

Étudiant : **TRAN Thanh Duy**

Maitre de stage
Tuteur pédagogique

: **Jean-Christophe Burie**
: **Muriel Visani**

RAPPORT DE STAGE

SUBIA

Laboratoire Informatique, Image, Interaction

Université de La Rochelle

Master 1 ICONE – Formation Initiale

Laboratoire Informatique, Image, Interaction

Université de La Rochelle



SOMMAIRES

Résumé	3
Remerciement	5
Introduction	6
Université de La Rochelle	6
Laboratoire Informatique, Image et Interaction	6
Présentation du stage	7
Chapitre 1 : Les études et les approches pertinentes	9
Chapitre 2 : Base théorique	10
Vision numérique – évolution de la nature vers l’Intelligence Artificielle.....	10
Réseau Neuronal Convolutif (RNC) – le cœur de la Vision Numérique.....	10
Inception.....	12
MobileNet.....	13
Détection d’Objet	13
Chapitre 3 : Résultats Expérimentaux	15
Préparation de l’environnement	15
Plateforme : Google Colaboratory	15
Langage de programmation : Python.....	15
Framework : Tensorflow	16
Les autres outils, librairies utilisés.....	16
Préparation des données.....	16
Format de données	16
Format des labels	17
Prétraitement des données brutes	18
Sous-étape 1 : Éliminer les données cachées et les répertoires (« Nettoyage de données »)	18
Sous-étape 2 : Éliminer les données avec format incorrect (« Identification du format »)	18
Sous-étape 3 : Équilibrer le nombre de données et labels (« Synchronisation de données »)	19
Partitionnement des données	19
Distribution des espèces marines dans les jeux de données	19
Conversion des annotations de texte en PASCAL VOC.....	21
Générer les structures de données particulières (TFRecord, Label Map).....	22
Protocol Buffer : une structure donnée particulière	22
Label Map : référence des labels.....	23
Customisation des paramètres et Entrainement des modèles	23
Common Objects in COntext (COCO).....	23
Oxford-IIIT Pet.....	23
Choisir les modèles disponibles.....	24
Méta-Graphe (Meta Graph)	25

Point de Contrôle (Checkpoint)	26
Évaluation de la performance et de la qualité des modèles	26
Commentaires Préliminaires	29
Charger le modèle entrainé	29
Problèmes techniques restants	29
Conclusion	31
Références	32
Annexe A: Implémentation du projet SUBIA	35

Résumé

Protéger et maintenir la biodiversité - en général, et des espèces marines en particulier, sont l'une des tâches les plus importantes de l'humanité de nos jours. Pour La Rochelle, ville où la mer joue un rôle important dans la vie quotidienne ainsi que la culture, ce problème est encore plus important. Suivre ainsi que évaluer le développement des espèces marines va fournir un panorama aux experts afin de protéger l'abondance biologique marine. L'objectif de ce stage est à présenter une approche utilisant plusieurs modèles d'Apprentissage Profond basés sur les algorithmes de Single-Shot Detector (SSD) pour détecter les espèces marines. Tout d'abord, l'ensemble de données brutes sous forme d'images sera labellisé pour identifier la présence de chaque espèce. Ces annotations, ensuite, seront converties en format PASCAL Visual Object Classes (PASCAL VOC) basé sur eXtensible Markup Language (XML) pour pouvoir les utiliser pour l'entraînement des modèles SSD. Par la suite, les modèles entraînés,



y compris la structure et l'ensemble de paramètres, seront stockés sous forme de métagraph et de point de contrôle, afin de générer un graph d'inférence gelé - une structure de données particulière permettant le déploiement des modèles à tout moment sans ré-entrainer le modèle. La qualité de chaque modèle sera évaluée en fonction de la fonction de perte, de la stabilité du processus d'entraînement, et des critères de COCO ; est puis visualisée en utilisant l'outil Tensorboard. Pour ce projet, la mise en œuvre de tous les modèles de ce projet est entièrement basée sur Tensorflow - un framework open-source d'Intelligence Artificielle développé par Google.

Mots clés : Biodiversité, espèces marines, Apprentissage Profond, Pascal VOC, Single-Shot Detector, Tensorflow

Protecting and maintaining biodiversity in general, and marine species particularly is one of humanity's most important tasks nowadays. For La Rochelle, a city that the sea plays an important role for daily life as well as culture, this matter is even more urgent. The tracking through each period as well as the evaluation of the development of marine species would provide the panorama for experts in protecting the marine biological abundance. The purpose of this topic is to present an approach using many Deep Learning models based Single-Shot Detector (SSD) to detect the marine species. Firstly, the raw dataset being raw image will be labeled to identify the name of each species. These annotations, after that, will be converted to Pascal Visual Object Classes (Pascal VOC) in form of eXtensible Markup Language (XML) in order to well adapt with SSD model training. Subsequently, the training models including the architecture and the set of parameters and weights will be stored at meta-graph and checkpoint and then generate to a frozen inference graph – a special data structure which allows the deployment of models anytime without re-training the model. The quality of each model will be assessed based on its loss function, stability of the training process, and COCO criteria; and is visualized using the Tensorboard. For this project, the implementation of all models in this project is based entirely on Tensorflow – an open-source Artificial Intelligent framework developed by Google.

Keywords : Biodiversity, Marine species, Deep-Learning, Pascal VOC, Single-Shot Detector, Tensorflow

Remerciement

En premier lieu, j'adresse mes sincères remerciements à mon maître de stage, Monsieur Jean-Christophe Burie, le Directeur Adjoint de Laboratoire Informatique, Image, Interaction, Université de La Rochelle, qui m'a beaucoup supporté pendant 10 semaines de mon stage ainsi que pour son accueil, son partage des connaissances et ses guidances dans chaque réunion.

Je voudrais également dire mes profonds remerciements au mon tuteur pédagogique, Madame Muriel Visani, pour ses précieuse expertises dans la conduite de ce stage ainsi que de la rédaction de ce rapport. En outre, ses instructions du parcours de Données comprenant structure, stockage et visualisation des données sont une partie indispensable de ce stage.

Je désire aussi remercier Monsieur Jean-Loup Guillaume, Responsable du programme de Master, Monsieur Arnaud Revel et Monsieur Patrick Franco, Responsables du Stage et Madame Erlandri Chaigneaud, Secrétaire du Département Informatique, Université de La Rochelle. Ils ont toujours été à mon écoute et ont su m'apporter un soutien sans faille, notamment en ce qui concerne les démarches administratives relatives à mon stage ainsi que me supporter à chercher ce stage.

De plus, je voudrais remercier également tous les enseignants et les enseignantes qui m'ont enseigné tout au long de l'année scolaire. Les connaissances acquises ont largement contribué au succès de ce stage.

Pour finir, un grand merci à ma famille et mes camarades, pour leur conseils, ainsi que pour leurs précises soutiens pendant mon stage, à la fois physiquement et mentalement.



Introduction

Université de La Rochelle



Situé en centre de la ville, et étant 1 des 5 universités publiques de la région Nouvelle-Aquitaine (outre de l'Université de Pau, Poitiers, Limoges, et Bordeaux), Université de La Rochelle est la plus jeune université de la France. La première pierre de l'Université est posée en 22 mai 1992 par François Mitterrand, Président de la République Française, Helmut Kohl, Chancelier de la République Fédérale d'Allemagne, Michel Crépeau, Député-Maire de La Rochelle, François Blaizot, Président du Conseil Général de la Charente-Maritime et Jean-Pierre Raffarin, Président du Conseil Régional de Poitou-Charentes.

L'université compose 3 Facultés et 3 Instituts, y compris :

1. Faculté de *Droit, Science Politique et Gestion* avec 2 départements : Droit et Science Politique, Management
2. Faculté des *Lettres, Langues, Arts et Sciences Humaines* (FLASH) avec 4 départements : Langues Étrangères Appliquées, Lettres Modernes, Sciences Humaines et Sociales, et Centre Universitaire de Français Langue Étrangère (CUFLE)
3. Faculté des Sciences et Technologies avec 8 départements : Biologie, Biotechnologie, Chimie, Génie Civil, Informatique, Mathématiques, Physique, et Science et la Terre
4. Institut Universitaire de Technologie (IUT)
5. Institut d'Administration des Entreprises (IAE) – École Universitaire de Management
6. Institut Universitaire Asie-Pacifique

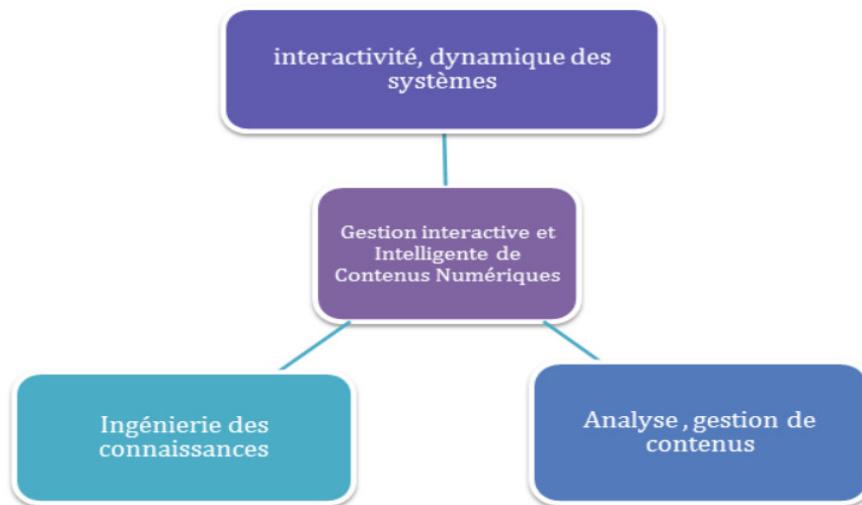
En outre, l'Université a 10 laboratoires de recherche, 4 fédérations de recherche labellisées par CNRS et 1 école doctorale pluridisciplinaire EUCLIDE. L'effectif de l'Université est près de 10,000 étudiants et près de 500 enseignants et chercheurs.

