Institut National de la Recherche Scientifique

Centre Eau Terre Environnement

Québec

Filière : Science Informatique

Détection, Identification, Classification et Journalisation Automatique des Poissons dans le Passage des Poissons avec le Machine Learning

Rapport de Stage

Présenté par

Charles BEAULIEU

Sous la Direction des Professeurs

Normand BERGERON (INRS)

Erwan GLOAGUEN (INRS)

Usef FAGHIHI (UQTR)

16 août 2022

Résumé

Ce travail est rédigé dans le cadre de mon stage de l'été 2022 au sein de l'INRS (Institut National de la Recherche Scientifique). Le projet vise à développer une application d'apprentissage profond permettant la détection, l'identification, la classification et la journalisation automatisé des poissons sur les images vidéo. Cette méthode a déjà été utilisée avec succès dans d'autres contextes (ex. fréquentation de récifs de corail) pour l'identification des poissons, mais la méthode doit-être adaptée au contexte du fleuve St-Laurent par un entrainement spécifique du modèle à identifier les espèces locales. À terme, ce système constituera une très grande avancée pour toutes études futures de suivi migratoire de poissons à l'aide d'image vidéo.

Abstract

This work is written as part of my summer 2022 internship at the INRS (National Institute for Scientific Research). The project aims to develop a deep learning application allowing the detection, identification, classification and automated logging of fish on video. This method has already been used successfully in other contexts (eg. frequentation of coral reefs) for the identification of fish, but the method must be adapted to the context of the Saint-Laurent River by specific training of the model with local species. Ultimately, this system will be a major step forward for all future studies of fish migration monitoring using video image.

Table des matières

1.Introduction

Il existe déjà différentes techniques pour collecter des données sur le comportement des poissons telles que l'usage des transpondeurs RFID et la télémétrie acoustique, mais ces techniques ne sont pas parfaites. Ces techniques peuvent être fastidieuses (pêche pour pose de transpondeur, opération chirurgicale, etc.), coûteuses, parfois illégales (pour certaines espèces), relativement fiables et intrusives pour les poissons. L'usage de caméra pour collecter des images des poissons pour ensuite les identifier est une bonne façon de collecter des données sans être intrusif pour les poissons, mais l'évaluation des vidéos par des gens qualifiés est une opération coûteuse et fastidieuse. C'est pourquoi l'utilisation d'apprentissage profond (Deep Learning) pour détecter les individus sur les vidéos, puis les classifier et les journaliser semble être une solution très intéressante pour l'avenir.

Mon projet consiste donc à développer une application d'apprentissage profond pour détecter, classifier et compter les poissons présents dans les images obtenues dans des corridors de migrations mis en place dans la jetée servant à la démolition de l'ancien pont Champlain à Montréal. Le but est d'étudier les effets de la déconstruction du pont sur le comportement des différentes espèces de poissons présentes dans le fleuve Saint-Laurent.

Ce rapport est divisé en 3 parties. Premièrement une partie visant à faire l'analyse de la littérature afin d'analyser les travaux semblables ayant été réalisés auparavant. Deuxièmement, une partie sur les données (collecte, format, tri, etc.). Troisièmement, une partie sur la mise en place d'un modèle d'apprentissage profond permettant la détection, la classification et le tracking des individus du fleuve Saint-Laurent.

2.Littérature

Dans cette section, il sera question de présenter quelques écrits et ou projets pertinents pour l'avancer du travail. Le premier ouvrage présenté est: Automated Detection, Classification and Counting of Fish in Fish Passages With Deep Learning. Ce travail est très similaire au miens. La plus grande différence est que les image sont des images de sonar acoustique, c'est donc un type d'image qui est différent, mais les technologies utilisées sont des technologies que je peux utiliser dans le cadre de mon projet aussi. Le deuxième travail que je vais présenter s'intitule: Video Image Enhancement and Machine Learning Pipeline for Underwater Animal Detection and Classification at Cabled Observatories. Ce travail est intéressant, car il y a beaucoup de ressemblance avec mon projet.

2.1 Automated Detection, Classification and Counting of Fish in Fish Passages With Deep Learning.

Cet article a été rédigé à la faculté des sciences informatiques de l'Université d'Dalhousie à Halifax au Canada par Vishnu Kandimalla, Matt Richard, Frank Smith, Jean Quirion, Luis Torgo and Chris Whidden. Les chercheurs qui ont travaillés sur ce projet avaient sensiblement le même but que moi. C'est-à-dire, d'automatiser la détection, faire la classification et compter les poissons en utilisant l'apprentissage profond.

Dans cette étude, on mentionne qu'il y a plusieurs contraintes pour l'identification par vidéos telles que la variation de luminosité, le camouflage des poissons, les arrières plans complexes, l'eau trouble, la résolution faible, la déformation des formes quand les poissons nagent et les variations subtiles entre les différentes espèces de poissons. (Jalal et al., 2020). Certains résultats positifs ont été obtenus grâce à une technique qui consiste à identifier l'heure et la date de la journée, ce qui permet d'évaluer l'image selon des critères différents selon l'heure et la saison (Terayama et al., 2019).

2.1.1 Données

Le jeu de données utilisé provient de DIDSON. Contrairement à moi, dans cette recherche, les chercheurs utilisent des image haute résolution de vidéo acoustique (High Resolution Visual Acoustic Video). Les images ont été capturés dans une rivière de Ocqueoc au Michigan aux États-Unis. Le jeu de données contient 100h de vidéo, 525 extraits annotés au niveau de la position, de l'images et des trajectoires par des expert de la pêche.

TABLE 1 | Number of videos and extracted images of each species in the Ocqueoc River DIDSON dataset.

Species	Number of videos	Number of extracted images
Carp	51	7,953
Lamprey	190	6,986
Largemouth bass (Lmbass)	1	160
Pike	2	350
Smallmouth-bass (Smbass)	65	11,690
Steelhead	6	582
Sucker	100	21,676
Walleye	109	11,973

Les modèles de détection d'objets et la classification demande des images annotées avec des boîtes ou des masques. Dans le cadre de mon projet, les images devront être annotés par des personnes ayant les connaissances pour identifier les espèces de poissons. Dans leur travail l'application utilisée pour annoter était LabelImg tool (Tzutalin, 2015). Puis, pour fonctionner avec les modèles, le format devra être un jeu d'images annotés (sets of labeled image). Une des choses à laquelle il faut faire attention est l'overfitting (le modèle mémorise le jeu de données à la place d'apprendre à compléter la tâche). Pour pallier ce problème, les chercheurs de cet article n'annotaient pas tous les Images. Cette méthode évite que deux images presque identiques se retrouve dans le même « trainning set ». Dans le cadre de leur projet, les images contenant des poissons étaient annotées, mais les image n'en contenant pas ne le sont pas, alors ils ont codé un script pour qu'elle le soit aussi. Après avoir identifié toutes les images désirées, on classe les images en dossiers selon les différentes identifications. Le script python utilise 80% des données pour l'entrainement et 20% pour les tests. De plus, les données ont été mélanger pour réduire le

biais. La répartition des espèces de poissons n'est pas bien balancée comme on peut le voir ci-dessous.

