Compte-Rendu

Détection et comptage des poissons

M2. Master en Systèmes Embarqués et Systèmes Intégrés
Projet Master 2
Étudiant : JAVIER LÓPEZ José Antonio
N° Etudiant : 21804920
Date de livraison : 24 janvier 2020
Professeur : Johann Laurent
Encadrant du projet : Baptiste Verneau

MARPORT

Faculté sciences & sciences de l'ingénieur
ubs
Université Bretagne Sud

Sommaire

Liste de Figures

1 Introduction

Marport est une entreprise qui conçoit et fabrique des capteurs de contrôle, de capture et de surveillance des filets, des sondeurs, des profileurs de courant et des sonars pour la flotte de pêche mondiale. Afin d'innover et de créer de nouveaux produits, l'entreprise travaille en partenariat avec l'UBS.

L'un des projets de Marport est d'étudier une méthode de comptage et de détection des poissons basée sur des caméras embarquant de l'intelligence artificielle. Le projet prend appui sur deux axes de travail :

- Détection simple. Dans une image, il s'agira de déduire l'objet correspondant. Cette première approche permettra de comprendre le fonctionnement de TensorFlow ; et différentes documentations sont également disponibles pour son développement.
- Détection multiple. Dans une image, nous pouvons avoir plus d'un objet : les détections multiples trouveront les objets et préciseront leur emplacement dans l'espace. La documentation est très focalisée pour le développement sur d'autres plates-formes comme Android, ce qui signifie qu'il faut extrapoler l'information pour exporter le modèle sur la carte JeVois.

2 Développement

La détection des poissons peut être faite de différentes manières. Les plus connues sont le traitement d'image en utilisant Open CV et l'intelligence artificielle. Avec ces deux manières, il est possible de détecter et classifier l'image. Cependant, les principales différences sont les suivantes :

- TensorFlow est une Framework et Open CV une bibliothèque.
- Open CV est une bibliothèque spécialisée dans le traitement d'images en temps réel.
- TensorFlow est un Framework pour travailler avec les tenseurs et la différenciation automatique.

TensorFlow est une bonne option puisqu'il est théoriquement simple à utiliser. De plus, beaucoup de développeurs l'utilisent et sa licence est Apache License 2.0. Cette communauté de développeurs est donc un avantage parce qu'il est possible de trouver des solutions aux problèmes qui se posent pendant le développement du projet. De plus, Marport a suggéré d'utiliser l'*Artificial intelligence* (AI) pour effectuer le processus de détection. TensorFlow est donc le candidat idéal pour ce processus puisqu'il offre l'opportunité de créer des modèles à partir de zéro ou de réutiliser des modèles selon nos besoins. Cependant, il existe d'autres outils pour faire les modèles à partir de zéro comme Pytorche et Keras.

Le développement de l'inférence simple ou multiple est différent. Cependant, il y a quatre concepts importants à retenir :

- Un modèle est une fonction dont les paramètres peuvent être modifiés afin de diminuer l'erreur entre la valeur obtenue et la valeur souhaitée. Sur TensorFlow, il y a des modèles préconstruits, ce qui facilite le développement d'un projet.
- Jeu de données. Ce sont des images qui seront utilisées pour l'entraînement et l'évaluation du modèle. Il est nécessaire d'avoir une grande quantité d'images pour obtenir un bon modèle.
- Entraînement et test. Pendant le process d'entraînement, certains paramètres du modèle sont auto-ajustés afin de réduire l'erreur. De plus, les données d'évaluation permettront de décider si le modèle fonctionne correctement.
- Graphiques de flux de données. TensorFlow utilise des graphiques de flux de données pour représenter des calculs. Dans un graphe de flux de données, les nœuds représentent les unités de calcul tandis que les arêtes représentent les données consommées ou produites par une unité de calcul.

Le process résumé est le suivant :

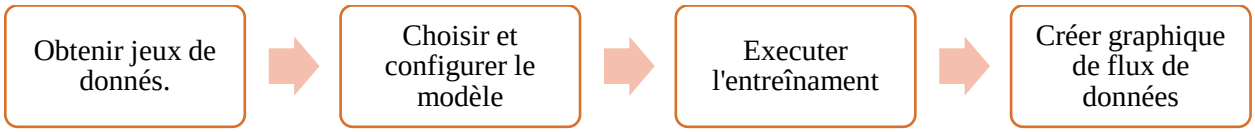

Figure 1. Process basique pour la création d'un modèle

Pour développer le projet, deux autres éléments ont été sélectionnés :

- Systèmes d'exploitation (SE). TensorFlow peut être utilisé sur différents SE tels que Windows, Linux et iOS. Le SE choisi a été Linux puisque l'installation des dépendances et des programmes est plus facile ; il y a une grande communauté qui utilise TensorFlow dans Linux et la carte Raspberry l'utilise pour son fonctionnement.
- Gestion de versions. GitHub a été utilisé pour avoir une meilleure traçabilité du projet mais aussi parce que c'est une façon simple de travailler sur le même projet à partir de différentes machines. Sur Github se trouve une documentation plus détaillée sur le projet allant de l'installation des programmes à la création et à la modification des fichiers entre autres. Le dépôt a 3 branches : la branche *master* qui contient un modèle entraîné 250 000 fois avec des images de basse résolution et aussi avec une documentation plus complète et détaillée ; la branche *ImagesHautQualite* qui contient un modèle entraîné 60 000 fois avec des images de haute résolution, -les résultats n'ont pas été à la hauteur des attentes- ; la branche *gcolab* qui est similaire à *Master* mais peut être exécuté sur une plateforme virtuelle.

  La documentation détaillée du projet se trouve sur les liens suivants (il faut copier et coller, l'accès direct ne fonctionne pas) :
  - https://github.com/josantoniojl/FishLite
  - https://github.com/josantoniojl/FishLite/tree/master/Inference/tensorflow-for-poets-2
  - https://github.com/josantoniojl/FishLite/tree/master/DetectionMultiple
  - https://github.com/josantoniojl/FishLite/tree/gcolab

L'objectif de ce document est d'expliquer de manière générale le développement du projet avec un accent particulier sur les résultats obtenus.

2.1 Détection simple

2.1.1 Adaptation d'un modèle

L'entreprise qui fabrique la carte JeVois a commencé chez KickStarter. En 2017, les premières versions de la carte ont été vendues. La carte peut capturer, faire le traitement et l'affichage d'image Dans les faites, la carte JeVois peut fonctionner comme un ordinateur autonome parce qu'elle peut communiquer via le port série.

L'entreprise propose un tutoriel pour effectuer une détection simple en utilisant TensorFlow. Cette méthode est similaire aux quatre concepts mentionnés ci-dessus. Pour adapter le modèle, il est important de récolter plus de 50 images par classe. Comme le modèle est préconfiguré, une seule commande doit être exécutée pour lancer l'entraînement. Lorsque l'entraînement est terminé, le diagramme de flux de données sera aussi créé.

Pour exécuter le modèle sur la carte JeVois, il est nécessaire d'utiliser TensorFlow Lite. Cette version permet d'exécuter les modèles TensorFlow sur les appareils mobiles et embarquées. Il est composé de deux éléments principaux :

- L'interpréteur exécute des modèles spécialement optimisés sur de nombreux types de matériel comme la carte JeVois ou les dispositifs Android.

- Le convertisseur transforme les modèles TensorFlow en une forme efficace qui peut être utilisée par l'interpréteur. Le convertisseur peut aussi introduire des optimisations pour améliorer la taille et les performances binaires.