Laboratoire Informatique, Image et Interaction



Créé en 1993 et situé à la Faculté des Sciences et Technologies; Laboratoire Informatique, Image et Interaction (L3I) est une laboratoire de recherche du domaine des sciences du numérique de l'Université de La Rochelle. Depuis 1997, L3I a devenu l'Équipe d'Accueil – EA 2118 – et a été labellisé le label d'Équipe de Recherche Technologique (ERT) par le Ministère de la Recherche. En plus de promouvoir la recherche avec des partenaires nationaux, L3I entretient également de bonnes relations avec des partenaires internationaux avec des nombreux centres de recherche à travers le monde (Espagne, Japon, Vietnam, Malaisie, Tunisie). L'effectif de L3I de nos jours est près de 100 membres, tous situés sur le site de La Rochelle. Depuis 1er juillet 2016, L3I est porté par le directeur étant Monsieur Yacine Ghamri-Doudane et le directeur adjoint étant Monsieur Jean-Christophe Burie.

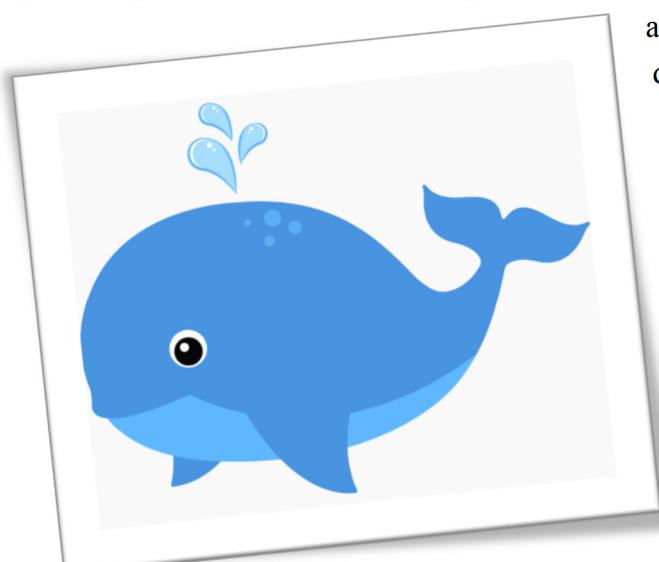
L'objectif et la stratégie de recherche de Laboratoire est à concerner à la gestion intelligente et interactive des contenus numériques à l'ère de l'interaction humaine et informatique devient plus intelligente et plus complexe. Il y a 3 domaines de recherche disponibles de L3I actuellement: *Modèles et Connaissances* (MC), *Images et Contenus* (IC), *Dynamique des Système et Adaptivité* (eAdapt).



La structuration de L3I (Source : <https://l3i.univ-larochelle.fr/Equipes>)

Présentation du stage

La conservation biologique est depuis longtemps une priorité pour l'humanité, et aujourd'hui elle devient de plus en plus une priorité en raison du changement climatique et de la dégradation de l'habitat. Une statistique de l'ONU [1]



affirme que la perte d'habitat actuelle est à un niveau alarmant, dans lequel la diversité des espèces terrestres indigènes a chuté d'au moins 20% depuis 1900. En outre, ce rapport a souligné qu'environ 1 million d'espèces de la Liste rouge sont menacées d'extinction dans les prochaines décennies en raison du développement industriel [1]. En termes d'espèces marines, la situation actuelle est très menacée. Une recherche de l'UNESCO montre qu'en 25 ans, de 1980 à 2005, au moins 30% des espèces marines mondiales ont disparu [2]. Pour La Rochelle, une ville côtière du sud-ouest de la France, la mer fait partie intégrante du peuple et du gouvernement de la ville, et

donc la conservation de la vie marine est une tâche urgente. Pour ce faire, la première chose à faire est de suivre et d'évaluer le développement des espèces marines à différents stades. L'apprentissage Automatique, notamment Deep-Learning, serait un choix parfait pour ce problème. Mon stage – SUBIA (SUivi de la Biodiversité par IA) - vise à mettre en place un suivi des espèces vivant sous les pontons du Port de Plaisance de La Rochelle en faisant appel à l'intelligence artificielle pour traiter d'énormes jeux de données d'images. L'objectif est extraire des données sur la biodiversité à partir d'images numériques sous-marines via des méthodes d'apprentissage et de reconnaissance basées sur les apprentissages profonds. Mon stage consistera à tester différentes architectures de Deep-Learning pour localiser et comptabiliser certaines espèces sous-



marines comme les ascidie, les pétoncles, les huitres, les moules, les bryozoaires. Ce rapport est divisé en 5 parties suivantes:

- Dans la première partie, ce rapport va présenter certaines études et approches pertinentes à la détection des espèces marines
- Ensuite, nous allons arriver à la Détection d'Objet, avec la colonne vertébrale étant le *Réseau Neuronal Convolutif (Convolutional Neural Network)* avec 2 modèles – Inception et Mobilenet
- La troisième partie va présenter les résultats expérimentaux, du traitement des données brutes à la configuration et ensuite au déploiement de différents modèles basés sur SSD
- Enfin, la conclusion résumera résultats obtenus via ce stage ainsi que les problèmes à résoudre et également présentera certaines perspectives de développement dans l'avenir
- En outre, les Annexes vont présenter les résultats obtenus dans ce stage sous forme graphique

Ce stage dure 10 semaines, à partir du 27 avril au 4 juillet 2020, et tout le travail se fait à distance en raison de la pandémie Covid-19. Généralement, ce stage est divisé en 6 étapes, chaque étape correspondant à 1 ou 2 semaines, en fonction de la complexité des tâches, y compris :

1. Étape 1: Renforcer les compétences techniques, particulièrement l'Apprentissage Profond, et Rechercher les documents et des études le concernant. Cette étape déroule en *1 semaine* – à partir de 27 avril à 3 mai
2. Étape 2 : Implémenter les scripts pour prétraiter les données et Visualiser la distribution de chaque espèces marines. Cette étape déroule en *1 semaine* – du 4 mai au 10 mai
3. Étape 3 : Implémenter et Déployer au fur et à mesure 61 modèles de détection d'objets sur la plateforme de Google Collaboratory pour identifier les modèles les plus convenables. Cette étape déroule en *2 semaines* – à partir du 11 mai jusqu'au 25 mai en raison des limites de l'infrastructure technique.
4. Étape 4 : Réimplémenter et Redéployer les 11 modèles les plus convenables pour SUBIA et Faire les statistiques pour évaluer la performance de chaque modèle. Cette étape déroule en *1 semaine* – du 25 mai au 31 mai
5. Étape 5 : Rédiger le rapport de stage avec le support du maître de stage et tuteur pédagogique. Cette étape sera finie le 12 juin
6. Étape 6 : Améliorer les modèles implémentés dans ce stage dans le sens de modification des paramètres et essaie-erreur. Cette étape sera déroulera après la soutenance du stage, jusqu'au 4 juillet



Chapitre 1 : Les études et les approches pertinentes

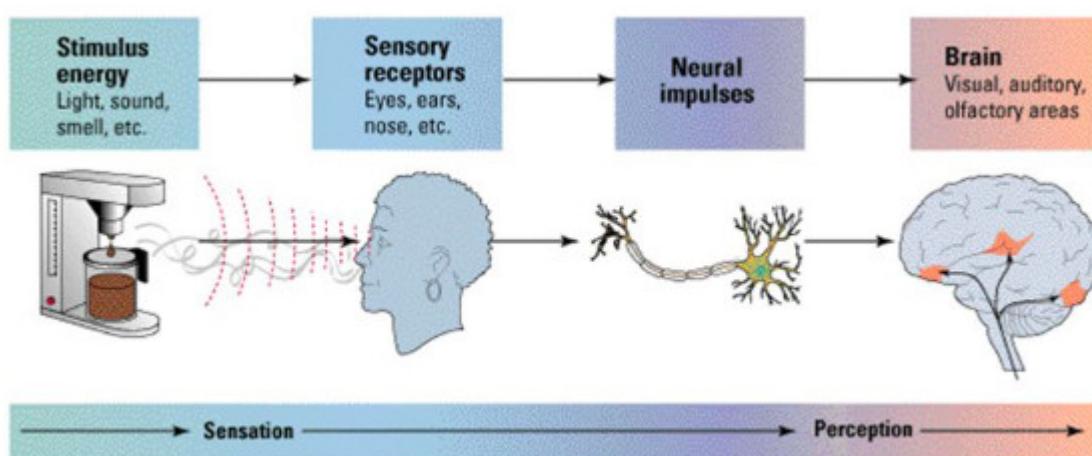
Détection des espèces marines est à la fois un problème fondamental en océanographie et un défi pour le domaine de l'Intelligence Artificielle. Néanmoins, actuellement, grâce à l'état de l'art de l'Apprentissage Automatique, de plus en plus de solutions modernes ont été proposées actuellement afin de chercher une solution idéale – intégrant tout en un.

