Crapauduc – MLBD

Joé Donzallaz, Dylan Mamié & Jérôme Vial

Ce projet a été réalisé dans le cadre du cours de Master « Machine Learning on Big Data » proposé par la HES-SO Master.

1 Contexte général et objectifs

Le canton de Vaud a construit plusieurs tunnels sous la route dans la commune de Gimel, pour permettre aux animaux de pouvoir traverser en tout sécurité. Dans le but de justifier les coûts engendrés par ce projet, des caméras ont été installées pour compter le nombre de passages d'animaux. Ces caméras sont programmées pour prendre une série de photos lors de la détection d'un mouvement, que ce soit de jour ou de nuit (figure 1).

À l'origine, **le comptage des tritons, grenouilles ou crapauds** sur les images capturées dans le tunnel était réalisé manuellement. Cette tâche prend du temps et il est facile de manquer un animal. L'objectif principal de ce projet vise donc à automatiser cette tâche à l'aide des images déjà labellisées et d'un modèle de machine learning.

Les tâches à réaliser peuvent être découpées de la manière suivante :

1. Sélection des images
2. Traitement des images
3. Choix des features
4. Choix d'un modèle pré-entraîné et de ses paramètres
5. Évaluation de la qualité des résultats

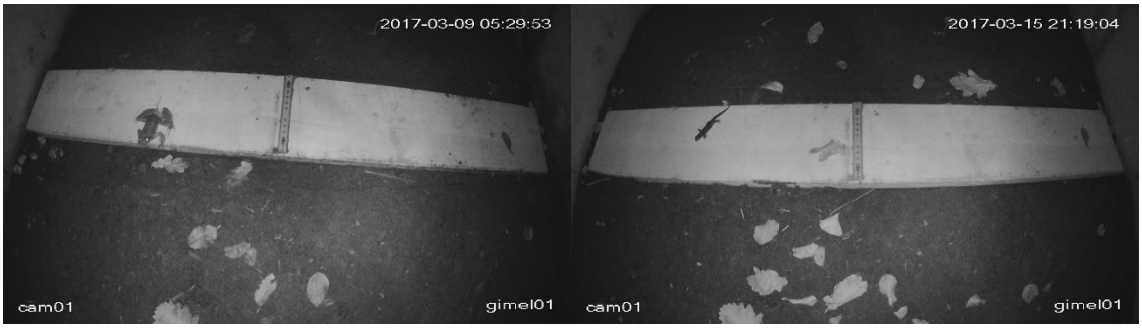

Figure 1 - Extrait de la base de données avec une grenouille et un triton

2 Description de la base de données

La base de données est composée d'un peu moins d'un million d'images capturées automatiquement dans différents tunnels pour animaux. Celle-ci se trouve sur un serveur de la HEIG-VD auquel nous avons un accès à distance.

Les images ont été capturées par un total de 18 caméras dont l'angle de vue est équivalent. La luminosité est fortement influencée par la météo et l'heure de la journée. Les caméras infrarouges permettent de capturer les déplacements nocturnes : ces images sont donc exclusivement en noir et blanc.

Ce grand nombre d'images ne nous est pas véritablement bénéfique puisque seules ~2000 images sont labellisées. Les informations suivantes sont disponibles :

- Nom de l'image
- Taille de l'image
- Type d'élément (grenouille, triton, insecte, souris, feuille ou plastique)
- Délimitations de l'élément dans l'image (bounding box)

Cependant, il existe de nombreux faux positifs causés par les déplacements des feuilles mortes ou par les visites des employés en charge de l'installation.

3 Pré-traitement des données et extraction des features

Nous avons utilisé les images des caméras 4 et 11 puisqu'elles offrent une belle diversité en termes d'éléments labellisés. Il n'empêche que le nombre de grenouilles et de tritons sont limités à ~150 chacun : c'est pourquoi nous avons mis en pratique le principe de « data augmentation » pour générer des variantes de nos images afin d'agrandir notre base de travail.

La libraire *imgaug* [1] permet d'apporter des modifications aux images tout en conservant la position des délimitations des éléments. En conséquence, 5 variantes ont été générées aléatoirement pour chaque image.