Comme l'interpréteur est déjà installé sur la carte JeVois, il faut juste convertir le modèle obtenu en une version Lite et le placer sur la carte Je vois. Les résultats sont assez encourageants car il est assez facile de détecter les poissons. C'est aussi une façon de comprendre comment fonctionne TensorFlow et comment il est structuré. Cependant, cette procédure est très limitée car le *threshold* (seuil) ne peut pas être modifié, la surface de détection est petite et un seul élément peut être détecté par image. De plus, pendant l'exécution des tests, il y a eu plusieurs faux positifs. Pour cette raison, il n'est pas recommandé d'utiliser cette option. Les résultats obtenus sont les suivants :

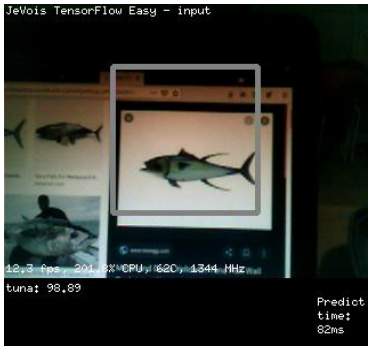

Figure 2. Détection du poisson thon

Figure 3. Détection du poisson thon2

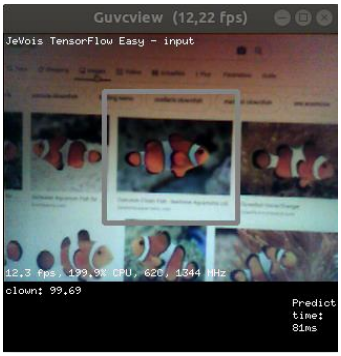

Figure 4. Détection du poisson clown

2.1.2 Réutilisation d'un modèle

Le modèle est déjà placé sur la carte JeVois. Malgré son utilisation facile, il n'est pas possible de le personnaliser. Le modèle est similaire au précédent mais il dessine un carré autour de l'élément trouvé. Le modèle peut apprendre de façon dynamique. Il faut donc prendre une photo et indiquer son nom. A partir de ce moment, le modèle essaiera de trouver des similitudes en temps réel. S'il trouve des similitudes avec la photo prise précédemment, ce sera une détection positive.

Le modèle est limité à une photo par élément, ce qui provoque une détection très faible. Il a également été remarqué que plus on place de photos, plus la reconnaissance est lente. Bien que ce soit un moyen simple de détecter des éléments, il n'est pas recommandé car il est lent et non paramétrable. Cependant, c'est une bonne façon d'aborder la carte JeVois et de comprendre son fonctionnement. En effet, la carte JeVois n'a pas d'interface graphique comme une Raspberry ; pour accéder à la carte, il faut effectuer une communication série. Cette procédure permet donc de connaître quelques commandes de base pour utiliser la carte. En utilisant cette méthode, les résultats sont les suivants :

Figure 5. Détection de Pomacentrus

Figure 6. Détection d'Amphiprion

Figure 7. Détection d'Amphiprion 2

2.2 *Détection multiple*

2.2.1 Développement

TensorFlow offre aussi la possibilité d'avoir une détection de plusieurs objets dans une image avec des cases délimitées. Théoriquement, il est possible de reconnaître 80 classes d'objets différentes. La détection multiple est un processus plus long. Le processus global peut être résumé dans le diagramme suivant :

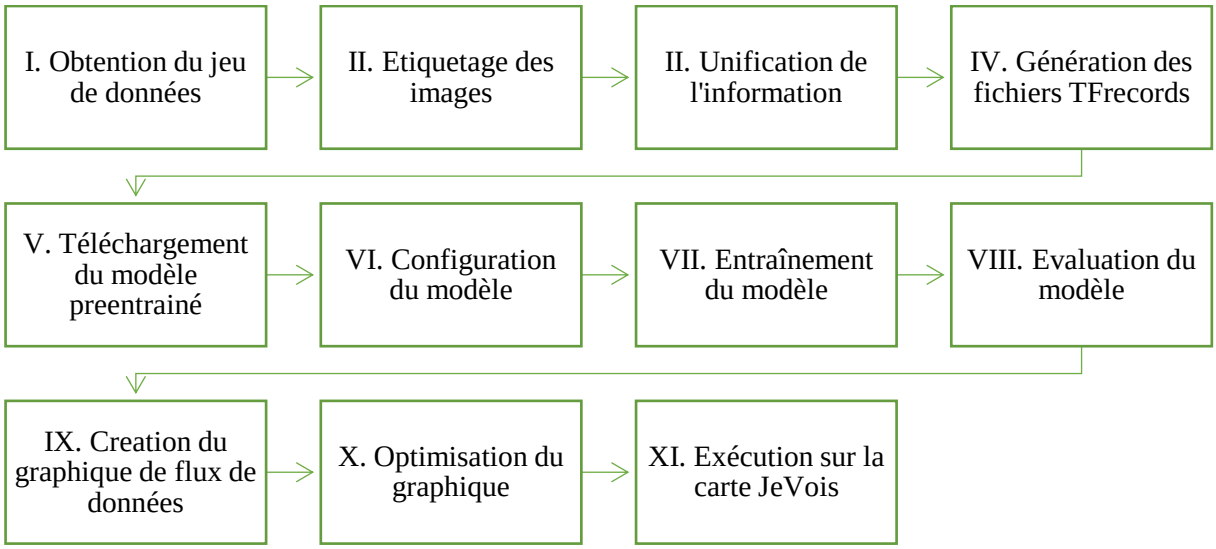

Figure 8. Flux de travail pour la détection multiple

2.2.1.1 Obtention du jeu de données

Il faut sélectionner les images qui seront utilisées pour l'entraînement et la validation du modèle. La façon la plus simple d'obtenir les images est de partir de l'environnement dans lequel le modèle va travailler. Cependant, pour le développement du projet, les images ont été téléchargées sur le site Fish4Knowledge (licence MIT).

Pour le développement de ce projet, seulement 6 classes ont été utilisées. Chaque classe représente alors les espèces de poissons suivantes :

Figure 9. Amphiprion

Figure 10. Dascyllus

Figure 11. Myripistis

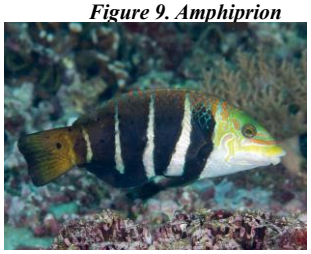

Figure 12. Hemigymnus

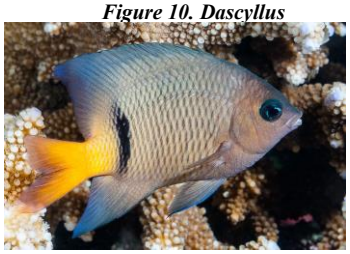

Figure 13. Plectroglyphidodon

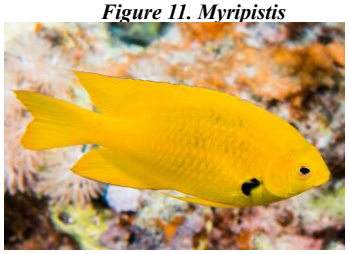

Figure 14. Pomacentrus

Pour le projet, chaque classe a plus de 100 images. TensorFlow recommande d'avoir au moins 1000 images par classe (Google, imagenet2012). Cependant, il y a plusieurs tutoriels où seulement 50 images par classe sont utilisées. Il n'y a donc pas de règle pour définir la quantité d'images nécessaires par classe. Cependant, certains

facteurs doivent être pris en compte :

- Photos à 360° des poissons.
- Photos avec différentes sources de lumière.
- Des photos avec des arrière-plans différents.
- Photos cadrées de façons différentes.