Walther et al. [3] ont proposé l'algorithme d'attention sélective (*Selective Attention Algorithm – SAA*) afin d'analyser automatiquement les vidéos sou-marines. Itti et al. [4] ont suggéré l'approche de l'attention de manière ascendante basée sur la saillance (*Saliency-based bottom-up attention*) pour extraire les cibles saillances en utilisant le filtrage de l'intensité, couleur et échelles spatiales. Spampinato et al. [5] ont appliqué l'algorithme de *CamShift* pour automatiquement suivre les poissons basé sur les histogrammes des couleurs. Correia et al. [6] ont développé un modèle d'attention visuelle dans le but d'automatiquement détecter les homards de Norvège. Kim et al. [7] a construit PVANET – un modèle léger de détection d'objet en temps réel – pour détecter les poissons. Trucco and Olmos-Antillon [8] ont utilisé le modèle de Jaff McGlamery (*Jaff McGlamery model*) pour automatiquement détecter les espèces marines basés sur la transmission de radio. Mane et Pujari [9] ont développé un modèle pour détecter les mouvements dans les vidéos – basés sur le modèle Gaussien. Barat et Phlypo [10] ont proposé un système de détection automatique des contours – appliquée pour segmenter et puis détecter les objets sous-marines. Rizzini et al. [11] ont appliqué l'algorithme de détection de cible multi-caractéristique pour automatiquement détecter les ordures plastiques et les espèces marines. Yamashita et al. [12] ont adopté le modèle de synchronisation de couleur – pour détecter les espèces marines via le couleur et la température de l'eau. Le modèle de détection des poissons et coraux basé sur *Réseau Neuronal Convolutif*, proposé par Salman et al. [13] atteint une précision supérieure à 90%. Boussarie et al. [14] ont développé un système capable de détecter automatiquement en temps-réel les poissons via les vidéos en haute-résolution. Kannapan et al. [15] ont appliqué l'algorithme de segmentation des images pour suivre et évaluer l'évolution des organismes de pétoncles. Garcia et al. [16] ont implémenté l'identification et segmentation d'objets à travers le processus de segmentation générique afin de détecter l'occurrence des espèces marines. Sun et al. [17] ont proposé un algorithme de reconnaissance automatique pour l'identification des couleurs et les formes. Sermanet et al. [18] ont suggéré l'algorithme de OverFeat – basé sur *Réseau Neuronal Convolutif* afin de détecter, reconnaître et classifier les objets. Ren et al. [19] ont appliqué Faster-RCNN – combiné avec RPN pour générer des propositions de région – et puis Réseau Neuronal Convolutif pour la classification. En outre, Alsahwa et al. [20] ont proposé un modèle combiné utilisant HOG (*Histogram of Oriented Gradients*) et SVM (*Support Vector Machine*) pour détecter les espèces marines.

Chapitre 2 : Base théorique

Vision numérique - évolution de la nature vers l'Intelligence Artificielle

Les sens, en particulier, et les yeux, en général sont le cadeau de la nature aux l'humanité et à tous les êtres vivants de la planète. Grâce aux yeux, nous pouvons acquérir - percevoir - et traiter les images, formes et couleurs des objets alentours. Kassin et al. [21] ont défini la perception comme l'activité par laquelle un sujet fait l'expérience d'objets présents dans son environnement.



Acquérir – Percevoir – Traiter les informations alentours

(Source : Kassin, "Essential of Psychology" – ©2004 Prentice Hall Publishing)

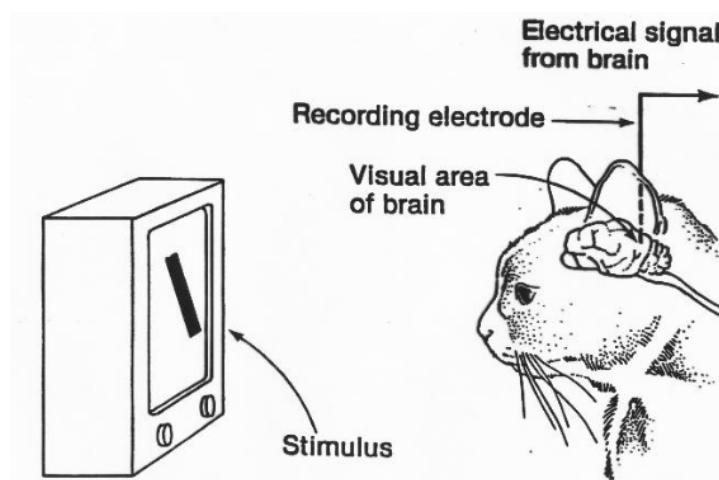
Actuellement, grâce aux progrès de la science, généralement et de l'Apprentissage Automatique, en particulier, nous avons redonné cette puissance aux ordinateurs. En exploitant les algorithmes avancés et les modèles d'Apprentissage en Profond, nous avons entraîné l'ordinateur à simuler les activités oculaires. La domaine qui étudie ces activités s'appelle Vision Numérique (*Computer Vision*) – une branche hybride de l'Apprentissage Automatique et de la Science Informatique.



Le potentiel d'application de la vision numérique est extrêmement vaste, de l'économie, finance à la médecine, l'automatisation des processus, le trafic. Certaines applications de la vision numérique peuvent être courantes dans la vie quotidienne sont comme la classification des images, la détection et reconnaissance des objets, le traitement des documents, la contextualisation des images et vidéos, les voitures autonomes,...

Réseau Neuron Convolutif (RNC) - le cœur de la Vision Numérique

Le clé de la puissance et du succès de la vision numérique réside dans le Réseau Neuron Convolutif (*Convolutional Neuron Network – CNN*) – dérivé de la recherche physiologique de Hubel et Wiesel [22] – dans laquelle, ils ont proposé un modèle pour expliquer comment les animaux perçoivent les informations via les systèmes optiques. En 1980, Fukushima [23] a proposé un modèle multicouches s'appelé *Neucognitron* – et devenu pilier des RNCs dans l'avenir. En 1998, le premier prototype de RNC – qui s'appelle LeNet-5 - a été offert par LeCun et al. [24]. Ce modèle a été réussi à classifier et reconnaître l'écriture manuscrite.

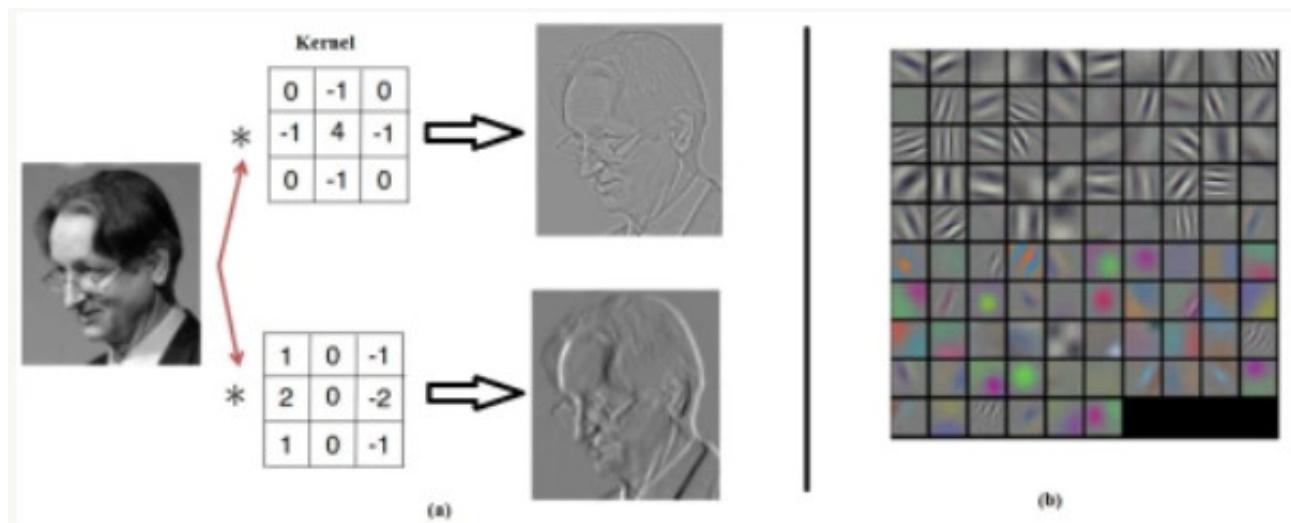


Modèle de perception – proposé par Hubel et Wisel [22]

Bien qu'il existe beaucoup de variantes différentes, tous les RNCs doivent être basés sur la conception de Convolution ou Fenêtre Glissante ou Noyau (*Kernel*) – mathématiquement, c'est une matrice qui peut glisser sur image en 3 dimensions pour extraire les caractéristiques – comme la couleur, le bord, la profondeur, le contraste,... de l'image – en appliquant une opération de convolution. Le résultat obtenu après avoir appliqué la convolution est la carte de caractéristiques (*feature map*) – contenant les caractéristiques pertinentes. En réalité, la convolution a la forme d'une matrice à 3 dimensions – représenté en 3 dimensions d'un image – longueur, largeur et profondeur – indiquant le nombre canaux de couleur (3 pour les images en Rouge-Vert-Bleu et 1 pour images en Noir et Blanc).

L'architecture de RNC comporte beaucoup de couches convolutives – dans lesquelles chaque couche sert à extraire une caractéristique spécifique correspondant à une carte de caractéristique spécifique.

En plus, pour optimiser la performance ainsi que réduire le nombre des paramètres de RNC et les informations redondantes, on va utiliser les couches de Pooling or SubSampling or DownSampling (*Pooling layer*, *SubSampling layer*, *DownSampling layer*) – en gardant la plus grande valeur pour chaque région convulsive.