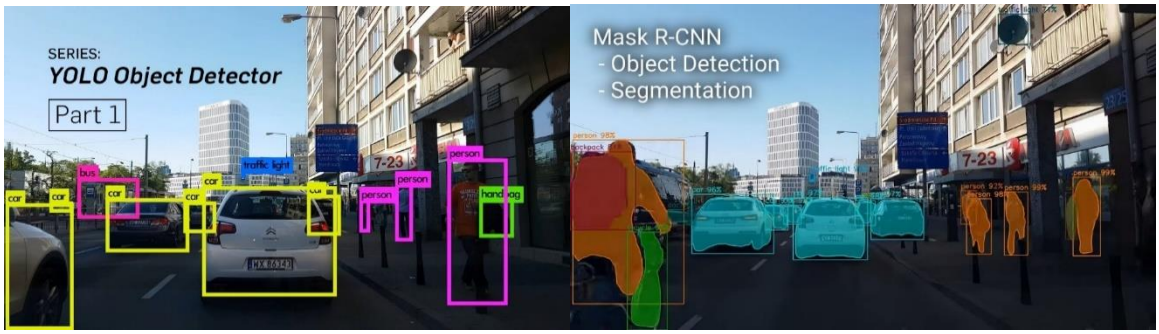
Species	Total number of images	Total Number of train images	Number of positive samples in train set	Number of test images	Number of positive samples in test set
Lamprey	2,741	2,193	36.6%	548	34.4%
Smbass	1,425	1,140	70.0%	285	72.9%
Carp	1,235	988	64.2%	247	64.8%
Sucker	1,155	924	86.4%	231	86.1%
Steelhead	582	465	44.3%	116	45.6%
Walleye	573	450	71.3%	114	72.8%
Pike	350	280	46.7%	70	50.0%
Lmbass	160	128	25.0%	32	21.8%

Le type de données collectés était des vidéos de 30 images par seconde avec une résolution de 1280 x 960. Un total de 24 000 images a été utilisées dans cette étude.

2.1.2 Modèles et Algorithmes

La famille de modèles de détection et classification YOLO (You Look Only Once) est une famille de modèles efficaces et optimisés pour une détection fiable et rapide en temps réel sur des appareilles embarqués. L'architecture du modèle utilisé (YOLOv3) est essentiellement un extracteur de caractéristiques. L'extracteur de caractéristiques est un réseau de neurones convolutif à 53 couches qui est nommé le Darknet-53. Les caractéristiques extraites sont fournies au réseau de détection qui va les utiliser pour détecter et identifier les entités.

La famille des modèle RCNN (region-based convolutional neural network) est une autre famille de modèle de détection qui dessine des masques plus précis autour des entité à la place de boîte comme la famille YOLO. Il existe plusieurs modèles comme Mask-RCNN et Faster-RCNN. Cette famille de modèles fonctionne en deux étapes « Backbone » et « Head » (Lin et al., 2017). La couche « Backbone » sert à proposer une zone qui devrait contenir des objets. Pour sa part, la couche « Head » sert à décerner l'entité, l'identifier et la classer.



YOLOv4 ajoute des composants à YOLOv3, ce qui permet d'augmenter la précision des boîtes. YOLOv4 comporte 3 principales étapes. Premièrement un extracteur de caractéristiques connue sous le nom de CSP (Cross-Stage-Partial-Connections) Darknet53 (Wang et al.,2020). Ensuite, un « Neck » qui relie le « Backbone » avec la « Head ». Le « Neck » est composé d'un SPP (spatial pyramid pooling) (He et al., 2015) et d'un PANet path-aggregation-network (Liu et al., 2018). Pour ce qui est de la « Head » elle est identique à celle utilisé pour YOLOv3.

Il est important de noter que lorsque les chercheurs ont commencé ce travail, l'implémentation de YOLOv4 n'était pas encore disponible.

Pour le suivie des poissons, les outils utilisés ont été la librairie Norfair combiné avec YOLOv4. (Bochkovskiy et al., 2020). Norfair est une librairie légère disponible sur python. Elle est destinée au suivie des objets en temps réel et peut être modifiée pour l'adapter au besoin. Pour faire fonctionner Norfair, on doit lui fournir une entré (input) sous la forme d'une détection, dans leur travail, YOLOv4 est utilisé comme détecteur.

Norfair fonctionne en établissant des prédictions sur la position de chaque futur point basé sur la position précédente et la vélocité estimé à l'aide d'un filtre de Kalman (estimation linéaire quadratique). L'algorithme essaie d'aligner la position précédente avec la nouvelle position que le détecteur donne en utilisant la distance euclidienne ou une autre fonction de distance.

2.1.3 Entrainement

Pour l'entrainement du premier modèle, dans leur cas YOLOv3, ils ont utilisé un modèle avec des poids de convolution pré-entrainé sur ImageNet (Russakovsky et al., 2015). Le modèle a été pré-entrainé sur un grand jeu de données avec plus de 80 classes d'objet.

Pour l'entrainement du deuxième modèle, c'est-à-dire Mask-RCNN, ils ont utilisé un modèle avec des poids de convolution pré-entrainé sur le jeu de données MS-COCO (Microsoft Common Objects in Context) (Lin et al., 2014).

Pour le troisième modèle, c'est-à-dire YOLOv4, il a été pré-entrainé sur le jeu de données ImageNet pour définir les poids de convolution. Ce modèle a été utilisé pour le suivie de poissons contrairement aux deux premier qui était pour la détection et la classification.

2.1.4 Résultats

Après avoir appliqué les deux modèles YOLOv3 et Mask-RCNN pour la détection et la classification de poissons, les résultats étaient encourageants, mais pas assez bon pour s'en contenter. Il existait un groupe d'images que le modèle n'avait pas réussis à identifier ou qu'il avait mal classé. Ils ont pris un sous jeu de données puis l'ont examiné, puis ont catégorisé les images en trois classes, les images clairement visibles, partiellement visible et les objets de type « Edge ». Ensuite, ils ont appliqué des techniques d'augmentation de données pour simuler un plus grand jeu de données pour mieux entrainer le modèle à gérer les poissons qui ne sont pas clairement visibles.

TABLE 3 | Results of the YOLOv3 model on each species in the Ocqueoc River DIDSON dataset with IOU thresholds of 0.4 and 0.5.

Species	TP	FP	AP	TP	FP	AP
	IOU = 0.5			IOU = 0.4		
Lamprey	30	37	0.16	44	23	0.36
Smbass	63	13	0.42	69	7	0.50
Carp	81	100	0.30	108	73	0.54
Sucker	73	40	0.36	81	32	0.47
Steelhead	12	12	0.24	9	15	0.28
Walleye	44	3	0.66	45	2	0.70
Pike	9	13	0.17	15	7	0.42
Lmbass	0	0	0.00	0	0	0.00

Species are sorted by the number of labeled images in descending order.

TABLE 4 | Results of the Mask-RCNN model on each species in the Ocqueoc River DIDSON dataset with IOU thresholds of 0.4 and 0.5.

Species	TP	FP	AP	TP	FP	AP
	IOU = 0.5			IOU = 0.4		
Lamprey	5	58	0.01	7	56	0.06
Smbass	106	100	0.24	127	79	0.38
Carp	114	116	0.38	142	88	0.57
Sucker	142	214	0.17	180	177	0.30
Steelhead	9	52	0.07	13	48	0.19
Walleye	55	26	0.24	60	21	0.35
Pike	15	30	0.31	18	27	0.47
Lmbass	1	7	0.00	2	4	0.25

Species are sorted by the number of labeled images in descending order.

En utilisant YOLOv3 et Mask-RCNN sur un échantillon de 1000 images avec un seuil de 0.5, les chercheurs ont obtenu 804 images où aucun objet n'était détecté. Ces images ont ensuite été traité avec un seuil de 01, 0.2, 0.3 et 0.4. Après le traitement, les images ont été classé en trois catégories, soit les objets clairement visibles **figure 4**, ceux dit « Edge » **figure 5** et ceux partiellement visibles **figure 6**.