2.2.1.2 Etiquetage des images

Une image peut contenir différentes classes et aussi différentes informations sans importance. Le processus d'étiquetage permet de délimiter les informations intéressantes. Le processus d'étiquetage définit la classe et sa position. Le processus peut être effectué en utilisant le programme ImgLabel. Le processus est long car toutes les images doivent être étiquetées manuellement. Au moment d'étiqueter une image, un fichier xml est créé ; il contient les données essentielles telles que les coordonnées de longitude et de latitude de chaque classe. Sans un bon étiquetage, l'apprentissage du modèle peut être affecté. L'identification des poissons peut donc être inexacte.

2.2.1.3 Unification de l'information

L'étiquetage des images génère un fichier par image. Pour faciliter le traitement des informations, tous les xml sont unifiés dans un seul CSV. Le fichier aura donc les colonnes suivantes : nom du fichier, largeur, hauteur, classe, xmin, ymin, xmax, et ymax. Deux fichiers CSV seront créés : un pour les images d'entraînement et le deuxième pour les images d'évaluation.

2.2.1.4 Génération des fichiers TFrecords

Un modèle peut travailler avec de grands ensembles de données. L'utilisation d'un format de fichier binaire pour le stockage des données peut avoir un impact significatif sur la performance et par conséquent sur le temps d'entraînement du modèle. Les données binaires prennent moins d'espace sur le disque et le temps d'accès pour obtenir l'information souhaitée est plus rapide.

Il est possible d'écrire le programme à partir de zéro. Les informations de base peuvent être trouvées sur ce site : *TensorFlow TFrecords*. Cependant, le programme développé par Datitran (Tran, 2018) avec une License MIT a été utilisé. Ce code est utilisé parce que c'est un programme stable que plusieurs tutoriels recommandent. De cette façon, il est possible de gagner du temps dans le développement du projet.

2.2.1.5 Téléchargement du modèle pré-entraîné

Un modèle pré-entraîné est un modèle qui a été entraîné sur un grand ensemble de données de référence pour résoudre un problème similaire à celui que nous souhaitons résoudre.

Il est proposé de choisir un modèle basé sur COCO puisqu'il a été fait pour la segmentation d'objets, la reconnaissance dans le contexte et il peut supporter 330k images avec 80 classes. Le modèle par excellence est MobileNet puisqu'il convient mieux aux applications de vision mobile et embarquées où la puissance de calcul est insuffisante. Ce modèle utilise des convolutions séparables en profondeur, ce qui réduit considérablement le nombre de paramètres par rapport au réseau avec des convolutions normales de même profondeur dans les réseaux (S, 2018). Cette opération est représentée dans la figure suivante :

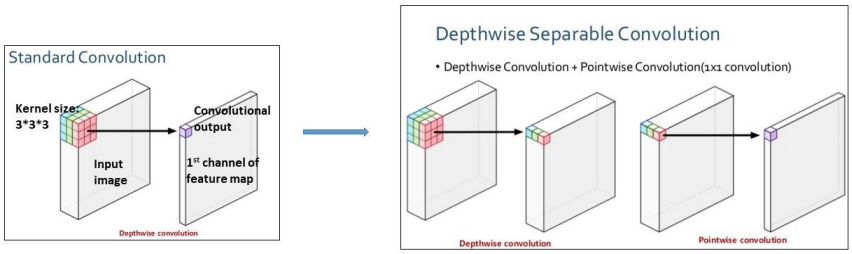

Figure 15. Différence de convolution (Liu, 2018).

Il existe d'autres modèles tels que Inception, Resnet, MnasNet entre autres. Bien que leur précision soit meilleure, ils ne peuvent pas être utilisés dans des systèmes aux ressources limitées. C'est pourquoi il est recommandé d'utiliser les différentes versions de MobileNet. Les performances des différents modèles peuvent être comparées sur la figure suivante :

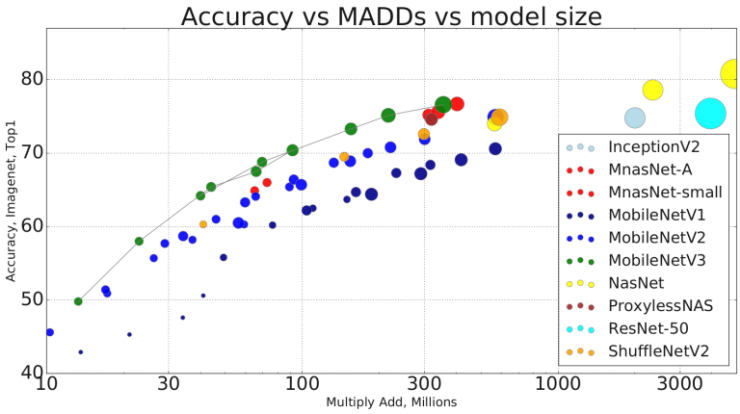

Figure 16. Comparaison entre modèles (Sezer, 2019).

Le modèle choisi a été la version V1 car c'est un modèle qui a été testé par plusieurs développeurs. Cependant, pour obtenir une meilleure performance, il est proposé d'utiliser la version MobileNet V3. Il existe également des modèles quantifiés qui permettent de réduire la taille du modèle tout en améliorant la latence du processeur et de l'accélérateur matériel. En revanche, il y a une dégradation dans la précision du modèle.

2.2.1.6 Configuration du modèle

Il y a des aspects importants à propos d'un modèle :

- Le modèle est exécuté en deux phases. Tout d'abord, les objets d'une image sont détectés. Ils sont ensuite classés en fonction de la classe à laquelle ils appartiennent.
- Le modèle utilise un réseau neuronal convolutif. Le réseau est un algorithme d'apprentissage profond qui peut prendre une image d'entrée, attribuer de l'importance poids et biais à divers objets de l'image et être capable de les différencier les uns des autres (Saha, 2018).
- Le modèle fait des prédictions dont la valeur est comprise entre 0 et 1. La valeur du *threshold* permettrait de définir quand un élément a été trouvé. Si le *threshold* est trop bas, il y aura de nombreux faux positifs. Un *threshold* neutre est de 0,5 mais cette valeur peut être optimisée.
- Le taux d'apprentissage (learning rate) est un hyperparamètre qui contrôle à quel point il faut changer le modèle en réponse à l'erreur estimée à chaque fois que les poids du modèle sont mis à jour. Le choix du taux d'apprentissage est difficile car une valeur trop faible peut entraîner un long processus d'entraînement qui pourrait se bloquer, tandis qu'une valeur trop élevée peut entraîner l'apprentissage trop rapide d'un ensemble de poids sous-optimaux ou un processus d'entraînement instable (Brownlee, 2019).
- Les fonctions de perte (loss) sont utilisées pour déterminer l'erreur (aussi appelée "la perte") entre la sortie de nos algorithmes et la valeur cible donnée. En termes simples, la fonction de perte exprime la distance

qui nous sépare de la valeur de sortie calculée (AI, 2018). Le modèle utilise deux façons différentes pour calculer l'erreur : Smooth L1 (Reddit, 2019) et Sigmoid-weighted (Elfwing, 2018).
- Le *batch* définit le nombre d'images qui seront propagées dans le réseau. À la fin du *batch*, les prédictions sont comparées aux variables de sortie attendues et une erreur est calculée. À partir de cette erreur, l'algorithme de mise à jour est utilisé pour améliorer le modèle en se déplaçant, par exemple, vers le bas le long du gradient d'erreur.