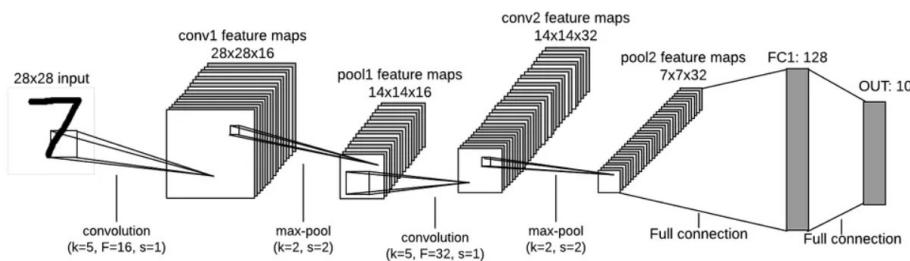
Exemple d'une convolution pour extraire le bord et le profondeur de l'image

(Source: <https://tiendv.wordpress.com/2016/12/25/convolutional-neural-networks/>)

Un RNC présentent les 3 avantages suivantes :

1. Le nombre de connexions entre neurones est bien inférieur à celui des réseaux traditionnels en se basant sur les champs récepteurs locaux (*Local Receptive Field*)
2. La performance d'extraction des caractéristiques est la meilleure parmi les réseaux actuels
3. Le nombre de caractéristiques pouvant être extraites est illimité

La deuxième partie d'un RNC est une ou des couches de classification en utilisant des connexions complètes (*fully-connected*) pour traiter les résultats convolutifs. Enfin, le résultat final est la classification des données entrées en se basant sur les labels prédéfinis.

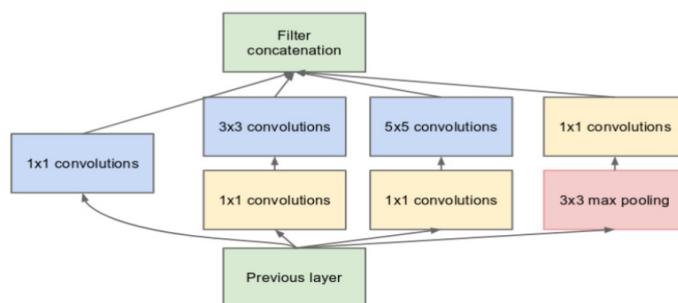


Architecture d'un RNC (Source : <https://www.easy-tensorflow.com/tf-tutorials/convolutional-neural-nets-cnns>)

Actuellement, il y a beaucoup de variantes de RNCs comme AlexNet [25], développé par Krizhevsky et a remporté le concours ImageNet 2012; VGG-16 [26] – proposé en 2014 par Simonyan et Zisserman, ResNet [27], GoogleLeNet-Inception [28],...

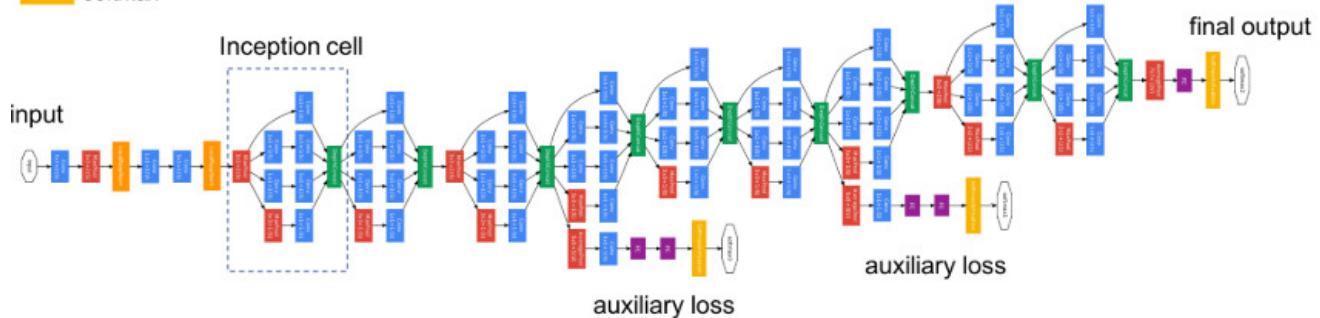
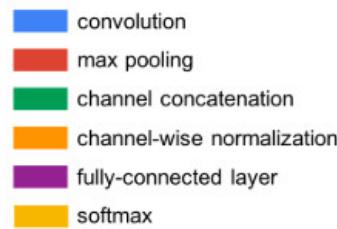
Inception

En se basant sur un film américain (Inception, 2010), l'idée principale d'Inception [28] est de minimiser le nombre de couches de réseau tout en assurant la profondeur nécessaire du réseau pour extraire des caractéristiques pertinentes. Au lieu d'empiler beaucoup de couches convolutives - augmenter la profondeur verticalement – ce qui peuvent causer le surajustement (ou sur-apprentissage), Inception va appliquer le mécanisme de parallélisation en augmentant la profondeur horizontalement – dans laquelle une les caractéristiques seront extraites en plusieurs fois, sur une même couche – chaque extraction correspondant à une branche. Ce modèle est composé de multiples “Inception Cells” – ayant la structure suivante.



Un “Inception Cell” – sert à extraire de multiples caractéristiques avec une même couche sans augmenter la profondeur du réseau

(Source : Inception V1 [28])



La structure générale un modèle Inception – composée par des “Inception Cell”

(Source : <https://towardsdatascience.com/a-simple-guide-to-the-various-versions-of-the-inception-network-7fc52b863202>)

Actuellement, il y a beaucoup de versions du modèle Inception, comme Inception v1 [28], Inception v2 [29], Inception v3 [29], Inception v4 [30] ; ainsi que la combinaison entre Inception et les autres modèles, comme Inception-ResNet [30],...

MobileNet

MobileNet [31], développé par Google, est vraiment une révolution parmi les modèles d'Apprentissage en Profondeur – en combinant les 2 objectifs les plus importants – la précision et la taille du modèle. Tandis que les modèles traditionnels – ayant une haute précision – doivent être déployés sur des grandes plateformes et puissantes (comme GPU,TPU), les modèles de MobileNet ont les tailles très petites – et donc peuvent être déployés sur les plateformes ayant une configuration minimale (comme Raspberry Pi). En plus, la précision de MobileNet est très élevée, et même supérieure à celle des modèles traditionnels. Le clé de ces modèles est à l'améliorant de la convolution traditionnelle – en utilisant *Depthwise Separable Convolution* [32] – afin de diminuer le nombre des paramètres.

Détection d'Objet

La détection d'Objet – le plus domaine le plus étudié de vision numérique – est une technique de localisation des objets et de détection d'un objet spécifique dans une image ou vidéo. Elle est utilisé dans beaucoup d'applications actuelles, comme la détection des visages, détection des véhicules pour des voitures autonomes,... Bien qu'il y ait beaucoup de modèles de détection d'objets, généralement, ils appartiennent aux certaines groupes suivantes :

1. Les modèles basés sur R-CNN (*Region with CNN features*) – développés par Girshick et al. [33] visent à identifier les régions caractéristiques – représentées par une boîte englobante (*bounding box*) basées sur RNCs. Il y a 3 sous-modèles principaux de ce groupe, y compris R-CNN, Fast-RCNN [34], et Faster-RCNN [35]. La précision de ces modèles, en général, est bonne, néanmoins, le temps d'exécution est trop long et donc pas adapté aux modèles en temps réel.

2. Les modèles de YOLO (*You Only Look Once*) – développé par Redmon et al. [36] – avec les différentes versions – qui génèrent les résultats plus rapides que R-CNN et que l'on peut appliquer pour automatiquement détecter les objets dans les vidéos. Cependant, la précision de ces modèles n'est pas aussi efficace que celle des modèles basés sur de R-CNN.
3. Les modèles de SSD [37] – pouvant résoudre les points faibles de R-CNN et YOLO – sont développés sur la base d'une modèle standard comme VGG-16, Inception, ResNet,... pour extraire les caractéristiques et puis supprimer tous les connexions pleines ainsi que personnaliser certaines architectures de détection d'objet. Ce stage va exploiter seulement les modèles SSD basés sur *Inception* et *Mobilenet*.

Chapitre 3 : Résultats Expérimentaux

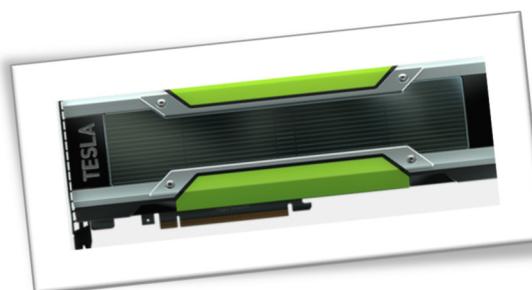
Préparation de l'environnement

Plateforme : Google Colaboratory

Ce projet est implémenté et puis déployé sur le plateforme Google Colaboratory pour profiter pleinement la puissance du traitement des GPU (Graphics Processing Unit) et TPU (Tensor Processing Unit), utilisée uniquement pour des applications d'apprentissage automatique de Google. La combinaison entre GPU, TPU et Tensorflow ont fait la puissance de Google Colaboratory en optimisant le temps d'entraînement des modèles, en particulier des modèles d'apprentissage en profondeur.

Le GPU utilisé pour ce projet est NVIDIA®Tesla K80 avec la puissance suivante (source : <https://www.nvidia.com/en-gb/data-center/tesla-k80/>)

- 4992 cœurs CUDA avec conception bi-GPU
- 2,91 TFlops de performances en double précision
- 8,73 TFlops de performances en simple précision
- 24 Go de mémoire GDDR5
- 480 Go/s de bande passante globale



Langage de programmation : Python

Le langage de programmation utilisé pour ce projet est Python avec la version 3.6 minimale. Les 5 raisons pour laquelle Python a été choisi (pas les autres langages) pour ce projet sont :

- La syntaxe est simple et facile à lire. En outre, par rapport avec Java ou C, un programme écrit en Python est plus court
- Python possède de nombreuses librairies et frameworks gratuits et disponibles, particulièrement liés au traitement des données et l'apprentissage automatique. Certaines exemples sont Keras, Tensorflow, Pytorch et Scikit-Learn pour Apprentissage Automatique, Numpy pour analyse des données, Matplotlib et Seaborn pour visualisation des données,...
- Python est indépendant de la plateforme. Bien que ce projet soit développé sur la plateforme Linux, il est compatible et fonctionne bien sur d'autres plateformes comme MacOs, Windows,...
- La communauté de Python est énorme, donc la recherche de support est plus facile et favorable. De plus en plus de grandes entreprises (Google, Amazon, Facebook,...) utilisent Python pour leurs plateformes.