La transformation la plus importante consiste à recadrer les images au format carré, puisque les modèles utilisés dans les prochaines étapes ont tous été entraînés avec des images carrées. Comme le recadrage est réalisé aléatoirement, nous vérifions qu'au moins 80% de l'élément soit toujours présent après recadrage. Le cas échéant, la variante est regénérée depuis zéro.

Comme les conditions dans lesquelles sont capturées les images diffèrent selon le moment de la journée et la saison, nous appliquons finalement de légères modifications à la luminosité et à la netteté de l'image originale.

4 Techniques de Machine Learning

Cette section explique les deux principales techniques de Machine Learning utilisées lors de la résolution de ce problème : *Object detection* et *Transfer learning*. L'implémentation de celles-ci est basée sur une ressource officielle publiée par TensorFlow [2].

4.1 Object detection

Chaque image peut potentiellement présenter plusieurs scénarios : une grenouille, plusieurs grenouilles, une grenouille et un triton, rien du tout, etc. Il ne suffit donc pas de prédire un label par image, mais il faut aussi identifier où se situent les éléments au sein de l'image. En l'occurrence, chaque prédiction est accompagnée d'un score (entre 0 et 1) exprimant la certitude du modèle.

Il existe une multitude de méthodes différentes pour résoudre cette tâche : SSD, RetinaNet, YOLOv3, R-CNN, etc. Nous avons utilisé SSD – *single-shot detector* – dans le cadre de ce travail puisqu'il en existe plusieurs variantes avec des modèles de base différents. Cela nous permet de facilement passer d'un modèle à l'autre sans devoir apporter de grandes modifications au code.

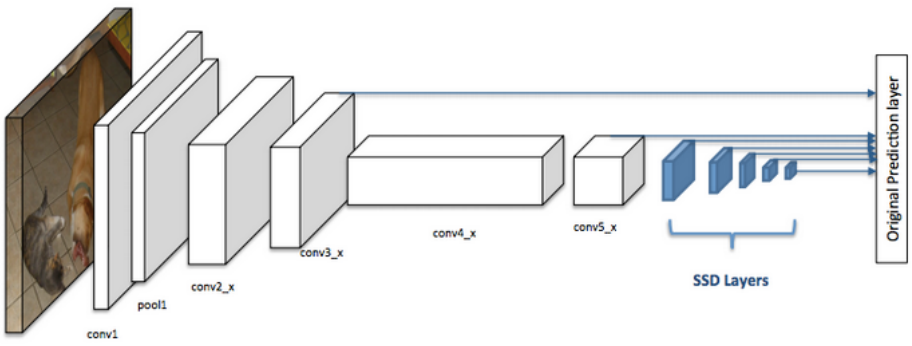

Figure 2 - Architecture d'un CNN avec des couches SSD [3]

SSD utilise un modèle pré-entraîné (dont la couche de classification finale a été supprimée) tel que ResNet ou ImageNet pour extraire les features des images. Il ajoute ensuite ses propres « convolutional layers » en tête du modèle pour obtenir les délimitations des objets accompagnées de leurs classes et scores en sortie (figure 2).

Les paramètres du modèle final sont donc dépendants du modèle pré-entraîné sélectionné.

4.2 Transfer learning

Nous avons utilisé des modèles pré-entraînés proposés par TensorFlow [4] pour n'avoir qu'à entraîner les couches de « haut-niveau » et ainsi gagner un temps conséquent. Ces modèles ont été pré-entraînés sur le dataset *COCO 2017* [5] qui est composé de plusieurs centaines de milliers d'images dont les objets ont été délimités manuellement. Ces images sont toujours au format carré, mais leur taille diffère selon le modèle choisi : 512x512, 640x640, etc.

Comme les grenouilles et tritons ne font pas partie des éléments labellisés, il est nécessaire d'apprendre à ces modèles à les reconnaître avant de passer à la phase de prédiction.

5 Expérimentations et résultats

Cette section démontre les différents paramètres utilisés dans le but d'améliorer la performance du modèle, que ce soit au niveau de la génération de variantes d'images, de la sélection du modèle ou des hyperparamètres de celui-ci.