Certains paramètres qui peuvent être configurés dans le modèle sont les suivants :
- Définir le fichier TFrecords pour l'entraînement et l'évaluation (input data).
- Définir les classes qui seront détectées.
- Définir la taille du *batch*.
- Définir le nombre de steps.

Chaque modèle a des paramètres différents qui doivent être modifiés. C'est pourquoi il est important que le modèle choisi dispose d'une bonne documentation ou d'une bonne communauté qui a travaillé avec le modèle au cas où il y aurait des doutes sur sa configuration.

2.2.1.7 Entraînement du modèle

L'entraînement permet au modèle d'améliorer ses prédictions. Le nombre d'étapes (steps) est un hyperparamètre qui définit le nombre de fois où l'algorithme d'apprentissage va travailler sur l'ensemble des données d'apprentissage. Une étape signifie que chaque échantillon de l'ensemble de données d'apprentissage a eu l'occasion de mettre à jour les paramètres internes du modèle.

Cela veut dire qu'au début de la phase d'apprentissage, il y a une grande marge d'erreur et au fur et à mesure que les étapes d'apprentissage sont exécutées, le modèle ajuste ses paramètres pour diminuer l'erreur (information d'entrée contre information de sortie).

Il n'y a pas de règle pour définir le nombre d'étapes nécessaire pour entraîner le modèle. Pourtant, le plus simple est d'utiliser l'outil Tensorboard qui permet du surveiller le processus d'apprentissage. Deux courbes sont à surveiller : pertes (losses) contre nombre d'étapes et précision contre nombre d'étapes. Lorsque l'un des graphiques a tendance à être constant, il est conseillé d'arrêter le processus d'apprentissage. Dans le cas du modèle mobilnet, l'entraînement est considéré comme étant bon quand le graphique de *losses* est inférieur à 2.

Cependant, il y a deux critères qui doivent être pris en compte :
- *Underfitting* : Quand le modèle peut être encore amélioré. Cela peut se produire pour un certain nombre de raisons : le modèle n'est pas assez puissant, le modèle est trop régularisé ou l'entraînement n'a pas été suffisant.
- *Overfitting* : Si le modèle est entraîné trop longtemps, il ne prendra pas en compte les cas généraux et ne fonctionnera qu'avec des images répondant à certaines caractéristiques. Pour éviter le surentraînement, la meilleure solution est d'utiliser des données d'entraînement plus complètes.

2.2.1.8 Evaluation du modèle

L'étape d'évaluation permet de vérifier si le modèle fonctionne correctement. Ce processus peut être répétitif puisque, selon les résultats obtenus lors de l'évaluation, l'étape d'apprentissage peut être exécutée de nouveau en tenant compte des problèmes d'*Underfitting* et d'*Overfitting*.

2.2.1.9 Création du graphique de flux de données

Lorsque l'entraînement est exécuté, plusieurs fichiers sont créés qui représentent ensemble le modèle. Afin d'intégrer l'ensemble du modèle dans un seul fichier, un processus appelé *freezing* est effectué. Ce processus intègre également tous les éléments tels que le poids, les graphiques et autres dans un seul fichier. Le fichier créé est en lecture seule, donc le fichier ne peut pas être modifié ou utilisé comme point de départ d'un autre entraînement. Le processus de *freezing* doit être compatible avec TensorFlow lite puisque le fichier créé peut être optimisé par la suite en fonction de l'architecture souhaitée.

2.2.1.10 Optimisation du graphique

Le convertisseur TensorFlow Lite prend un *freezing model* et génère un FlatBuffer. Le FlatBuffers est une bibliothèque de sérialisation multi-plateforme efficace pour C++, C#, C, Go, Java, JavaScript, Lobster, Lua, TypeScript, PHP, Python et Rust.

Il existe différentes configurations lors de la conversion du modèle en version Lite. La configuration dépendra du type de modèle choisi. Aussi, il est conseillé de consulter la documentation du site TFLite Converter, afin d'effectuer un export efficace. Pour le développement du projet, une simple exportation a été faite. Cependant, il y a quelques options qui peuvent améliorer la performance du modèle exporté.

2.2.1.11 Exécution sur la carte JeVois

En suivant les étapes de 2.1 Détection simple, il est possible de faire fonctionner le modèle sur la carte JeVois. Une autre étape sera de lancer un script Pyhton afin d'obtenir la position des poissons et de procéder au comptage des poissons.

2.2.2 Résultats

2.2.2.1 Installation du TensorFlow.

TensorFlow a débuté le 9 novembre 2015. À ce jour, plus de 20 versions ont été publiées. Le plus grand défi présenté dans le développement du projet était l'installation correcte de TensorFlow. Trois méthodes différentes ont été utilisées pour installer TensorFlow :

- Ordinateur personnel : TensorFlow fonctionne avec des instructions spécifiques du processeur. A partir de la version 1.6, TensorFlow utilise les Advanced Vector Extensions (AVX). AVX se concentre sur l'amélioration de l'efficacité des calculs vectoriels (principalement les calculs en virgule flottante). Les jeux d'instructions peuvent être utilisés pour : le traitement de l'image, le traitement vidéo, le traitement audio, la modélisation 3D, les services d'analyse financière et les logiciels d'ingénierie et de fabrication.
  Par conséquent, si l'ordinateur ne dispose pas de ce jeu d'instructions, TensorFlow ne peut pas être utilisé. La seule version compatible est TensorFlow 1.5, mais elle est considérée comme une version obsolète parce que la plupart des scripts essentiels pour l'entraînement et l'exportation du modèle ont besoin de cet ensemble d'instructions. Cependant, il est possible de travailler avec TensorFlow 1.5. Le gros inconvénient est qu'il existe actuellement plusieurs commandes qui ne sont pas compatibles avec la version 1.15. Aussi, la grande difficulté pour trouver sa documentation fait que cette version n'est pas recommandée pour le développement du projet.
- Raspberry Pi 4. Afin de tester le potentiel des cartes de développement pour l'intelligence artificiel, il a été décidé de travailler avec une carte de développement Raspberry Pi 4. La version de TensorFlow semi-stable pour la Raspberry est 1.13.1. Cependant, les fichiers d'installation proposés par Google ne sont pas configurés correctement. Pour exporter le modèle dans sa version lite, il faut utiliser l'outil Nigthly qui n'est pas activé dans les versions TensorFlow pour Raspberry. Les façons d'obtenir cette configuration sont les suivantes : compiler TensorFlow depuis la Raspberry Pi ; réaliser une compilation croisée ; trouver les codes binaires à partir d'un autre programmateur. Les deux premières solutions prennent beaucoup de

temps. La dernière solution fonctionne temporairement, mais ce n'est pas une solution à long terme.
La partie d'entraînement provoque que la Raspberry occupe tous ses processeurs. La carte commence à chauffer rapidement, jusqu'à ce qu'elle atteigne plus de 75°C. L'augmentation de la température provoque un phénomène appelé *throttling*. Afin d'assurer le bon fonctionnement de la carte, un boîtier a été construit ; il protège en même temps la carte et il comprend un espace pour placer un ventilateur. De plus, des dissipateurs de chaleur ont été placés sur la carte, ce qui a permis d'obtenir une température inférieure à 50° C.