Python is powerful... and fast; plays well with others; runs everywhere; is friendly & easy to learn; is Open.

(Slogan de Python)

Framework : Tensorflow

Afin de construire des modèles d'apprentissage automatique pour ce projet, le candidat le plus idéal est Tensorflow. Tensorflow est une framework open-source développé et supporté par Google. Tensorflow a été développé à l'origine par l'équipe Google Brain dans le but de rechercher et produire les produits internes de Google, et puis publiée sous la licence open-source Apache 2.0 en novembre 2015. La version actuelle de Tensorflow est 2.0, publié en 2020. Tensorflow supporte 2 plateformes principales, l'un pour le CPU et l'autre pour le GPU (performance supérieures). Ce projet utilise Tensorflow pour GPU avec la version spécifique 1.14.0.



Les autres outils, librairies utilisés

En plus des 3 piliers ci-dessus, ce projet également exploite certaines outils et librairies, tels que :

- Numpy (avec la version spécifique 1.15.4) pour les calculs arithmétiques avancés
- Matplotlib pour la visualisation des données
- Pandas pour le traitement des données



Préparation des données

Format de données

L'ensemble des données brutes de ce projet, fournit par le L3I, est en sous forme d'image (avec extension de JPEG, PNG ou BMP), collectés manuellement au fur et à mesure à partir des 3 pontons 51, 55 et 57 du Porte de Plaisance de La Rochelle, en Mars 2020. Toutes les images sont en forme Rouge-Vert-Bleu (*Red-Green-Blue*).



Images des espèces marines au Porte de Plaisance, La Rochelle

Format des labels

Ensuite, cet ensemble d'images sera étiqueté manuellement avec l'aide d'un spécialiste en biologie pour déterminer la présence de chaque espèces, comme les moules, les ascidies, les pétoncles, les huîtres, les bryozoaires, Chaque espèce correspond à une annotation spécifique, caractérisé par un nombre entier. Les espèces ainsi que les annotations prédéfinies dans ce stage sont présentées dans la table suivante :

Identification du label	Nom d'espèce
1	Ascidie
2	Ascidie Bouche
3	Pétoncle
4	Moule
5	Huître
6	Bryozoaire
7	Bryozoaire Dense

Pour faire les annotations, ce stage utilise l'outil de labellisation interne développé par le L3I. Un exemple d'une image labélisée et des labels correspondants est donné ci-dessous.

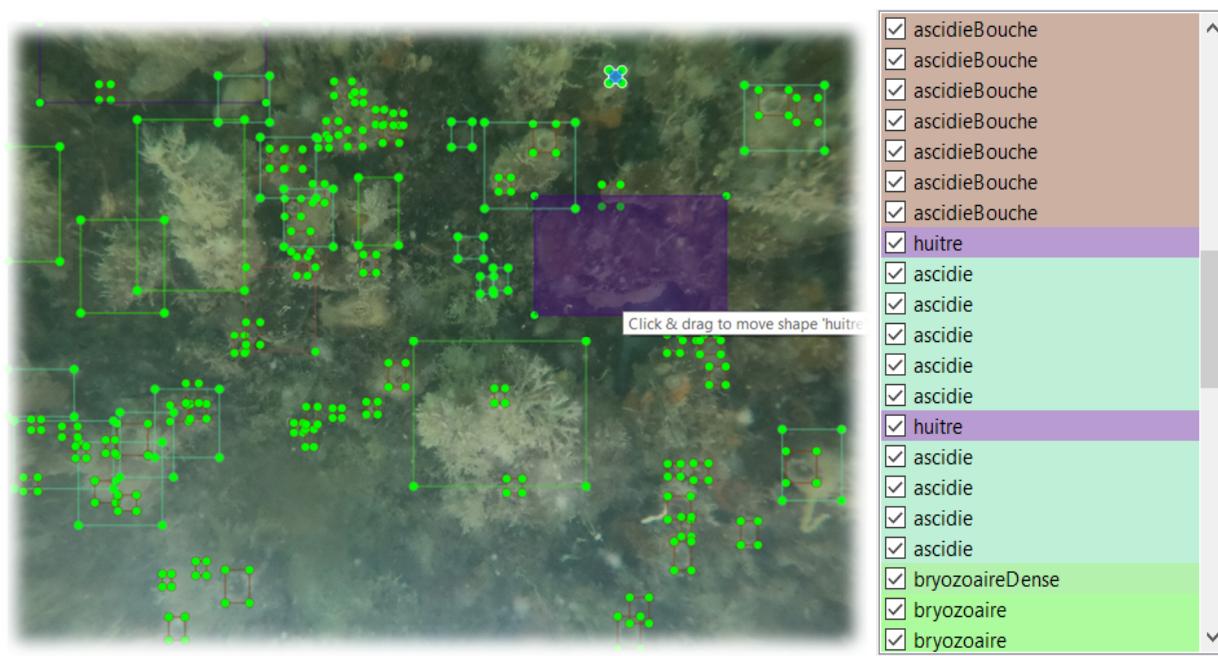
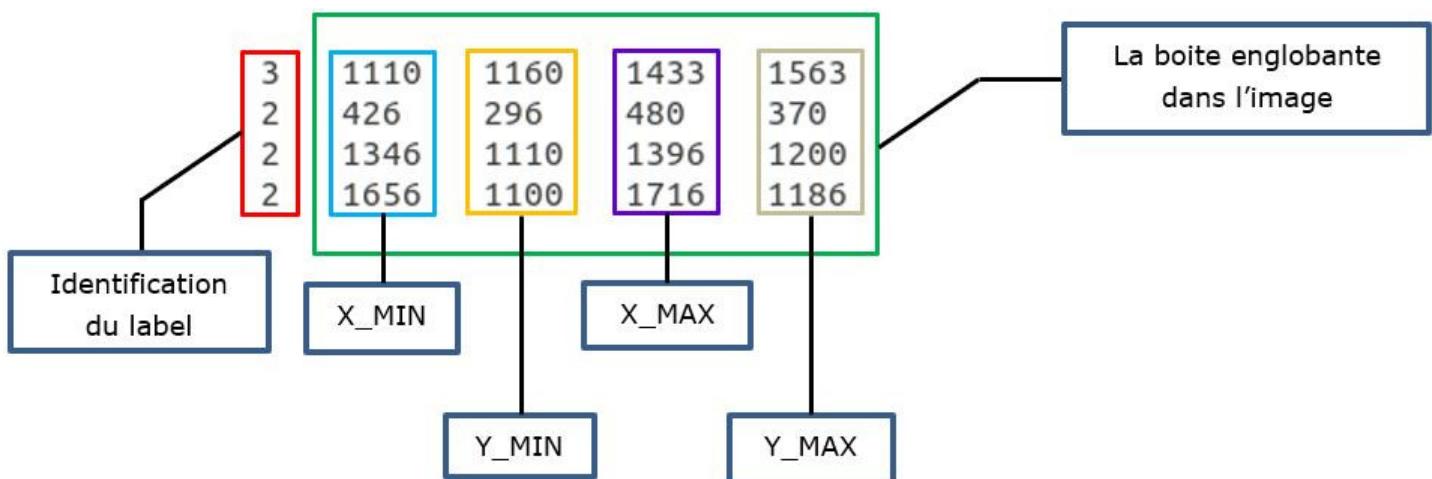


Image labellisé et la liste des labels correspondants (chaque label a un nom et couleur unique)

Après avoir fini la labellisation, la liste des annotations est sauvegardée sous la forme d'un fichier Texte (avec l'extension de .TXT). Chaque ligne de ce fichier indique la présence d'une espèce marine dans l'image correspondante. Chaque ligne a cinq colonnes, et la signification de chaque colonne est :

1. La première colonne indique l'identification du label sous forme d'un nombre entier. La correspondance entre cette identification et le nom de l'espèce a été présenté dans la table ci-dessus.
Les quatre colonnes restantes indiquent la boite englobante dans l'image, dans laquelle:
2. La deuxième colonne indique la coordonnée extrêmement la plus gauche de X, notée X_MIN
3. La troisième colonne indique la coordonnée extrêmement la plus gauche de Y, notée Y_MIN
4. La quatrième colonne indique la coordonnée extrêmement la plus droite de X, notée X_MAX
5. La cinquième colonne indique la coordonnée extrêmement la plus droite de Y, notée Y_MAX



La structure des labels bruts (en format de texte)

Prétraitement des données brutes

Les données brutes originales ne peuvent pas être directement utilisées pour l'entraînement du réseau. Elles doivent être prétraitées pour éliminer les anomalies des données (les données cachées, les répertoires, les données avec format incorrect,...), équilibrer le nombre des images et labels (s'il existe). Cette étape vise à générer un propre jeux de données pour l'entraînement des modèles.

Sous-étape 1 : Éliminer les données cachées et les répertoires (« Nettoyage de données »)

L'apparition de n'importe quelle donnée cachée, répertoire ou structure arborescence dans le jeux de donnée ruinerait le processus d'entraînement en premier temps. Cette étape vise à garantir l'homogénéité du jeux de données en supprimant les problèmes ci-dessus. L'API de cette sous-étape que j'ai développé est écrit en Python, disponible [ici](#).