La performance a été mesurée à l'aide de l'indicateur « Mean Average Precision » [6] qui consiste à diviser l'intersection de deux bounding boxes (prédiction & vérité terrain) par leur union. Il est donc nécessaire d'utiliser des images labellisées en tant que données de test : c'est pourquoi nous prenons un faible pourcentage de nos images pour jouer ce rôle, bien que la taille de nos données d'entraînement s'en retrouve ainsi réduite.

5.1 Data augmentation

Comme expliqué précédemment, nous avons appliqué de nombreuses (mais légères) modifications à chaque image pour en générer 5 variantes. La modification la plus importante consiste à recadrer les images au format carré pour être en harmonie avec les modèles pré-entraînés.

De manière générale, les résultats obtenus sont meilleurs en utilisant le principe de « data augmentation » puisque ce dernier permet d'étoffer notre dataset, mais aussi de simuler différents niveaux de luminosité (potentiellement causés par les déplacements du soleil).

En explorant les résultats, nous avons constaté que certains animaux n'étaient pas détectés puisqu'ils se déplaçaient dans une direction qui ne figure pas dans nos données d'entraînement. Pour pallier ce problème, nous avons tenté d'appliquer une rotation aléatoire à chaque image. Le nombre de vrais positifs a en conséquence augmenté, tout comme le nombre de faux positifs (figure 3). Il est donc difficile à déterminer si cette transformation est globalement bénéfique.

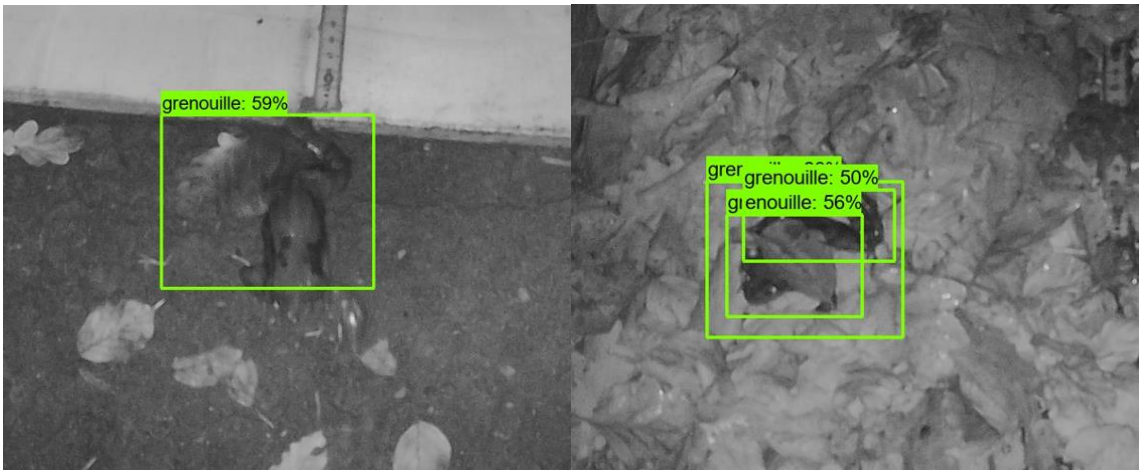

Figure 3 - Détections avec flou de mouvement puis avec des feuilles

Contrairement à nos attentes, le modèle était très confus lorsqu'il a été entraîné avec toutes les données labellisées : grenouilles, tritons, feuilles, souris, insectes et plastiques. Les résultats étaient plus convaincants en ne conservant que les grenouilles et tritons.

5.2 Modèles

Les modèles suivants ont tous été entraînés avec les mêmes paramètres (présents dans le notebook en annexe) avec des images de 640x640 pixels. À plusieurs reprises, nous avons modifié le nombre d'epochs, le learning rate, le momentum ainsi que l'optimizer (SGD & Adam) sans observer de changements conséquents dans la qualité des résultats.

Modèle	TF mAP [4]	mAP	Remarques
SSD ResNet50	0.343	0.25	Bonne détection dans l'ensemble. Le nombre de faux positifs est faible. Cependant, les grenouilles sont parfois labellisées comme étant des tritons.
SSD ResNet152	0.354	0.20	De nombreux tritons et grenouilles ne sont pas détectés. Le nombre de faux positifs est faible.
SSD MobileNet V1	0.291	0.11	La majorité des tritons et grenouilles ne sont pas détectés. Heureusement, le nombre de faux positifs est faible. Lors de la phase d'entraînement, la perte a très peu été réduite.