- Google Colab. Avec tous les problèmes rencontrés, un service en ligne capable de gérer TensorFlow a été recherché. Google Colab est un outil de recherche pour la formation et la recherche associées au machine learning. Il est aussi un environnement de notebook Jupyter qui ne nécessite aucune configuration et qui s'exécute entièrement dans le cloud. Il peut être utilisé pour l'exécution des scripts Python et pour développer des projets en utilisant Keras, TensorFlow, Pytorch et OpenCV. La caractéristique la plus importante qui distingue Colab des autres services on-Cloud est la gratuité d'utiliser un GPU sur les projets. De plus, Google Colab peut être liés à google drive ce qui permet d'accéder aux fichiers nécessaires pour exécuter l'entraînement.
  Le service google présente de nombreux avantages. Toutefois, l'environnement sur Google Colab a une durée d'inactivité de 90 minutes et une durée absolue de 12 heures. Cela signifie que si l'utilisateur n'interagit pas avec l'environnement pendant plus de 90 minutes, son instance est automatiquement terminée. De plus, la durée de vie maximale d'une instance Colab est de 12 heures (Gupta, 2020). Alors que la puissance de traitement est rapide, le développement du projet peut être ennuyeux car la connexion à google drive, la déclaration des variables et l'installation des dépendances doivent être exécutées à chaque fois que le programme est ouvert, lorsque l'instance est terminée ou lorsqu'il y a une perte de connexion internet.

Les autres solutions pouvant être envisagées sont :

- Ordinateur plus performant : TensorFlow travaille sur toutes les architectures 32 et 64 bits. Cependant, lorsque l'entraînement est effectué, il commence à consommer beaucoup de ressources. Selon le livre "*TensorFlow 1.x Deep Learning Cookbook*", il est recommandé de disposer d'un ordinateur avec les caractéristiques suivantes : CPU à 64 bits (avec des instructions AVX), entre 8 à 32 Go de RAM, entre 4-8 cœurs et optionnellement un GPU NVIDIA. De plus, il faut considérer un espace d'au moins 5Go pour le stockage du projet. Le principal avantage est de disposer de TensorFlow dans un seul ordinateur et de pouvoir l'utiliser à tout moment. Aussi, il sera plus facile d'effectuer différents types de tests afin de trouver la configuration la plus appropriée pour la détection des poissons. En revanche, le principal inconvénient est le prix de l'ordinateur.
- Gcloud : Au lieu d'envisager un superordinateur, il est possible d'utiliser un simple ordinateur simple qui peut faire fonctionner TensorFlow et d'utiliser des services de paiement. Google offre la possibilité d'utiliser ses ordinateurs pour entraîner le modèle en au moins 30 minutes. Ce service est payant et le prix dépendra des caractéristiques de l'ordinateur ainsi que de la durée de sa location. Google offre également un service d'étiquetage. L'étiquetage peut se faire à partir d'images ou d'une vidéo, ce qui permet de gagner du temps dans l'étiquetage des images et donc dans le développement du projet. Pour en savoir plus sur le sujet, les sites suivants sont disponibles :
  - Tarifs de Compute Engine.
  - Tarifs Data Labeling Service.
  - Présentation d'AI Platform.
  - Premiers pas : entraînement et prédiction avec TensorFlow Estimator.

  Cela peut être une excellente option si un gain de temps dans le développement du projet est souhaité car

les étapes qui prennent plus de temps seraient sous-traitées.

Une grande partie du temps passé sur le projet a été consacrée à essayer de corriger les bugs trouvés pendant le projet et installer une version fonctionnelle de TensorFlow. Cela signifie qu'avoir la bonne plateforme permet de diminuer les erreurs provenant d'une mauvaise installation de TensorFlow.

2.2.2.2 Comparaison résultats d'entraînement Raspberry vs Google Colab

Sur Google Colab, l'entraînement par étape est plus rapide (0,45s contre 2,8s). Cela signifie que l'entraînement par étape est 6,2 fois plus rapide. De plus, comme un *batch* plus grand peut être utilisé, le nombre d'étapes pour entraîner le modèle diminue (8 000 contre 250 000). L'impact est tel qu'au lieu d'effectuer un entraînement de 8,1 jours, seule une heure est nécessaire.

La fiabilité du modèle est aussi différente. L'entraînement exécuté sur le Raspberry Pi est très variable parce que le processus de détection est aléatoire comme le montre l'image ci-dessous :

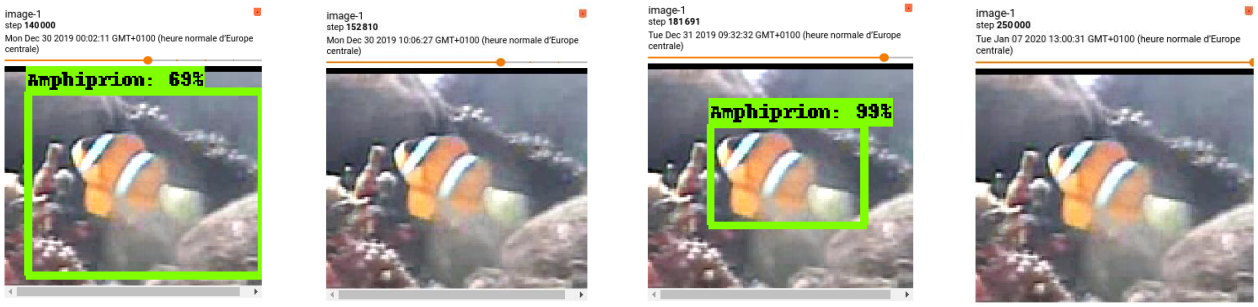

Figure 17. Evolution de la détection du poisson Amphiprion, version Raspberry Pi

Au lieu d'affiner la détection des poissons, il y avait des certains moments pendant lesquels les poissons n'étaient pas détectés. Cela se produit parce les paramètres du modèle sont ajustés après l'analyse d'une seule image (*Stochastic Batch*). Dans le graphique des *loss*, il est possible d'observer comment l'erreur ne se stabilise jamais. Une tentative a été faite pour stabiliser le modèle en l'entraînant 250 000 fois mais la stabilisation n'a jamais eu lieu. Cela signifie qu'un Batch égal à 1 n'est pas recommandé. De plus, l'entraînement aurait pu être arrêté au 150 000, ce qui aurait permis de gagner trois jours de travail. Les images suivantes montrent l'évaluation de l'erreur :

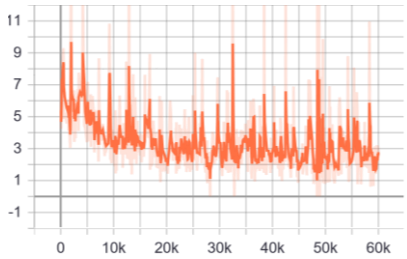

Figure 18. Evolution du loss (0-60k)

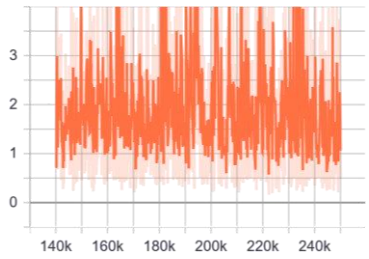

Figure 19. Evolution du loss (140k-250k)

En utilisant Google Colab il y a la possibilité d'utiliser un *batch* plus grand ce qui permet de généraliser le modèle. Cela explique que, lors de la phase d'évaluation, la plupart des poissons aient été correctement identifiés.