Sous-étape 2 : Éliminer les données avec format incorrect (« Identification du format »)

Comme mentionné ci-dessus, le format des données de ce projet est l'image et le format des labels est le texte (avec l'extension de TXT). L'apparition de toute autre formats ne correspondant pas aux formats ci-dessus va détruire le modèle d'entraînement. Par exemple, l'apparition d'un donnée de DOCX, XLSX ou LOG dans les labels est inconvenable. De même, l'occurrence des données non-image (les textes, les audios,...) dans les images est également invalide. Cette sous-étape, conséquemment, assurera l'uniformité de format de fichier, à la fois des images et des labels, en éliminant tous les données invalides. Par défaut, ce projet utiliser le format de Joint Photographic Experts Group (JPEG) pour les images. L'API de cette sous-étape que j'ai déjà développé est écrit en Python, disponible [ici](#).



Sous-étape 3 : Équilibrer le nombre de données et labels (« Synchronisation de données »)

La relation entre l'image et le fichier de label est une relation 1-1, c'est-à-dire, une image a seulement un fichier de label dont le nom est le même, sans exception. Néanmoins, en raison de certaines raisons objectives (erreur logicielle, erreur système,...) ou subjectives, il peut y avoir une différence entre le nombre d'images et de labels. Les données redondantes (images ou labels) sont appelées « données orphelines » (*orphan data*). Bien que ce problème soit peu probable, il peut ruiner l'entraînement du modèle en premier lieu. Donc, il est nécessaire à éliminer les données orphelines pour assurer l'équilibrage des données. L'API de cette sous-étape est écrit par Python, disponible [ici](#).

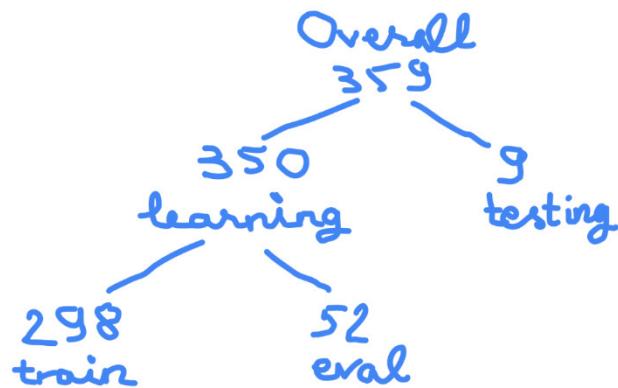


Partitionnement des données

Les données, y compris images et labels, après avoir prétraitées, doivent être partitionnées en deux parties (chaque partie correspondant à un jeu de données), l'un pour l'apprentissage (*learning dataset*), l'autre pour le test (*testing dataset*). Le jeu de données d'entraînement est encore divisé en 2 sous-catégories (sous-jeux de données) : l'un pour l'entraînement (*training dataset*), l'autre pour l'évaluation (*validation dataset*). Ces deux jeux de données sont divisés selon une proportion fixée. Normalement, le nombre de données de l'entraînement sera au moins 3 fois celui de l'évaluation. Après avoir essayé les différentes valeurs, ce projet a utilisé le taux : 85% pour l'entraînement et 15% pour l'évaluation. L'API de cette sous-étape que j'ai déjà développé est écrit en Python, disponible [ici](#).



Avec 359 données globales, ce projet utilise 350 données pour l'apprentissage et 9 données pour le test. Ensuite, les données d'apprentissage sont divisés en 298 (correspondant à 85%) données pour l'entraînement et 52 (correspondant à 15%) données pour l'évaluation.



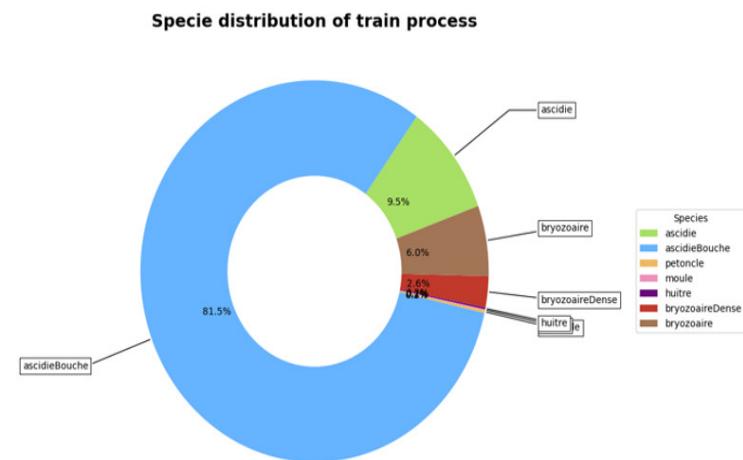
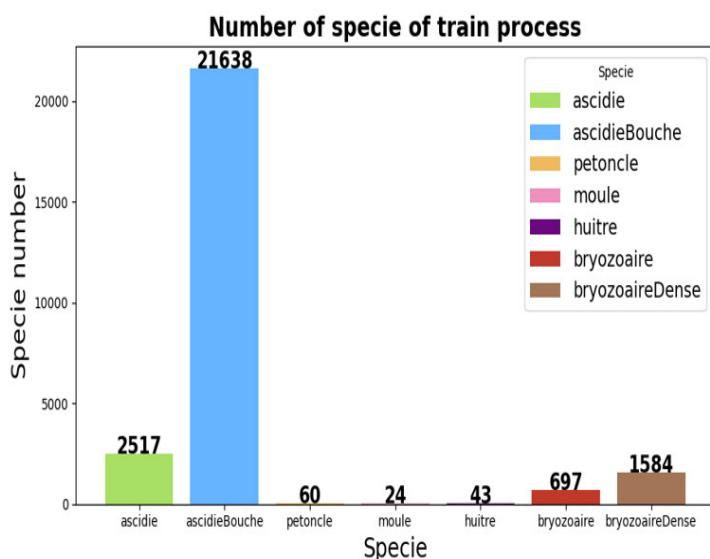
Organisation des données

Distribution des espèces marines dans les jeux de données

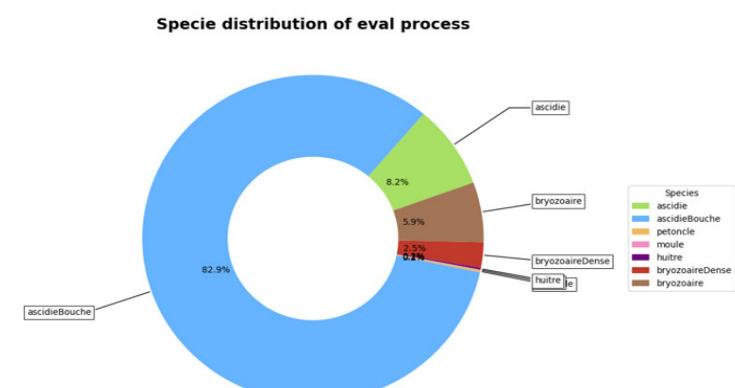
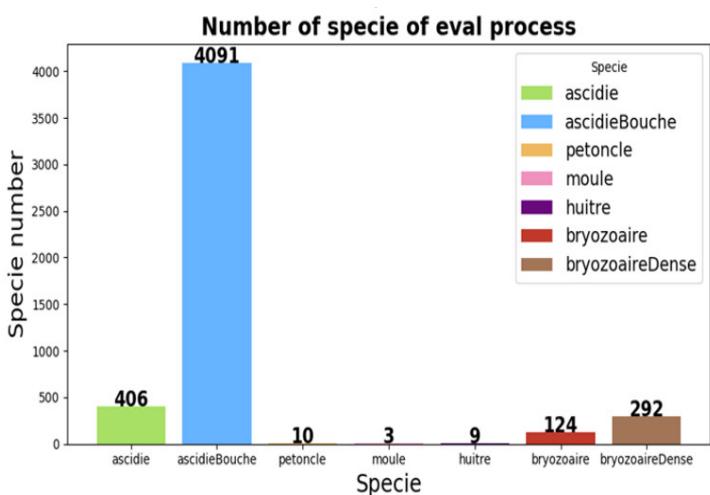
La visualisation des données joue un rôle utile pour soutenir l'analyse des données. Il donne un aperçu de la distribution des données, et de leur qualité des données. Grâce à cette visualisation, nous pouvons évaluer et prédire quelque peu la qualité d'entraînement du modèle, sans réellement le mettre en œuvre. Mes API sont disponibles [ici](#).

Après avoir extrait les informations à partir de 350 données d'apprentissage, la distribution de chaque espèce est visualisée sous forme diagramme à bandes et camembert. Voici quelques brefs commentaires sur la distribution ainsi que la qualité des données :

1. La distribution des espèces marines sur les ponts pontons 51, 55 et 57 est très inégale. Le nombre d'espèces avec la nomenclature scientifique *Ascidiaecea* est de plus de 90% du nombre total d'espèces, dont l'occurrence des *ascidies bouches* est la plus abondante. Au contraire, l'apparition des *moules*, *huîtres* et *pétoncles* est négligeable (le taux de distribution chacune de ces espèces est inférieur à 0,2%). En outre, pour confirmer l'exactitude des données, une autre statistique a été réalisée indépendamment par un collaborateur biologiste travaillant au L3I, dans laquelle il a déterminé l'occurrence de chaque espèce sans utiliser les labels prédéfinis. Les résultats de ces deux processus sont conformes, dans lesquels, l'ascidie bouche est l'espèce la plus présente sous les pontons.
2. En raison de cette inégalité, il est facile de prédire que les résultats d'entraînement seront plus enclins aux espèces d'*ascidie bouche* que les autres.