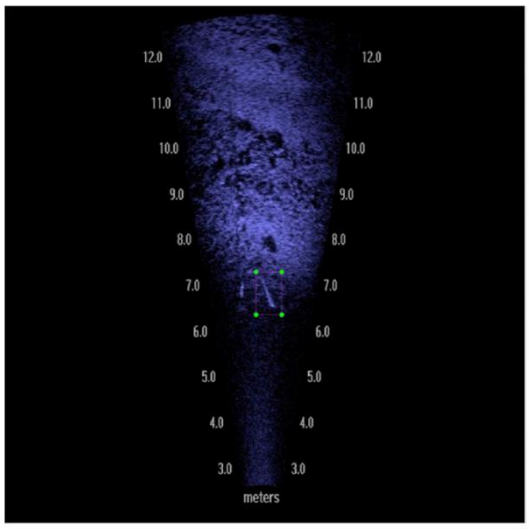


FIGURE 4 | Example image where the fish is clearly visible from DIDSON visual sonar.

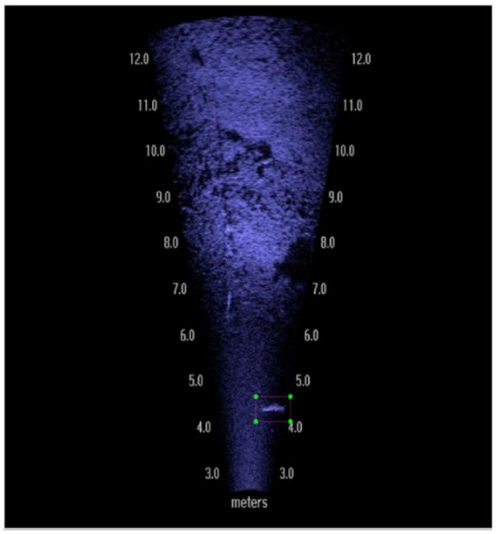


FIGURE 5 | Example image where the fish is at the edge of the DIDSON visual sonar image.

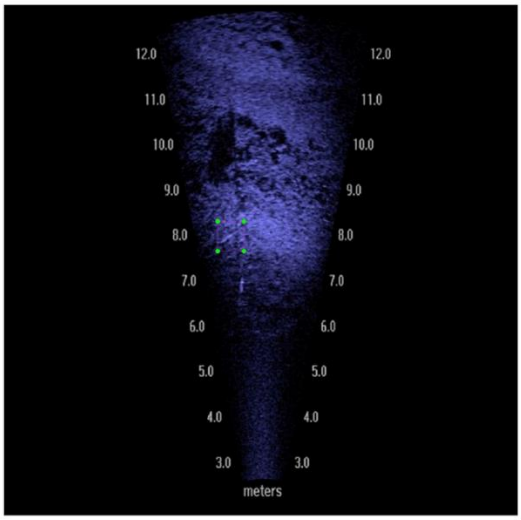


FIGURE 6 | Example image where the fish is partially visible.

2.1.5 Solution

Dans le but d'améliorer les résultats obtenus, les chercheurs on appliquer l'augmentation d'image qui consiste à modifier des caractéristiques de l'image telles que la saturation, exposition, le hue et appliquer des rotations aléatoires.

2.1.6 Résultats Après Solution

En résumé, les résultats ont été significativement meilleures une fois l'augmentation des images effectuée avec YOLOv3. De plus, même avec l'espèce de poissons avec le moins d'échantillon, le modèle a bien performé.

Pour ce qui est de Mask-RCNN, tout comme avec YOLOv3, les performances de la détection et de la classification ont été grandement améliorées.

Le résultat le plus élevé pour YOLOv3 est de 0.75 et pour Mask-RCNN on a 0.62. Cela démontre qu'il est bien possible de détecter et de classer les différentes espèces de poissons avec l'aide de la vision informatique et de vidéos de poissons.

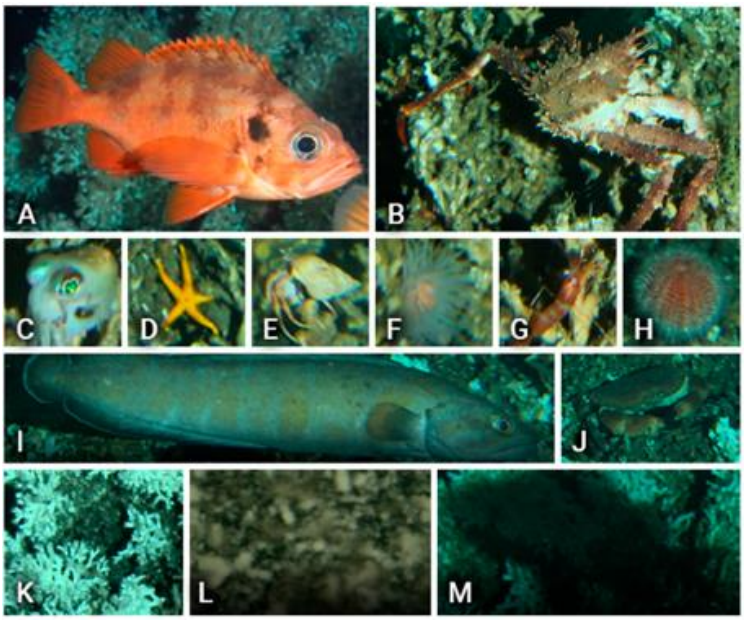
2.2 Video Image Enhancement and Machine Learning Pipeline for Underwater Animal Detection and Classification at Cabled Observatories

Cet article a été rédigé par Vanesa Lopez-Vazquez, Jose Manuel Lopez-Guede, Simone Marini, Emanuela Fanelli, Espen Johnsen and Jacopo Aguzzi en janvier 2020. Dans cet article le but est d'identifier et de faire la classification des différentes espèces marines avec à l'aide d'images. Cet article est pertinent, car les chercheurs ont testé une multitude d'algorithmes et de modèle différent avec différentes combinaisons d'hyperparamètre. De plus, un gros traitement a été effectuer sur les images pour les rendre les algorithmes de classification plus performant sur ceux-ci.

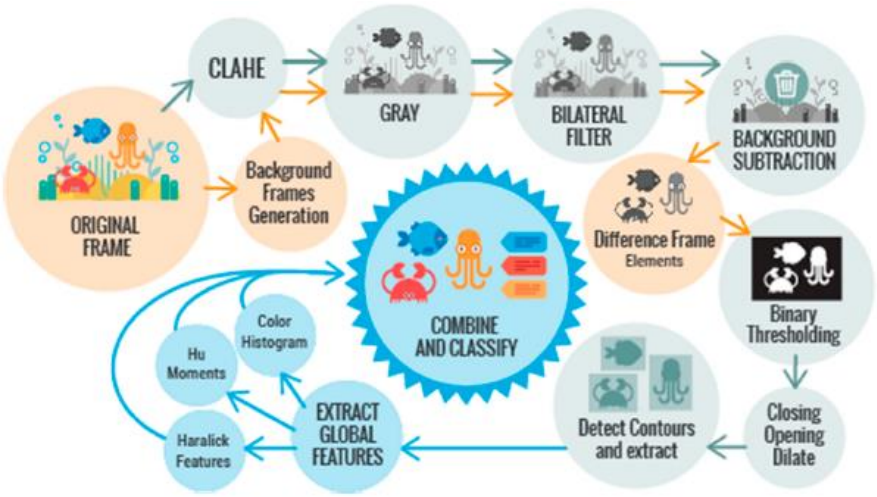
2.2.1 Données

Les données ont été collecté par un observatoire qui se situe à 20km des îles Lofoten en Norvège et à 260m de profondeur. Il se situe dans une crevasse qui à été créée par le mouvement des glaciers. L'observatoire en question porte le nom de: LoVe. Le jeu de données comporte les espèces ci-dessous. Voici aussi quelques exemples d'images :