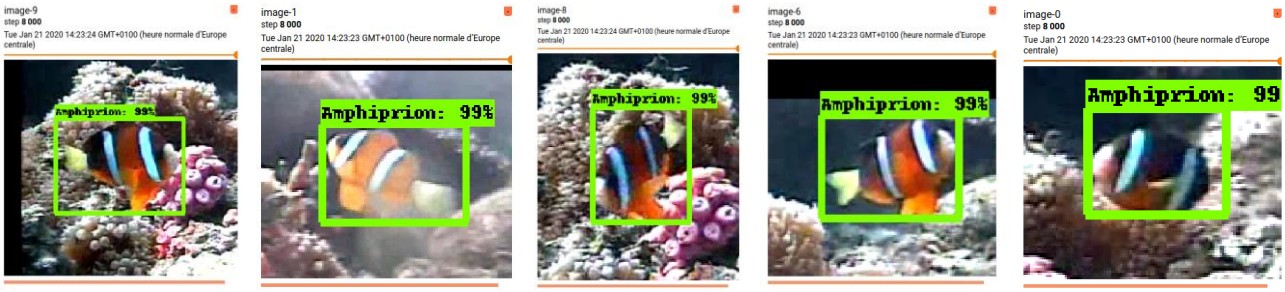

Figure 20. Détection du poisson Amphiprion, version Google Colab

Il existe deux graphiques de *losses* puisque le modèle identifie d'abord l'objet puis le classifie. En utilisant un batch plus grand, l'erreur atteint un point stable, au lieu de toujours changer. Les images suivantes montrent l'évaluation de l'erreur avec un batch égal à 32 :

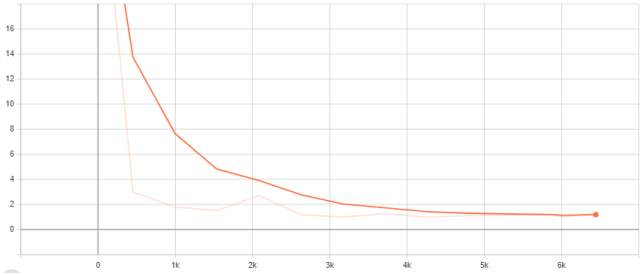

Figure 21. Evolution du loss de classification (0-6500)

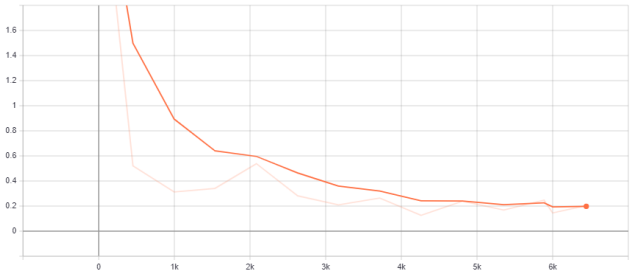

Figure 22. Evolution du loss de localisation (0-6500)

Google Colab offre de bons résultats, tant en ce qui concerne la rapidité de traitement de l'information que la fiabilité du modèle.

2.2.2.3 Détection de la position

Après l'analyse d'une image, le modèle génère quatre sorties :

- Coordonnées de la boîte de délimitation des objets détectés.
- Indice de classe des objets détectés.
- Confiance dans les objets détectés.
- Nombre total d'objets détectés.

```
Boxes detected :
 [[ 0.25632688  0.31366354  0.7111433   0.754855  ]
 [ 0.11891437  0.45066372  0.8118657   0.72292674]
 [ 0.7644105   0.13223247  0.9834882   0.8083858 ]
 [ 0.33924904  0.27875412  0.85836583  0.53569555]
 [-0.02056137  0.11706067  0.26720336  0.6415852 ]
 [ 0.6978015   0.16883518  0.9671905   0.58673596]
 [ 0.5231212   0.2748067   1.1350187   0.4936361 ]
 [ 0.36240664  0.45428085  0.92060274  0.7543533 ]
 [ 0.00308307  0.35159472  0.276118    0.7687625 ]
 [-0.03930693  0.3451545   0.3882567   0.80303776]]

Classes detected :
 [0. 0. 3. 0. 0. 3. 2. 4. 4. 0.]

Scores detected :
 [0.8861187  0.07779583 0.05508175 0.05002752 0.04137269 0.03681442
 0.03571099 0.03382558 0.02549118 0.02357498]

Total objects detected :
 10.0
```

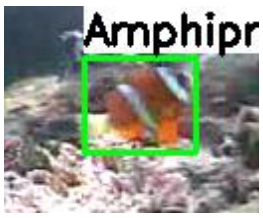

Figure 23. Sortie du modèle

La valeur la plus importante pour filtrer l'information est la confiance puisqu'il s'agit de la probabilité que l'élément soit celui souhaité. La valeur de confiance sera comparée à la valeur du *Threshold* : si la valeur est supérieure, cela signifie qu'il y a correspondance. L'étape suivante consiste à localiser la position du poisson et à dessiner un carré enveloppant sur celui-ci. Ce processus est exécuté en fonction du nombre de poissons trouvés dans l'image. Cependant, le modèle a été limité à seulement 10 objets par image pour un traitement plus rapide. Un schéma simple pour détecter les poissons est le suivant :

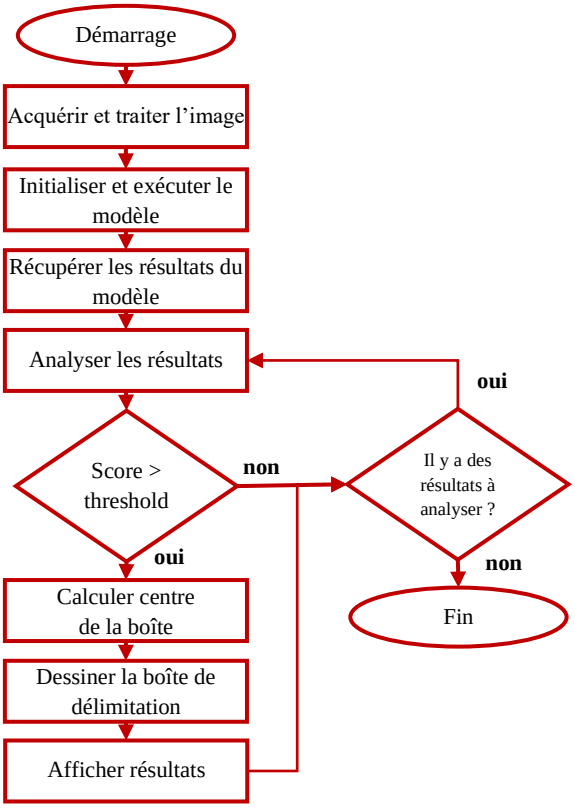

Figure 24. Détection et localisation des poissons

Les résultats obtenus sont les suivants :

Figure 25. Détection simple du poisson (basse résolution)

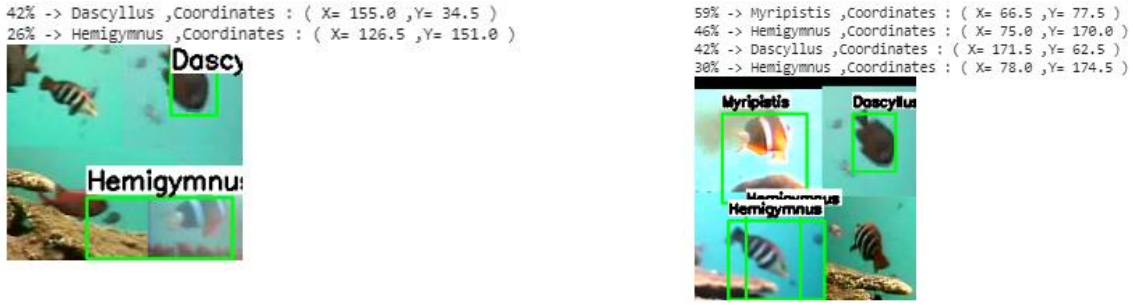

Figure 26. Détection multiple des poissons (basse résolution

Le programme fonctionne avec des images statiques. Cependant, quelques modifications simples doivent être apportées afin de pouvoir travailler avec une caméra. Par exemple, accéder à l'appareil photo pour prendre une photo et boucler le processus Prendre une photo -> analyser une photo. C'est pourquoi il est intéressant que le modèle fonctionne rapidement, de sorte qu'il donne la sensation de traiter les images en temps réel.