La distribution des espèces marines dans les données d'entraînement



La distribution des espèces marines dans les données d'évaluation

Conversion des annotations de texte en PASCAL VOC

PASCAL [38] (Pattern Analysis, Statistical Modelling and Computational Learning) VOC est un standard pour l'organisation et présentation des données, dans laquelle :

1. PASCAL VOC sauvegarde les labels sous forme XML. Chaque image correspond à un fichier XML
2. La structure arborescente d'un fichier XML de PASCAL VOC respecte les règles suivantes :
 - a. L'élément racine est toujours “*annotation*”
 - b. Le sous-élément “*folder*” indique le répertoire qui contient l'image originale
 - c. Le sous-élément “*path*” indique la référence absolue de l'image originale
 - d. Le sous-élément facultatif “*source*” avec le seul sous-sous-élément “*database*” indique quelle est la base donnée depuis laquelle l'image originale est sauvegardée
 - e. Le sous-élément “*size*” indique la taille en trois dimensions (largeur, hauteur et profondeur) de l'image originale. La profondeur est le nombre de canaux de couleur (1 pour image noir et blanc – *grayscale channel* ou 3 pour image couleur – *RGB channel*). Largeur, longueur et profondeur sont au fur et à mesure 3 sous-élément de l'élément “*size*” (*width*, *height*, *depth*)
 - f. Le sous-élément le plus important, “*object*” indique l'occurrence de chaque objet dans l'image originale. Le nombre d'objets dans l'image originale est également le nombre d'occurrence de l'élément “*objet*” dans le fichier XML. Chaque objet est présenté par les 5 informations suivantes (chaque information correspondant à un sous-élément) :
 - i. “*name*” indique le nom du label
 - ii. “*pose*” a 2 valeurs possibles “*unspecified*” et “*specified*”. Il est facultatif
 - iii. “*truncated*” indique la visibilité (partielle ou pleine) d'un objet dans l'image. Il a 2 valeurs possibles “*0*” (visibilité partielle) ou “*1*” (visibilité pleine). Il est aussi facultatif
 - iv. “*difficult*” indique la reconnaissance (difficile ou facile) d'un objet dans l'image. Il a aussi 2 valeurs possibles “*0*” (reconnaissance facile) ou “*1*” (reconnaissance difficile). Il est optionnel
 - v. “*bndbox*” (*bounding box*) indique la boîte englobante pour chaque objet. Cette boîte est définie par 4 paramètres : X_MIN, Y_MIN, X_MAX, Y_MAX



```
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  <folder>/home/ttduy/subia-data/dataTraining/images/train</folder>
  <filename>51_2A.jpg</filename>
  <path>/home/ttduy/subia-data/dataTraining/images/train/51_2A.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>4000</width>
    <height>3000</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>ascidieBouche</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>236</xmin>
      <ymin>450</ymin>
      <xmax>410</xmax>
      <ymax>613</ymax>
    </bndbox>
  </object>
  <object>
    <name>ascidieBouche</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>96</xmin>
      <ymin>786</ymin>
      <xmax>213</xmax>
      <ymax>920</ymax>
    </bndbox>
  </object>
</annotation>
```

L'image original et l'annotation correspondant en format de PASCAL VOC

Les labels bruts sous forme de texte seront convertis au standard PASCAL VOC en forme XML en utilisant les API fournis que j'ai déjà développé dans le script Python disponible [ici](#).

Générer les structures de données particulières (TFRecord, Label Map)

Dans un premier temps, l'ensemble des labels discrets ci-dessus sera centralisé dans un seul fichier Comma-Separated Values (CSV), dont la structure est composé des 8 colonnes suivantes :

1. “filename” indique le nom d'image (avec le format suivi)
2. “width”, “height” indiquent respectivement la largeur et hauteur de l'image
3. “class” indique le nom du label
4. “xmin”, “ymin”, “xmax”, “ymax” sont dans l'ordre les valeurs de X_MIN, Y_MIX, X_MAX, Y_MAX

filename	width	height	Class	xmin	ymin	xmax	ymax
51_18C.jpg	4000	3000	ascidieBouche	940	396	1000	463
51_18C.jpg	4000	3000	ascidieBouche	1010	356	1076	423
55-2C.jpg	4000	3000	moule	1996	920	2570	1436
55-2C.jpg	4000	3000	bryozoaire	1813	2216	2470	2906
57-31B.jpg	4000	3000	huitre	10	1926	900	2950
55-8C.jpg	4000	3000	petoncle	2190	400	2746	993

La structure de CSV

Cette étape va générer 2 fichiers CSV, l'un pour l'entraînement et l'autre pour l'évaluation. Le script Python de cette conversion, fournie par Google, est disponible sur [ici](#).

Protocol Buffer : une structure donnée particulière

L'étape suivante présente une structure de donnée particulière seulement réservée à Tensorflow, c'est *TFRecord* [39]. TFRecord est un format de donnée visant à stocker les données sous forme d'enregistrements de séquence binaire. TFRecord est particulièrement utile avec des données volumineuses, car il va accélérer le temps d'interaction avec les données (chargement, copiage, écriture et modification) ainsi que réduire la taille du stockage. Donc, la performance de l'entraînement du réseau sera optimisée significativement.



TFRecord

TFRecord est basé sur la base de Protocol Buffer [40] (*ProtoBuf*), une technologie développée par Google. Similaire à JSON ou XML, ProtoBuf sert à l'échange des données entre les applications en utilisant le format PROTO. Par rapport à XML ou JSON, la performance de ProtoBuf est meilleure en raison de l'utilisation de données binaires. Les deux comparaisons ci-dessous prouvent la performance de Protobuf par rapport à XML et JSON [41] :



1. La taille des données de ProtoBuf est très légère, au moins 3 fois inférieure
2. La puissance de traitement de ProtoBuf est au moins 20 fois plus rapide

Actuellement, ProtoBuf supporte de multiples langages, dont Java, Python, Golang, C et C++, et est compatible avec différents environnements. L'installation de ProtoBuf pour l'environnement Linux se fait via le package *Protobuf-Compiler* (avec la commande : `apt-get install protobuf-compiler`).

Les deux fichiers CSV ci-dessus seront convertis au fur et à mesure en TFRecord via un pipeline entre le CSV et les images correspondants en utilisant les API, fournis par Google, disponibles [ici](#).

Label Map : référence des labels

La carte des labels (*label map*) est utilisée pour que le modèle reconnaissasse chaque label dans TFRecord ci-dessus. La carte des labels respecte le mappage 1-1 entre ID et le nom de chaque label (chaque label possède un seul ID et nom). La structure de la carte des labels est similaire à un dictionnaire en Python en utilisant la structure Clé-Valeur (*Key-Value*). La carte des labels du projet SUBIA est sauvegardée en forme PBTXT (*Protocol Buffer Text*) et disponible [ici](#).

```
item {  
    id: 1  
    name: 'ascidie'  
}  
item {  
    id: 2  
    name: 'ascidieBouche'  
}
```

Exemple d'une carte de label

Customisation des paramètres et Entrainement des modèles

Common Objects in COntext (COCO)

COCO [42] est un grand jeu de données, développé et mis à disposition par Microsoft, Facebook, Mighty AI et CVDF, réservé à la vision numérique. COCO est composé 330,000 images et de plus de 220,000 labels, réservé à de nombreux problèmes différents, y compris : la Détection d'Objet, la Détection de points-clés (*Keypoint Detection*), la Segmentation d'Objet (*Stuff Segmentation*), la Segmentation Panoptique (*Panoptic Segmentation*), la Contextualisation d'Image (*Image Captionning*). Les APIs développés par COCO sont utilisées pour de nombreux problèmes et compatibles avec des plateformes différentes, comme Python (*Python API*), Lua (*Lua API*) et Matlab (*Matlab API*). Ce projet utilise les APIs spécifiques à Python avec l'outil `Pycocotools`, qui peut être installé via `pip3` (ou `pip` avec Python 2).

En outre, ce projet réutilise également des modèles déjà entraînés (*pretrained models*) par Google basés sur la base de COCO. Pour ces modèles, le nombre de couches réseau et la valeur des paramètres pertinentes ont été optimisés par Google basé sur l'expérience et la méthode essai-erreur.

Oxford-IIIT Pet

Similaire à COCO, Oxford-IIIT Pet [43], développé par Oxford Université (Grand-Bretagne), est un grand jeux de données, réservé à la vision numérique. Oxford-IIIT Pet compose 37 catégories différentes avec 7349 données incluant images et labels. Ce projet réutilise certains modèles pré-entraînés par Google basés sur la base de Oxford-IIIT Pet. Comme les modèles basés sur COCO ci-dessus, les paramètres sélectionnés dans les modèles de Oxford-IIIT Pet sont considérés les meilleurs parce qu'ils sont sélectionnés par la méthode essai-erreur.