Class (alias)	Species Name	# Specimens per Species in Dataset	Image in Figure 2
Rockfish	*Sebastes* sp.	205	(A)
King crab	*Lithodes maja*	170	(B)
Squid	*Sepiolidae*	96	(C)
Starfish	*Unidentified*	169	(D)
Hermit crab	*Unidentified*	184	(E)
Anemone	*Bolocera tuediae*	98	(F)
Shrimp	*Pandalus* sp.	154	(G)
Sea urchin	*Echinus esculentus*	138	(H)
Eel like fish	*Brosme brosme*	199	(I)
Crab	*Cancer pagurus*	102	(J)
Coral	*Desmophyllum pertusum*	142	(K)
Turbidity	-	176	(L)
Shadow	-	101	(M)



Le jeu de données compte un total de 8818 images. Les images obtenues proviennent d'un environnement non-contrôlé ce qui implique que les images sont caractérisées par un fonds de buissons coralliens où la turbidité et l'éclairage rendent difficile la détection d'entité avec le machine learning. Un processus de traitement d'image a été implémenté afin de rendre les images plus aptes à être évaluer par un modèle de machine learning. De plus pour alléger le traitement des images, les images ont été redimensionné, à la base elle était sous format de 3456 x 5184 pixels, puis elles ont été converties en 964 x 1296 pixels. Cela rend le traitement beaucoup plus rapide et léger. Voici un schéma du traitement qui est effectué.



2.2.2 Modèles et Algorithmes

Pour ce qui est des modèles et des algorithmes, dans cet article, une grande quantité d'algorithmes et de modèles ont été testé. Entre autres les algorithmes suivants : Support vecteur machine (SVM), méthode des k plus proche voisin (K-NN), Forêt aléatoire (RF), Réseaux de neurones convolutifs (CNN or ConvNets). Plusieurs versions de ces algorithmes ont été testés avec différent paramètre. De plus, des optimiseurs ont passés les paramètres en revus pour trouver la meilleure combinaison possible.

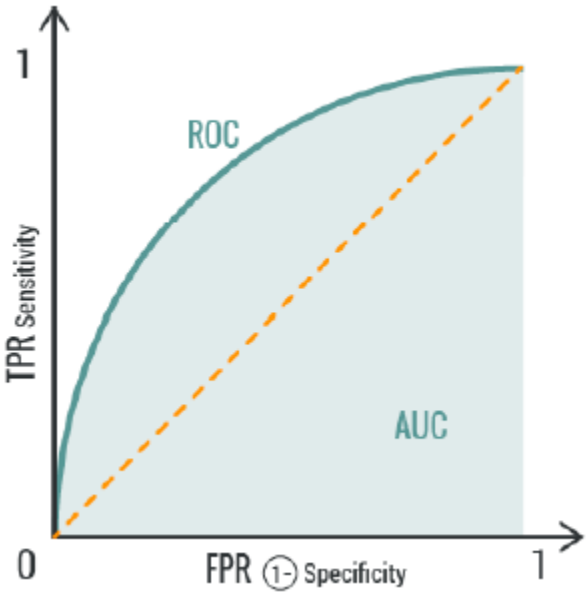
2.2.3 Métriques

Les métriques utiliser pour présenter les résultats sont la précision (accuracy) présenté ci-dessous.

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FP + TN + FN}$$

Où **TP** représente les vrais positif (True Positive), **TN** les vrais négatif (True Negative), **FP** représente faux positif (False positive), **FN** représente faux négatif (False negative), **P** représente vrai positif et **N** vrai négatif.

Une autre métrique utilisée est la AUC, une mesure de l'aire sous la courbe dans un graphique ou se croise le TPR et le FPR.



$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

2.2.4 Résultats

Pour ce qui est des résultats voici un tableau qui répertorie la précision (Accuracy) et l'aire sous la courbe pour l'ensemble des algorithmes et des modèles de classifications qui ont été testés.

Table 4. Accuracy and area under the curve (AUC) values with test dataset and training time obtained by different models.

Type of Approach	Classifier	Accuracy	AUC	Training Time (h:mm:ss)
Traditional classifiers	Linear SVM	0.5137	0.7392	0:01:11
	LSVM + SGD	0.4196	0.6887	0:00:28
	K-NN (k = 39)	0.4463	0.7140	0:00:02
	K-NN (k = 99)	0.3111	0.6390	0:00:02
	DT-1	0.4310	0.6975	0:00:08
	DT-2	0.4331	0.6985	0:00:08
	RF-1	0.4326	0.6987	0:00:08
	RF-2	0.6527	0.8210	0:00:08
DL	CNN-1	0.6191	0.7983	0:01:26
	CNN-2	0.6563	0.8180	0:01:53
	CNN-3	0.6346	0.8067	0:07:23
	CNN-4	0.6421	0.8107	0:08:18
	DNN-1	0.7618	0.8759	0:07:56
	DNN-2	0.7576	0.8730	0:08:27
	DNN-3	0.6904	0.8361	0:06:50
	DNN-4	0.7140	0.8503	0:07:16

On voit que les classifieurs classiques atteignent, au maximum, une précision de 65.27 % pour un temps d'entrainement de 8 secondes dans le cas du RF-2 (Random-forest 2). Pour ce qui est des modèles de deep learning, le score de précision le plus bas qui a été atteint est de 61.91%, ce score a été atteint par le réseau de neurone convolutif 1. Le réseau ayant atteint le plus haut score de précision est le réseau de neurones récurrent 1 avec le score de 76.18%. Cependant, on peut observer que les temps d'entrainement sont beaucoup plus longs pour ce genre de modèle que pour les classifieurs classique. En effet, le temps d'entrainement du meilleur modèle DNN-1 est de 7 minutes 56 secondes.

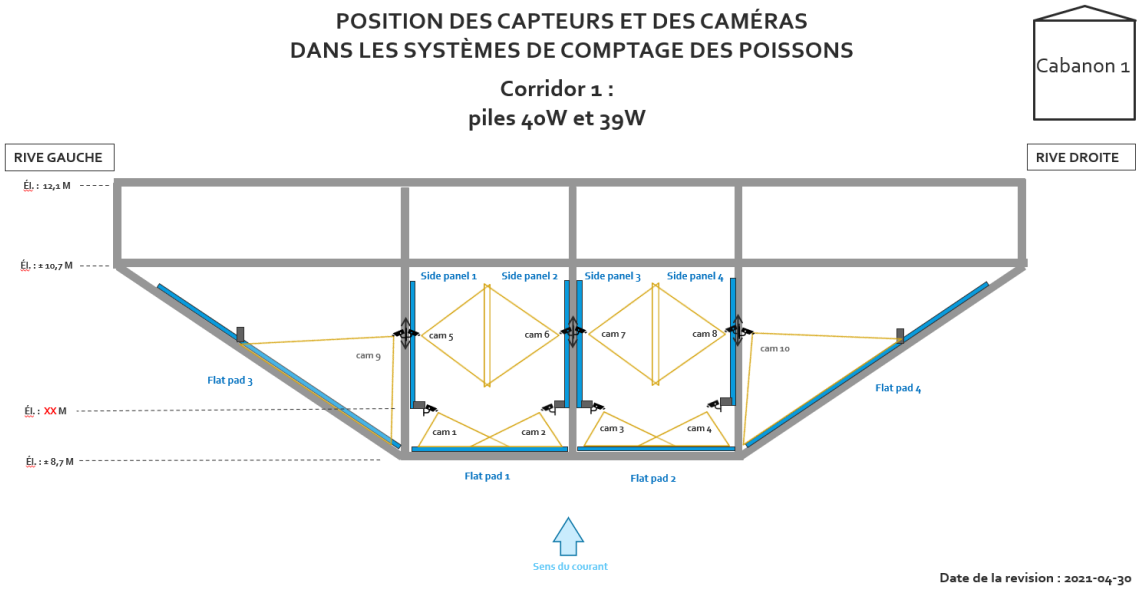
3. Données

Cette section a pour but de présenter la nature et la provenance des données, ainsi que les techniques utilisées pour le nettoyage des données.