2.2.2.4 Utilisation du modèle sur la carte JeVois

Le modèle doit être sur le format "*.tflite*" sinon la carte JeVois ne pourrait pas l'exécuter. Ce processus s'exécute sur lle système sur lequel l'entraînement a été effectué. De plus, le convertisseur TFLite doit être utilisé car c'est celui qui permet de transformer le modèle dans son format optimisé. Ensuite, le modèle pourra être lu par l'interprète de la carte JeVois. Le processus théorique de fonctionnement est illustré dans la figure suivante :

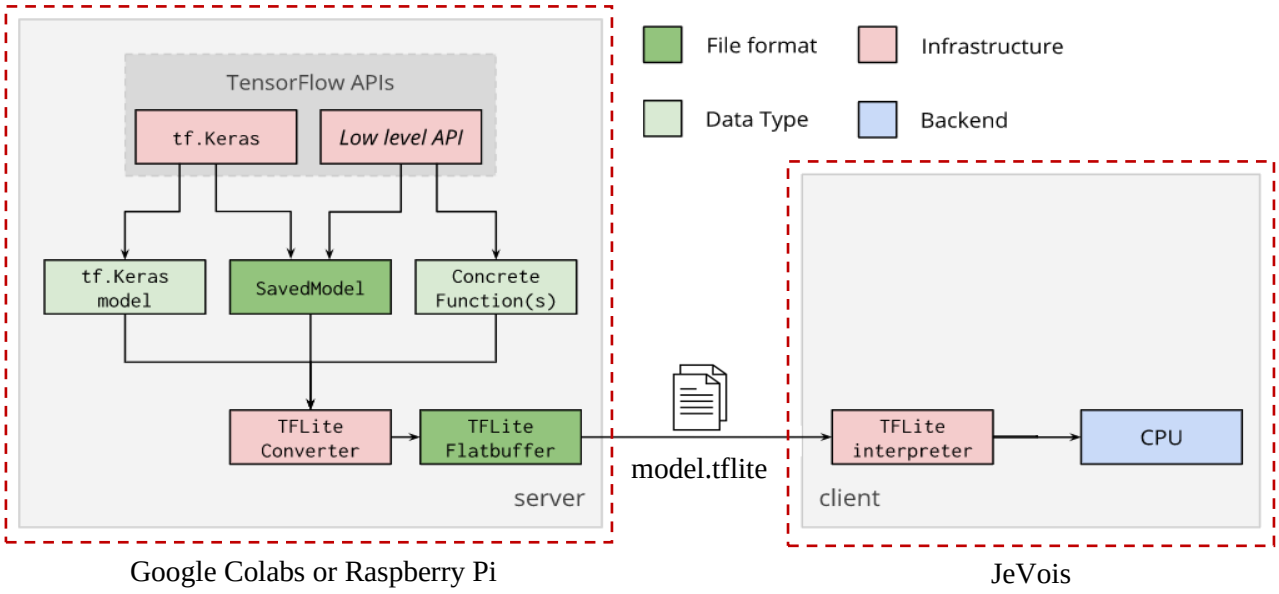

Figure 27. diagramme du processus de conversion (Google, TensorFlow , 2020)

La carte JeVois intègre un script pour faire fonctionner les modèles lite. Dans leur tutoriel, ils montrent comment le configurer et l'exécuter. Cela n'a cependant pas fonctionné correctement. Comme il existe plusieurs paramètres pour configurer la conversion du modèle, un programme a été créé pour vérifier son fonctionnement. Les résultats ont été les suivants :

Figure 28. Résultats du modèle TFLite

Cela signifie que le modèle lite fonctionne correctement mais qu'il ne peut pas être interprété par la carte JeVois. Le script développé par JeVois attend un certain type et une certaine quantité de résultats du modèle ; l'erreur se produit donc lorsque le modèle n'envoie pas les données attendues. Pour identifier l'erreur, il faut comprendre le fonctionnement des deux modèles. Le modèle simple ne détecte qu'un poisson par image. Sa sortie correspond à la probabilité de tous les poissons et leurs identifiants. En revanche, le modèle développé envoie pour chaque objet détecté les probabilités, les identifiants et les coordonnés de localisation des poissons. C'est le script qui trie l'information.

Cependant, au moment d'exécuter le modèle développé, le script reçoit plus de données que prévu. Pourtant, il reste bloqué dans une fonction d'erreur. Deux solutions possibles sont proposées :

- Modification du programme. En analysant le tutoriel, le programme qui tente d'exécuter le modèle est "*\JeVois\ TensorFlowEasy\ TensorFlowEasy.c*". Il est possible que qu'en apportant les modifications appropriées, le modèle créé puisse être exécuté.
- Exécution d'un script Python. Il existe un programme appelé *Inventor* qui permet de développer des modules de vision artificielle. Il est donc possible de prendre les programmes de test développés dans Google Colab et de les adapter à la structure de *Inventor*. Pour compter les poissons, il existe différentes façons de le faire. Une solution possible est de détecter les poissons sur l'image et d'afficher par espèce le nombre de poissons trouvés. Cependant, le problème est qu'il ne prendrait en compte que les poissons actuels et pas à un historique. Pour établir cette fonction, il faut diviser l'image en zones, la zone neutre et la zone de comptage. Si le poisson traverse de la zone neutre à la zone de comptage, alors il est compté, sinon il n'est pas pris en compte.

Cette dernière étape a été celle qui a arrêté le projet, de sorte que la détection multiple sur la carte JeVois n'a pas pu être réalisée.

3 Conclusion

3.1 Du projet

3.1.1 Concernant les images

- La plupart des images sélectionnées n'ont qu'un seul poisson, ce qui affecte modèle (réseau neuronal), car il y a toujours un poisson avec une probabilité élevée et les autres ont une probabilité très faible (voir **Erreur ! Source du renvoi introuvable.**). Une proposition consisterait à enrichir la base de données d'entraînement avec des photos contenant plusieurs poissons.
- Il est conseillé d'avoir suffisamment de photos, en tenant compte de plusieurs arrière-plans. Par exemple, Pomacentrus se trouve la plupart du temps à proximité d'un corail. Le réseau neuronal a appris que s'il y a un corail à proximité, il y a de fortes chances que le poisson soit un Pomacentrus. C'est pourquoi, à plusieurs reprises, il obtient ce faux positif.
- Amphiprion, Heminygus et Dascyllus ont des couleurs différentes. Cependant, elles coïncident en ayant des lignes blanches. Le modèle est donc souvent confondu avec ces espèces de poissons. Il semble également que le modèle tienne compte des couleurs (voir Figure 28. Résultats du modèle TFLite).