Bien entendu, les résultats sont légèrement variables d'une exécution à l'autre puisque la séparation des données d'entraînement et de test, tout comme la génération de variantes d'images sont des procédés aléatoires.

6 Analyse et conclusion

En fonction des paramètres utilisés, nos différents modèles ont été capables d'identifier une plus ou moins grande partie des grenouilles et tritons. Notre modèle est donc encore loin d'être parfait pour plusieurs raisons.

Premièrement, le nombre d'images de grenouilles ou tritons labellisées est relativement faible, alors que le nombre de facteurs influençant la détection est élevé : ces animaux peuvent être sur la planche, sur la terre ou dans les feuilles. Ils peuvent se déplacer dans n'importe quelle direction, mais aussi être en mouvement lors de la capture (ce qui a pour conséquence un flou de mouvement). Le principe de « data augmentation » nous a déjà permis de combler partiellement ce manque de diversité dans notre dataset.

Deuxièmement, nous avons eu de la difficulté à évaluer la qualité des résultats obtenus puisque les métriques habituelles ne s'appliquent pas directement à un cas de détection d'objets. C'est pourquoi de nouvelles métriques, telles que *mean Average Precision (mAP)* ou encore *Average Recall (AR)*, ont été développées [7]. Le seuil minimal de score joue aussi un rôle majeur et doit être ajusté dépendamment du modèle et des résultats obtenus. En l'occurrence, certains modèles étaient habituellement plus confiants que d'autres : ils nécessitent donc un seuil minimal de score plus élevé pour éviter un trop grand nombre de faux positifs.

Finalement, les modèles mis en place sont tous basés sur SSD. Il serait donc intéressant d'adapter le code pour expérimenter avec d'autres méthodes (YOLOv3, R-CNN) puis de comparer les résultats. Comme nous avons utilisé le principe de « transfer learning » pour réduire le temps d'entraînement, nous supposons qu'un modèle créé depuis zéro spécifiquement pour ce cas serait plus performant. Cependant, nous n'avons ni l'expérience ni la puissance de calcul nécessaire pour relever ce défi.

Ce travail démontre qu'il est possible d'utiliser le Machine Learning, avec les connaissances acquises lors des séances théoriques, pour remplacer un travail long et fastidieux s'il doit être réalisé à la main. Bien entendu, les possibilités offertes par le Machine Learning sont infinies et même si un modèle est excellent à un instant précis, il est tout à fait possible qu'il se fasse surpasser d'ici quelques années.

7 Références

[1] A. Jung, «imgaug,» 1 Juin 2020. [En ligne]. Available: https://imgaug.readthedocs.io/en/latest. [Accès le 12 Décembre 2021].

[2] TensorFlow, «Eager Few Shot Object Detection Colab,» 11 Juillet 2020. [En ligne]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/colab_tutorials/eager_few_shot_od_training_tf2_colab.ipynb. [Accès le 22 Décembre 2021].

[3] ArcGIS Developers, «Single-Shot Detector (SSD),» 2021. [En ligne]. Available: https://developers.arcgis.com/python/guide/how-ssd-works/. [Accès le 22 Décembre 2021].

[4] TensorFlow, «TensorFlow 2 Detection Model Zoo,» 7 Mai 2021. [En ligne]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md. [Accès le 22 Décembre 2021].

[5] COCO, «Common Objects in Context,» 25 Août 2021. [En ligne]. Available: https://cocodataset.org. [Accès le 22 Décembre 2021].

[6] A. Fawzy Gad, «Evaluating Object Detection Models Using Mean Average Precision (mAP),» 1 Novembre 2021. [En ligne]. Available: https://blog.paperspace.com/mean-average-precision/. [Accès le 3 Janvier 2022].

[7] R. Padilla, «Open-Source Visual Interface for Object Detection Metrics,» 2 Septembre 2021. [En ligne]. Available: https://github.com/rafaelpadilla/review_object_detection_metrics. [Accès le 29 Décembre 2021].

8 Table des figures