3.1 Collecte

Les données utilisées dans le cadre de ce projet de détection, identification, classification et journalisation automatique des poissons sont constitué de centaines de vidéos collectées à l'été 2021 par NHSL. Ces vidéos ont été récoltées dans deux jetées[1]. Les données ont été collectées à partir de plusieurs caméras qui avaient toutes des angles de capture différents. Les caméras 1, 2, 3 et 4 étaient les caméras présentant les meilleures images, alors seule celle-ci ont été utilisées pour le projet. Voici un schéma qui montre comment les caméras étaient placées dans les jetées.

Figure 3.1.1



[1] Canaux pour le passage des poissons construit pour la déconstruction du pont Champlain à Montréal

3.2 Spécificité

Les images ont été capturées sous un format de 480 pixels de hauteur et 704 pixels de largeur à une fréquence de 30 images par seconde. Les séquences vidéo sont d'une durée de 5 minutes chacune. Chaque séquence est donc constituée de 9000 images. Chaque disque dur contient environ 8 téraoctets de données. Le jeu de données est constitué de 12 disques durs pour un total d'environ 96 téraoctets, soit 96 000 gigaoctet de données.

Pour ce qui est de la qualité des données récoltées, certaines images sont excellentes (figure 3.2.1), c'est-à-dire qu'elles procurent une bonne visibilité et on voit bien passer les poissons durant toute la trame. Cependant, certaines images sont beaucoup moins utiles, car les images sont de moins bonne qualité (figure 3.2.2) à cause de la lumière, du courant, de la turbidité, des vagues, des bulles d'air, des algues etc. De plus, certaines vidéos ne contiennent aucun poisson et d'autres en contiennent plusieurs. De manière général, plusieurs segments de vidéo ne sont pas utiles, car il ne se passe rien, c'est pourquoi le nettoyage des données est indispensable.

Figure 3.2.1 Image de bonne qualité

Figure 3.2.2 Image de mauvaise qualité

3.3 Construction du jeu d'entrainement

3.3.1 Tri

Afin d'entrainer un modèle de Machine Learning, il est nécessaire de construire un jeu de données contenant des images des images des objets que l'on souhaite détecter et identifier avec ce modèle. De ce fait, il est nécessaire de construire un jeu de données annoté contenant des images des différentes espèces de poissons présentes dans le fleuve pour ce projet.

Afin de créer ce jeu de données, il a d'abord été nécessaire de trier les images contenues dans la base de données afin d'en extraire seulement les images contenant des individus. Pour ce faire, un algorithme de tri a été utilisé. L'algorithme reçoit en entrée, des fichiers vidéo de 9000 images chacun, il effectue quelques opérations, puis donne en sortie les images contenant beaucoup de mouvement. Grace à cet algorithme, le nombre d'images à analyser manuellement pour la construction du jeu de données à annoter a été grandement diminué.

L'algorithme de tri est un algorithme de segmentation d'arrière-plan/avant-plan basé sur un mélange gaussien. Cet algorithme permet d'obtenir un masque blanc sur un fond noir (figure 4.1.1). Le masque blanc représente les parties en mouvement dans l'image. Ensuite, on érode et dilate l'image afin de réduire le bruit, puis on calcul la valeur moyenne. La valeur des pixels, après traitement est de 0 (blanc) ou 255 (noir). Cela étant, une fois la valeur moyenne de tous les pixels de l'image calculée, les images ayant une valeur moyenne se situant au-dessus d'un certain seuil arbitraire sont enregistrées dans un dossier à part pour être triées manuellement par la suite.

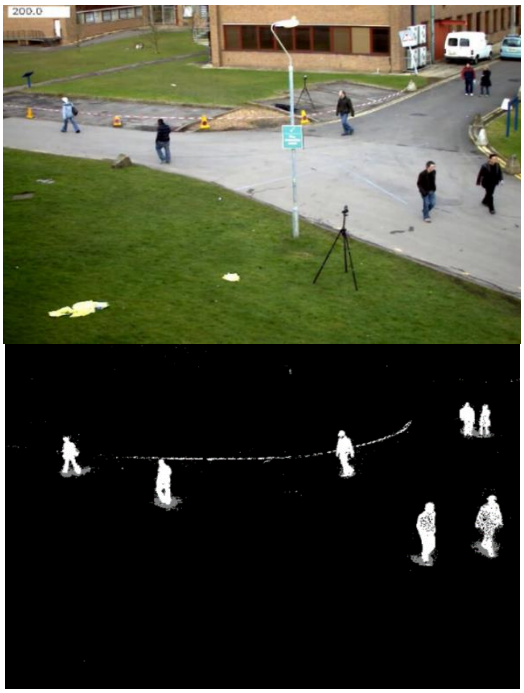
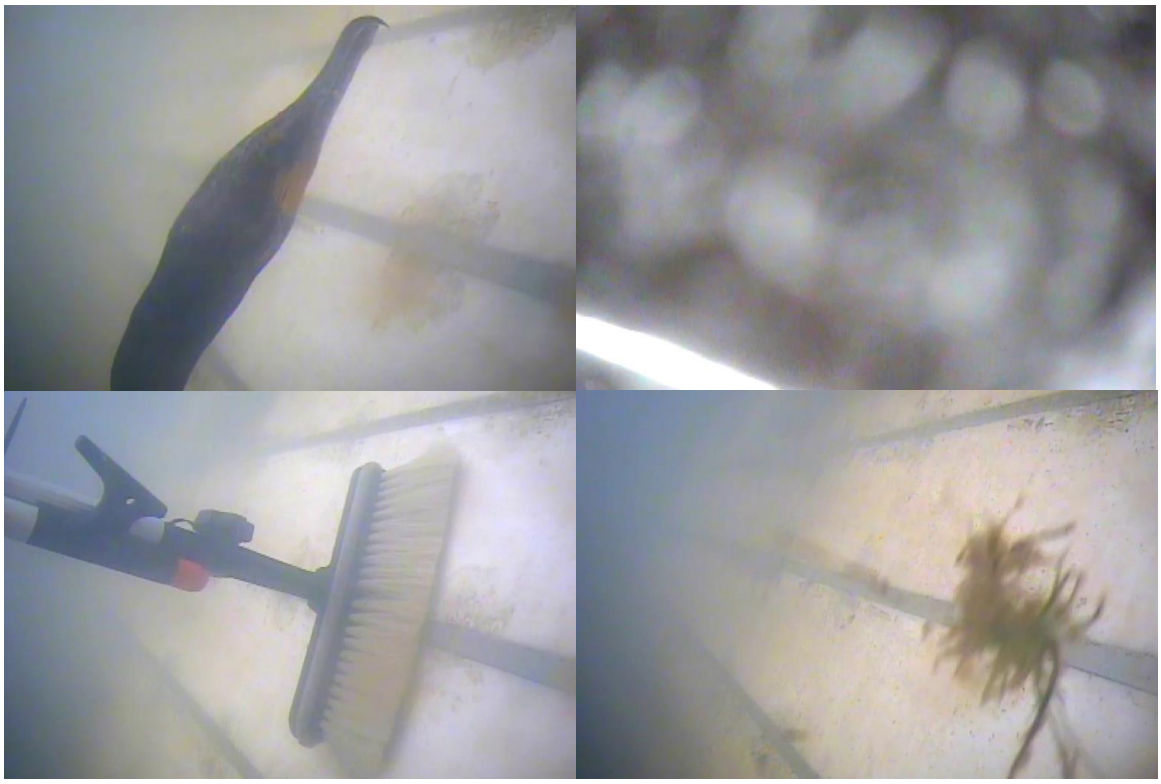


Figure 4.1.1 Image avant/après le traitement

L’algorithme de tri réduit considérablement le nombre d’images à analyser manuellement pour la suite, mais il n’est pas parfait, car il permet seulement de détecter le mouvement, mais pas quand une image contient un individu (figure 4.1.2). Cela fait en sorte que l’algorithme isole aussi des photos où l’objet en mouvement est simplement une algue, un cormoran, des turbidités ou autre. Un traitement manuel est encore nécessaire par la suite, mais le volume de données à analyser est réduit au maximum.

