved by tweaking the model and selecting a more appropriate one. Creating our own model before training it on the data could yield even better result since it would be designed for the problem. This would require extensive knowledge in image recognition and detection to pull off which is not common.

In conclusion, we can say that our experiments showed that it is indeed possible to detect and appropriately label newts and frogs on our dataset. While there are still some challenges, like

handling false positives and checking the accuracy of the bounding boxes, we showed that we could apply a transfer learning technique to learn from only a limited dataset.
The next steps of this project would be to increase the size of the labelled dataset, which would greatly improve the detection performances. A simple way to do that would be to manually take the correctly labelled images output by the model and add those to the training set. We could then repeat this operation to obtain a better dataset every time.

References

[1] https://github.com/tensorflow/models/blob/master/research/object_detection/colab_tutorials/eager_few_shot_od_training_tf2_colab.ipynb

[2] https://developers.arcgis.com/python/guide/how-ssd-works/

[3] https://arxiv.org/pdf/1512.02325.pdf

[4] https://www.datacamp.com/community/tutorials/object-detection-guide

[5] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

[6] https://www.makesense.ai/