3.1.2 Concernant TensorFlow

- Le *batch* a eu un grand impact sur la formation du modèle. Il a permis de diminuer le nombre de fois à entraîner le modèle et aussi d'obtenir un modèle stable.
- L'utilisation d'un *Threshold* est importante car elle permet d'éliminer les éventuels faux positifs. Cependant, des espèces correctes ont parfois été trouvées mais avec une valeur faible de confiance. Il est possible que ce problème puisse être résolu en élargissant la base de données.
- Les versions de MobileNets V2 et V3 ont des performances de vitesse de traitement et une fiabilité supérieure à celles de la version V1 (Google, Github, 2019). Il est recommandé de réaliser les tests avec ces versions.
- La possibilité de créer un modèle exclusif pour la détection de poissons doit être évaluée. Par exemple, il existe des modèles spécifiques pour la détection des visages, des animaux, des personnes et des objets. Il est donc possible d'envisager le développement d'un modèle spécifique plutôt que de prendre un modèle général et de l'adapter.
- La version la plus récente de TensorFlow est la 2.0 mais Google Colab a installé la version 1.15. Cela signifie qu'ils préfèrent utiliser une version plus stable et aussi parce que de nombreuses personnes

travaillent sur cette version.

- Pour profiter réellement de toutes les fonctionnalités qu'offre TensorFlow, il est nécessaire d'avoir des connaissances en intelligence artificielle et en apprentissage machine. Par exemple, le fichier de configuration du modèle peut être optimisé car il existe plusieurs paramètres qui permettent d'améliorer son fonctionnement.

3.1.3 Concernant les cartes de développement

- La Raspberry est une carte polyvalente. Toutefois, pour configurer et entraîner un modèle, il n'est pas recommandé d'utiliser cette carte. Cependant, cela pourrait être une bonne option par rapport à la carte JeVois car il est possible d'ajouter une extension de caméra ou simplement de connecter une webcam. L'environnement de développement sur la Raspberry est plus simple et l'exécution des programmes aussi. De même, il y a un grand nombre d'utilisateurs qui utilisent la carte cela signifie qu'il y aura plus de documentation et une communauté active. Enfin, il est recommandé que la Rasberry ait accès à Internet via ethernet. Le module WiFi lit et le partage internet à partir d'un ordinateur portable limite le débit internet. C'est pourquoi la mise à jour de Github peut être un processus long.
- La carte JeVois est un bon outil. À première vue, la documentation qu'ils proposent est assez compliquée. Cependant, lorsque vous commencez à créer votre propre matériel pour la carte, il y a peu de documentation à ce sujet. La communauté des développeurs est minime par rapport à celle de la Raspberry. Sur les deux questions posées dans le forum, une seule a reçu une réponse après trois jours.
- Si un environnement de travail simple est souhaité, l'utilisation de Raspberry Pi est recommandée. Sinon, la carte JeVois peut répondre aux besoins du projet.

3.1.4 Général

- Pour le développement du projet, une attention particulière a été accordée au type de licences. La plupart des licences qui occupent les images, les scripts et TensorFlow, sont de type MIT et Licence Apache 2.0. Ces licences sont gratuites, mais il est nécessaire de faire référence et de donner crédit à leurs créateurs. Par conséquent, si le projet continue à se développer, il n'aura pas de problème majeur pour être commercialisé.
- Il est déconseillé de créer un dépôt Git sur le Google Drive. Lorsqu'un changement de branche est effectué, Google Drive détecte que les fichiers ont été supprimés et les télécharge à nouveau. Cela provoque une perte de temps car il faut attendre que les documents se synchronisent.

3.2 Personnelles

L'intelligence artificielle est un outil assez puissant. Toutefois, certains paramètres sont obtenus en fonction de l'application, de l'expérience et du test. Des recommandations peuvent être trouvées mais cela rend la configuration du modèle non optimale. Néanmoins, en développant ce projet, j'ai pu acquérir les connaissances de base de l'intelligence artificielle, de l'utilisation du cadre TensorFlow et de la programmation en Pyhton.

Même si je connaissais le chemin à suivre pour atteindre l'objectif, le développement du projet était lent. A chaque avancée sur le projet, je me suis confronté à des problèmes et des défis. Cela m'a donné l'impression que parfois je stagnais. Aussi, il y a eu un grand écart entre le planning initialement présenté. Cependant, le projet a été réalisé à 80%, puisque le modèle est fonctionnel, il suffit de le modifier pour qu'il puisse être exécuté sur la carte JeVois et fonctionner dans un environnement réel.

De plus, si j'avais trouvé Google Colab, j'aurais pu développer le projet plus rapidement. J'ai perdu beaucoup de temps à essayer de faire fonctionner TensorFlow sur la Raspberry Pi et aussi sur mon ordinateur personnel. Aussi, le grand inconvénient d'utiliser des modèles déjà entraînés réside dans le fait que le process n'est pas

transparent. Cela signifie que la recherche d'erreurs n'est pas évidente et la recherche de solutions prend beaucoup de temps.

Finalement, la documentation est un processus qui prend beaucoup de temps dans le développement du projet. Cependant, la documentation développée dans le cadre de ce projet permettra à la société Marport d'aborder le projet de manière simple et de ne pas commettre les mêmes erreurs que moi.

4 Références

AI, D. (2018). *Deep AI.* Récupéré sur Loss Function: https://deepai.org/machine-learning-glossary-and-terms/loss-function

Brownlee, J. (2019, 01 25). *Machine Learning Mastery.* Récupéré sur Understand the Impact of Learning Rate on Neural Network Performance: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Elfwing, S. (2018, 11). *Science Direct.* Récupéré sur Neural Networks: https://www.sciencedirect.com/science/article/pii/S0893608017302976

Google. (2019, 11). *Github.* Récupéré sur MobilNets: https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet

Google. (2020, 01 07). *TensorFlow.* Récupéré sur Overfit and underfit: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

Google. (2020, 01 07). *TensorFlow* . Récupéré sur TensorFlow Lite converter: https://www.tensorflow.org/lite/convert?hl=it

Google. (s.d.). *imagenet2012.* Récupéré sur TensorFlow: https://www.tensorflow.org/datasets/catalog/imagenet2012

Google. (s.d.). *TensorFlow.* Récupéré sur TensorFlow Lite: https://www.tensorflow.org/lite/guide

Gupta, P. (2020, 01). *Clouderizer Help Center.* Récupéré sur Google Colab FAQs: https://help.clouderizer.com/en/articles/2240606-google-colab-faqs

Liu, C. (2018, 12 13). *Medium.* Récupéré sur Feature Extractor 1 — MobileNet V1 & V2: https://medium.com/@CecileLiu/feature-extractor-1-mobilenet-v1-v2-90011eeb559b

Reddit. (2019). *Reddit.* Récupéré sur Smooth L1 Loss in Faster R-CNN: https://www.reddit.com/r/learnmachinelearning/comments/bqqwjj/smooth_l1_loss_in_faster_rcnn/

S, A. R. (2018, 04 18). *Quora.* Récupéré sur What is MobileNet: https://www.quora.com/What-is-MobileNet

Saha, S. (2018, 12 15). *Medium.* Récupéré sur A Comprehensive Guide to Convolutional Neural Networks: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sezer, O. B. (2019). *Github.* Récupéré sur Cnn pytorch: https://github.com/topics/cnn-pytorch

Tran, D. (2018, 08). *Github.* Récupéré sur Gnerate tfrecord: https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py