Figure 4.1.2 Exemple d’images détectées par l’algorithme de tri



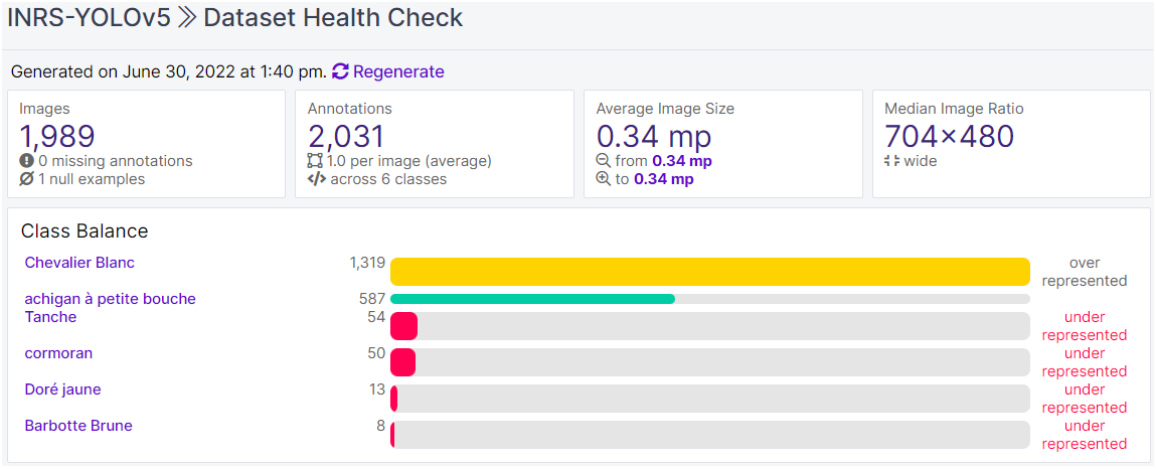
3.3.2 Annotation

Une fois les données triées une première fois par l'algorithme, il est nécessaire de trier les données manuellement afin de construire un jeu d'entrainement de qualité pour entrainer le modèle. Il est nécessaire de sélectionner des photos où l'on voit bien les poissons, mais aussi des images où on les voit un peu moins bien. Une fois un jeu de données de qualité construit, les images sont annotées selon le format demandé par le modèle de détection qui sera utilisé, soit YoloV5. Le format en question est une image couplée avec un fichier texte représentant les classes identifiées dans l'image et leur position. C'est ici que l'espèce de poissons est associée à un segment de l'image. La plateforme utilisée pour le l'annotation des images est RoboFlow. RoboFlow est une plateforme web d'annotation d'images en ligne qui permet d'effectuer les tâches d'annotation de façon collaborative. L'opération d'annotation consiste à placer une boîte précisément sur l'objet à identifier, puis de spécifier de quel objet il s'agit.

Figure 4.2.1 Exemple d'image annoté avec Roboflow

Après l'annotation du premier lot d'image, 1989 images ont été annotées, 6 classes différentes ont été identifiées. Voici un aperçu de la constitution et de la répartition des données annotées (figure 4.2.2)

Figure 4.2.2 Information sur la première version du jeu de données



On peut observer que l'espèce des chevaliers blancs représente plus de la moitié du jeu de données. De plus, les classes telles que les tanches, les cormorans, les dorés jaunes et les barbottes brunes sont sous-représentées. Cela pourrait grandement affecter la qualité des prédictions du modèle.

3.3.3 Augmentation

La plateforme Roboflow permet aussi de faire de l'augmentation[2] sur les données. Plusieurs jeux d'entrainement ont été créés, ces jeux de données sont tous différents. Ils ont tous été testés en entrainement pour voir lequel produisait les meilleurs résultats.

La première version est une version qui contient uniquement des images de poissons, cette version ne contient aucune image d'arrière-plan sans poissons ou d'image d'algues etc. Cette version se nomme yolov5-fish.

[2] Processus d'augmentation de la quantité de données d'entraînement par la création de nouvelles données à partir des données existantes.

Le deuxième jeu de données se nomme yolov5-fish-augmentation. Comme son nom l'indique, cette version implémente aussi du traitement d'image qui a pour but de d'améliorer les performances du modèle. L'augmentation des données comportait le renversement des images, les rotations à 90°, changement du niveau de gris de 25%, changement de luminosité de -25% et +25%, changement de l'exposition -14% et +14%, ajout de flou jusqu'à 2.5 pixels et finalement, ajout de bruit jusqu'à 2% dans les images.

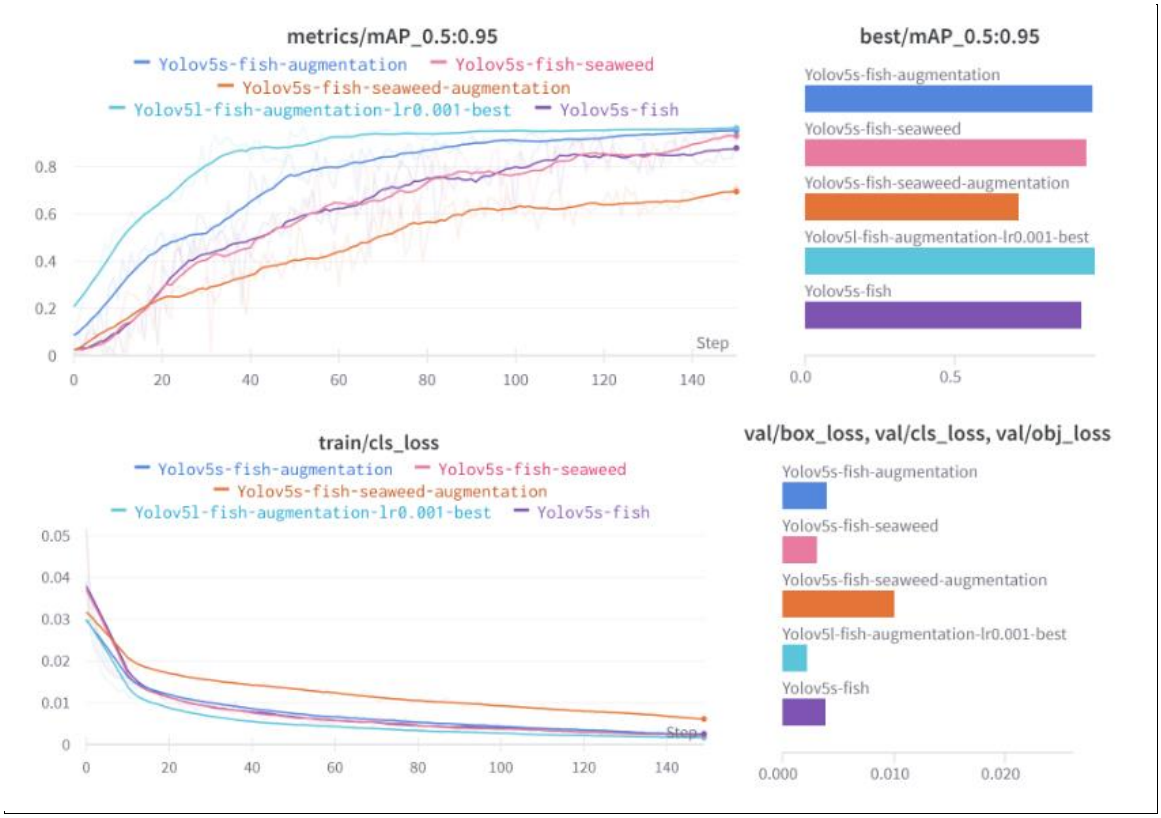
Le troisième jeu de donnée se nomme yolov5-fish-seaweed. Dans ce jeu de données, il n'y a pas d'augmentation d'image. Cependant, environ 200 images d'arrière-plan, d'algue etc. ont été ajouté dans le jeu de données. Ces images ont été ajouté dans le but d'apprendre au model à reconnaitre un arrière-plan sans poisson et de différencier une algue d'un poisson.

Le quatrième jeu de donnée est en fait le troisième, yolov5-fish-seaweed, auquel il a été ajouté du traitement sur les images, comme pour le deuxième jeu de données. Cependant, les traitements apportés sont bien différents. Les traitements qui ont été appliqué sont les renversements d'image, les rotation 90°, changement du niveau de gris de 25%, les coupures de l'image, les décalages de l'image, les changements de teinte de -50° et +50°, changement de luminosité, l'ajout de flou jusqu'à 5 pixels, l'ajout de bruit jusqu'à 4% dans les images, puis 8 coupures pour un total de 13% de l'image.

Finalement, une dernière version yolov5-fish-augmentation-lr0.001-best, qui est une version de yolov5-fish-augmentation dans laquelle le taux d'apprentissage a été diminué dans le but de ralentir l'apprentissage.

Les graphiques ci-dessous (figure 4.2.3) permettent de conclure que l'ensemble des jeux de données, sauf Yolov5-fish-seaweed-augmentation, ont permis au modèle d'apprentissage profond YoloV5 d'atteindre une précision moyenne (mAP) plus haute que 90% après 150 époques d'entrainement. De plus, il est possible d'observer que le modèle ayant appris le plus rapidement est le jeu de données Yolov5-fish-augmentation-lr0.001-best. C'est aussi ce jeu de données qui a obtenu le meilleur mAP après 150 époques.

Figure 4.2.3 Graphiques de comparaison des versions du jeu de données



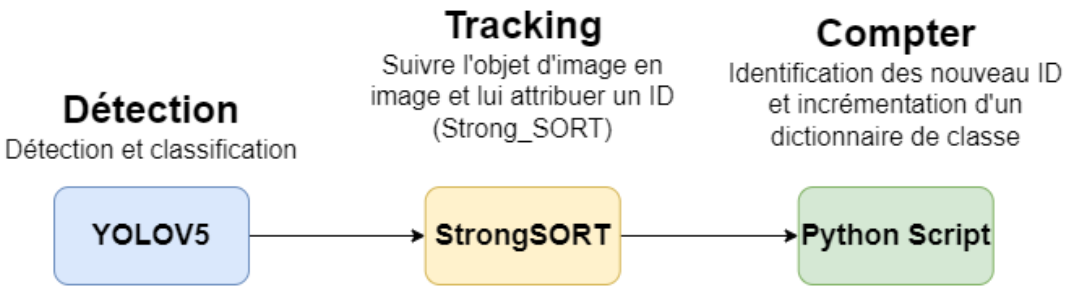
Il est possible de conclure que le modèle a performé vraiment moins bien avec le jeu de données Yolov5-fish-seaweed-augmentation. Les raisons pourquoi ce jeu de données procure de moins bons résultats sont nombreuses. Entre autres, le traitement d'image, auquel ce jeu de données a été soumis, modifie énormément les images originales, beaucoup plus que dans le cas de Yolov5-fish-augmentation. En effectuant cette augmentation probablement trop agressive, les images originales ne ressemblaient plus assez au genre d'image que le modèle peut voir.

3.4 Amélioration du Jeu de Données Initial

Comme la répartition des données du premier jeux de données était mauvaise, il a été nécessaire d'améliorer ce jeu de données. Il a été nécessaire de repasser des données dans l'algorithme de tri et de les annoter, puis de construire un nouveau jeu de données qui comporte des images des caméras utilisé, soit 1,2,3,4. De plus, dans cette version, il est important d'essayer d'obtenir une meilleure répartition des classes. (en cours)

4. Fonctionnement

Cette partie porte sur le fonctionnement du programme permettant d'effectuer la détection, la classification, le tracking et le vidéo comptage des poissons. Le programme se décompose en plusieurs parties. Premièrement, la détection et la classification des objets. Deuxièmement, le tracking des objets et finalement le comptage des objets. Éventuellement, une partie journalisation dans une base de données devra être implémenté.

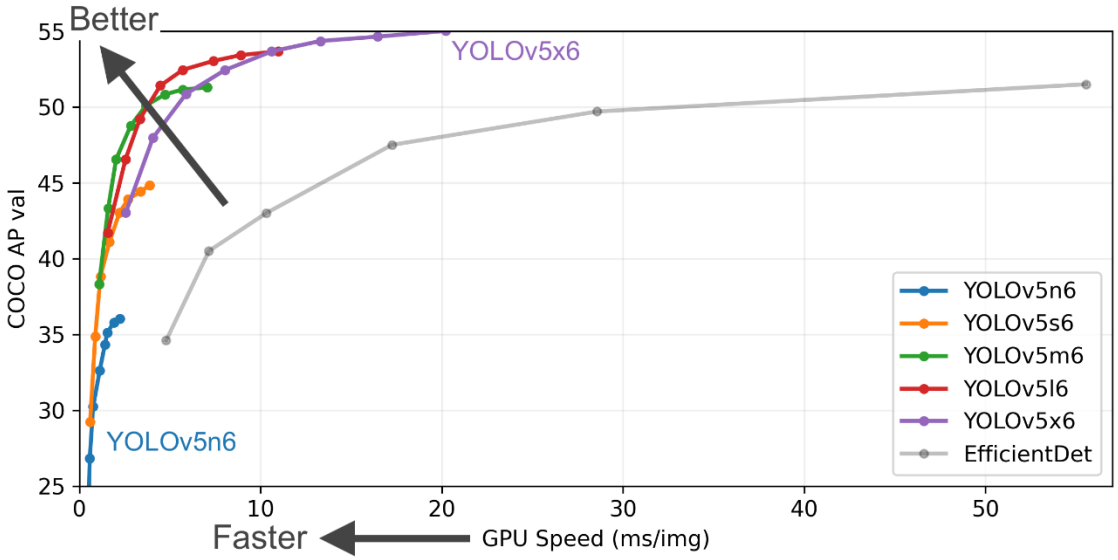


4.1 Détection

La détection des objets est la base de l'entièreté de ce projet, sans cette partie, la suite n'est pas possible. Un modèle de détection est construit pour créer des features à partir des images en entrée, puis les fournir à un système de prédiction afin de dessiner des boîtes alentour des objets et prédire leur classe.

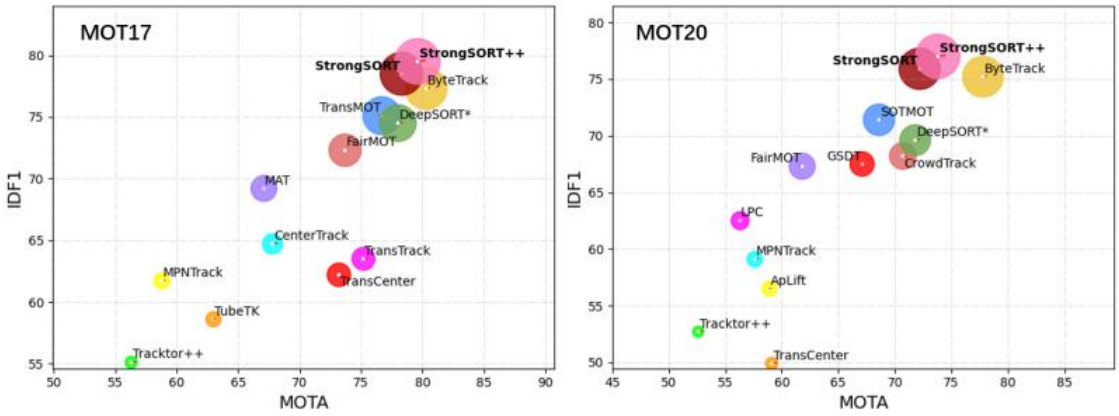
Le modèle utilisé pour la détection a été le modèle YoloV5, car il présente des caractéristiques très intéressantes qui correspondent totalement au projet. YoloV5 est un modèle qui a été développé dans le but de rendre son utilisation très simple, même avec un jeu de données personnalisé. Ce modèle permet d'effectuer la détection en temps réel et il est très robuste. L'une des raisons pourquoi il est si robuste est son architecture qui implémente automatiquement l'augmentation des données de son apprentissage automatique des encrages des boîtes englobantes.

Il existe plusieurs versions de YoloV5, la rapidité et la précision de ces versions varient beaucoup. Le diagramme ci-dessous montre les caractéristiques de chacune des versions. Dans le cadre du projet, la version utilisée a été YoloV5l, cette version présente un bon compromis entre la rapidité et la précision. La version utilisée pourrait être modifiée au besoin.

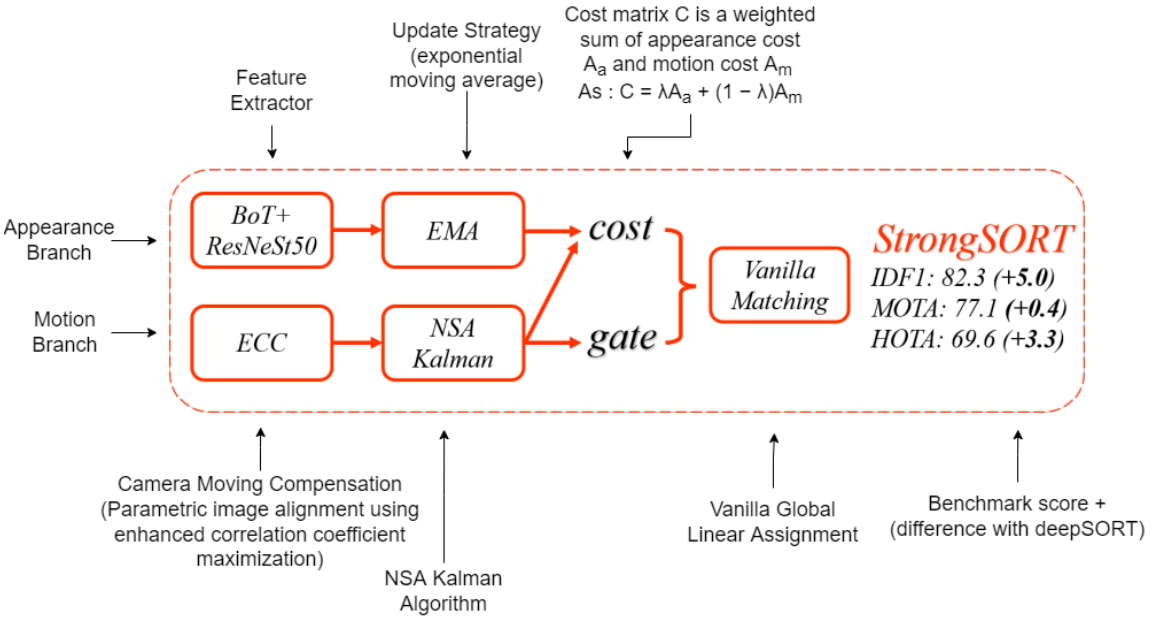


4.1 Tracking

Un des plus grands défis lors de ce projet a été d'associer les objets détecter dans les images d'une séquence, puis de déterminer s'il s'agit d'un seul objet. En effet, la détection des objets et la classification s'effectue image par image. Le modèle YoloV5 ne peut pas déterminer si un objet de l'image x_{i-1} et un objet de l'image x_i sont les mêmes, où i est l'index de l'image. Il est donc impossible de compter le nombre d'individus présent sur une séquence vidéo uniquement à l'aide de YoloV5. C'est pourquoi l'ajout d'un modèle de type SORT (simple online and real time tracking) est nécessaire. Il existe plusieurs modèles de tracking. Dans le cadre du projet, le modèle utilisé est StrongSORT, car c'est un des modèles qui présente les meilleures performances et il est conçu pour s'ajouter à un modèle de détection.



Le modèle StrongSORT se compose de deux branches, une branche dédiée à l'apparence et une branche dédiée au mouvement. L'ajout de ce modèle permet de suivre les objets au travers des images, puis de leur donner un identifiant unique (ID).

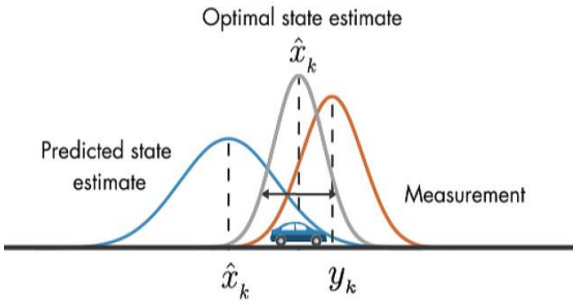


La branche dédiée à l'apparence se compose d'un premier composant, un extracteur de caractéristiques (BoT) basée sur ResNeSt50 et pré-entrainé sur le jeu de données DukeMTMC-reID. Ce réseau est utilisé pour extraire les caractéristiques physiques des objets. En addition, cette branche contient un deuxième composant qui est une stratégie de mise à jour. Cette stratégie permet de mettre à jour l'état d'apparence d'un objet e_i^t pour la i^e mini-trajectoire à l'image t en utilisant la moyenne mobile exponentielle (EMA) telle que :

$$e_i^t = \alpha e_i^{t-1} + (1 - \alpha) f_i^t,$$

Où f_i^t est l'intégration de l'apparence de la détection correspondante actuelle et $\alpha = 0{,}9$ est un terme de quantité de mouvement. La moyenne mobile exponentielle ne fait pas qu'améliorer la qualité des correspondances des objets, elle améliore aussi le temps de calcul.

Pour ce qui est de la branche reliée au mouvement, elle est composée d'un premier composant, l'alignement d'image paramétrique à l'aide d'une maximisation améliorée du coefficient de corrélation (ECC). Ce composant prend en charge la compensation de mouvement des caméras.



Le deuxième composant de cette branche est le NSA Kalman filter, une version améliorée du filtre de Kalman linéaire qui permet de calculer de manière adaptative la covariance du bruit.

Ensuite, l'information de la branche d'apparence et celle du mouvement est utilisé pour résoudre le problème d'affectation. La matrice des frais (Cost matrix) C est une somme pondérée des frais d'apparence A_a et des frais de mouvement A_m comme suit :

$$C = \lambda A_a + (1 - \lambda) A_m$$

Où le facteur de poids λ est établie à 0.98.

Finalement l'assignement est réalisé grâce à l'affectation linéaire globale (Vanilla Global Linear Assignement). En résultat, on obtient une vidéo où les objets sont détectés, classés et suivis. Cela permet de leur attribuer un ID et de tenir un compte des individus. L'implémentation est simple, lors de la détection d'un nouvel ID, le programme vérifie si l'ID était déjà répertorié dans la liste d'ID. S'il ne l'est pas, alors cela signifie qu'il y a un nouvel individu qui est détecté, alors on incrémente la valeur de la classe de l'individu dans le dictionnaire qui sert de compteur.