Choisir les modèles disponibles

Google fournit gratuitement 61 modèles basés pour le problème de la Détection Object. Tous ces modèles appartiennent à la famille de Faster-RCNN et peuvent être divisés selon les 4 groupes suivantes :

A. SSD avec 2 sous-catégories

- a. Mobilenet
- b. Inception

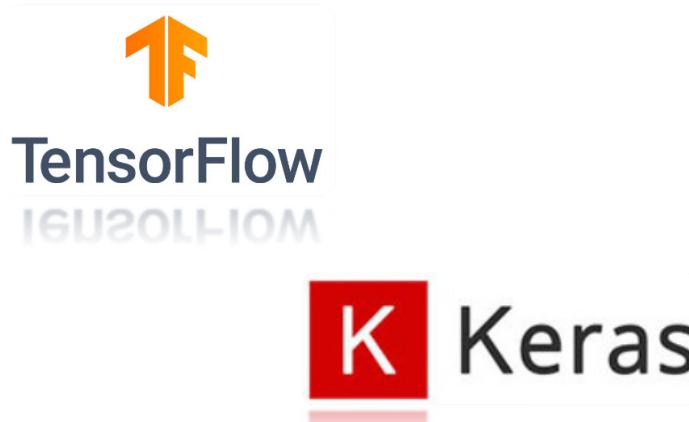
B. Faster-RCNN avec 3 sous-catégories

- a. Inception
- b. NAS [44]
- c. Resnet

C. Mask-RCNN [45] avec 2 sous-catégories

- a. Inception
- b. Resnet

D. RFCN-Resnet [46]



En outre, pour chacun des groupes ci-dessus, Google également fourni les modèles configurés sur les deux bases COCO (avec le suffix “_coco” comme “ssd_mobilenet_v2_coco”) et Oxford-IIIT Pet (avec le suffix “_pets” comme “ssd_inception_v2_pets”).

Tout les modèles ci-dessus sont implémentés sur Tensorflow et Keras, un framework intégré en Tensorflow et développé par François Chollet, pour les applications de Intelligence Artificielle.

Actuellement, ce projet exploite seulement les modèles basés sur SSD, y compris. Le résumé de ces modèles est présentée dans la table suivante.

ID	Catégorie	Nom du modèle	Profondeur du réseau	Taille de batch	Boîte d'ancre - Profondeur
1	Mobilenet	embedded_ssd_mobilenet_v1_coco	16	32	5
2		ssdlite_mobilenet_v1_coco	16	24	6
3		ssdlite_mobilenet_v2_coco	16	24	6
4		ssd_mobilenet_v1_coco	16	24	6
5		ssd_mobilenet_v1_focal_loss_pets	16	24	6
6		ssd_mobilenet_v1_pets	16	24	6
7		ssd_mobilenet_v2_coco	16	24	6
8		ssd_mobilenet_v2_focal_loss_pets_inference	16	24	6
9	Inception	ssd_inception_v2_coco	16	24	6
10		ssd_inception_v2_pets	16	24	6
11		ssd_inception_v3_pets	16	24	6

Tous les modèles ci-dessus seront entraînés avec 3,000 étapes (*epoch, step*) et les autres valeurs comme le taux d'apprentissage (*learning rate*), la fonction d'activation (*activation function*), la fonction de perte (*loss function*)... seront inchangés. Le temps d'entraînement des modèles est de 164 minutes à 879 minutes en utilisant TPU et GPU, ça dépend de certaines facteurs comme l'architecture de modèle, de la stratégie d'initialisation des paramètres

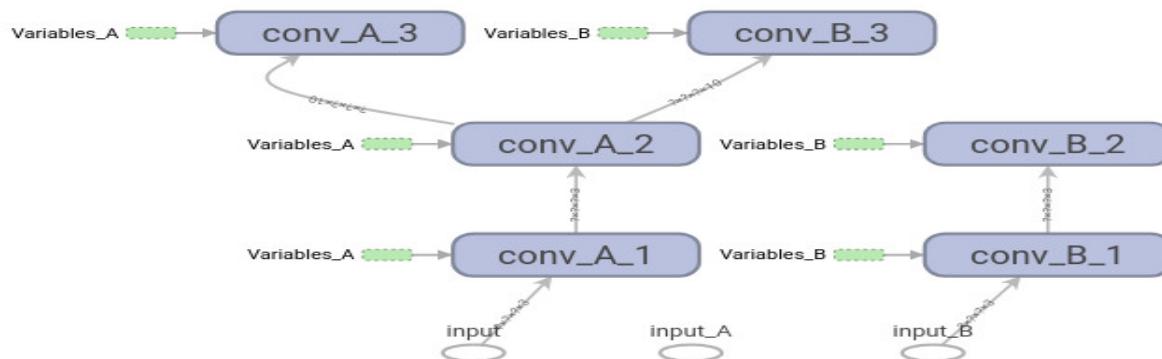
(comme Xavier [47]), la stratégie pour éviter le sur-apprentissage (comme Regularization L2 [48]), du mécanisme d'optimisation (comme Adam [49]),.... L'entraînement avec l'utilisation de CPU prendrait beaucoup de temps (en quelques jours, voire quelques semaines).

L'entraînement d'un modèle sera fait en configurant les paramètres du modèle (le fichier de configuration avec le format de CONFIG), puis en reliant cette configuration à la carte des labels via le pipeline. Les fichiers de configuration du projet SUBIA sont disponibles [ici](#).

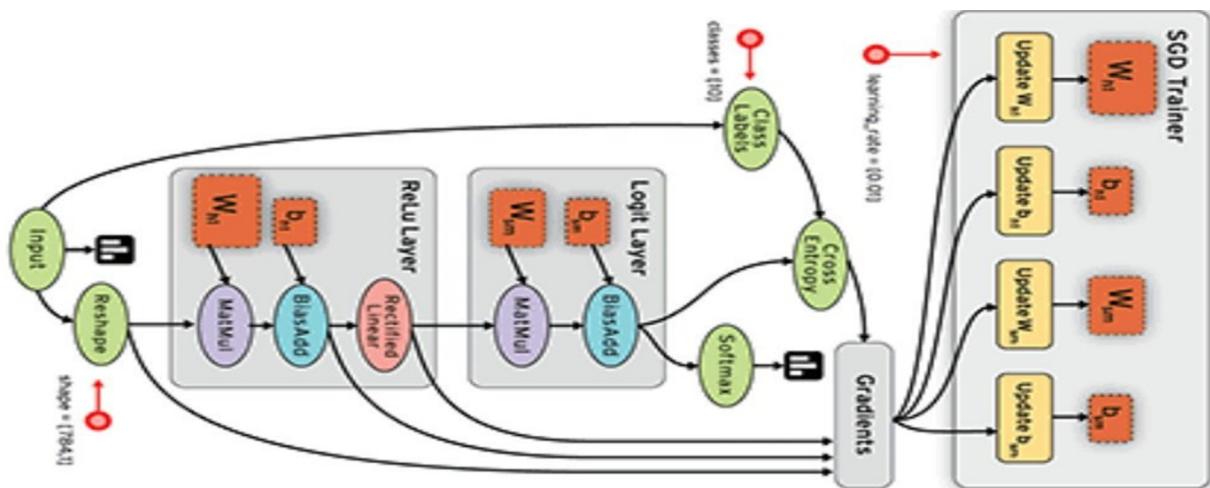
Méta-Graphe (Meta Graph)

Tensorflow, basé sur 2 concepts – Tensor et Flow, dans laquelle Tensor décrit la structure des données du modèle tandis que Flow décrit la structure du modèle. Autrement dit, Tensorflow est un outil fournissant les APIs nécessaires pour construire les flux tensoriels ainsi que manipuler ces flux. La colonne vertébrale de Tensorflow est les graphes de calcul (*Computational Graph*) – un ensemble d'opérations décrivant le flux tensoriel organisé en forme d'un graphe orienté, dans lequel :

- Les sommets représentent les variables, les opérations
- Les arêtes ou les arcs indiquent les flux de calcul entre les sommets



Exemple d'un graph de calcul en Tensorflow (Source : [Stackoverflow](#))



Exemple d'un graph de calcul en Tensorflow (Source : [DZone](#))

L'entraînement d'un modèle va générer un graphe correspondant (en format META) et le graphe de chaque modèle ci-dessus est disponible [ici](#).

Point de Contrôle (Checkpoint)



TensorFlow
ISUOLI W

Si le graphe de calcul détermine l'architecture et l'ensemble des opérations d'un modèle, le point de contrôle enregistre exactement tous les valeurs des paramètres d'un modèle. Un point de contrôle est représenté sous la forme de données binaires en format CKPT (*Check Point*). Tensorflow a un mécanisme pour enregistrer automatiquement les points de contrôles après une certaine période de temps (comme 60 minutes, 120 minutes,...). Certains des avantages de l'utilisation des points de contrôles sont :

1. Indépendants avec les codes sources et le graphe
2. Facilement à déboguer avec le mécanisme d'enregistrement périodique
3. Peuvent être déployés n'importe quand sans avoir besoin d'entrainer le réseau

Les points de contrôle des modèles ci-dessus sont disponibles [ici](#).

Évaluation de la performance et de la qualité des modèles

Pour évaluer la performance et la qualité des modèles, je vais utiliser les 3 critères suivantes :

1. La valeur finale de perte (table 1) qui représente la différence entre la valeur prédictive et la valeur réelle. Cette valeur reflète la qualité de l'entraînement du réseau. Plus cette valeur est petite, plus le modèle est efficace.
2. La stabilité des valeurs de perte (table 1) indique la fluctuation de modèle pendant l'entraînement, représentée par 2 niveau – diminuer/augmenter, fluctuer, y compris 4 valeurs de perte :
 - a. Classification (*Classification*)
 - b. Localisation (*Localization*)
 - c. Régularisation (*Regularization*)
 - d. Totale

Modèle	Valeur finale de perte	Valeur souhaitée	Stabilité de la fonction de perte			
			Classification	Localisation	Régularisation	Totale
embedded(ssd_mobilenet_v1_coco)	8.96	Moins de 0.1	Dim. Stable	Dim. Stable	Augm. Stable	Dim. Stable
ssdlite_mobilenet_v1_coco	8.7		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssdlite_mobilenet_v2_coco	8.79		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_mobilenet_v1_coco	9.02		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_mobilenet_v1_focal_loss_pets	4.68		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_mobilenet_v1_pets	9.01		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_mobilenet_v2_coco	8.63		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_mobilenet_v2_focal_loss_pets_inference	4.96		Dim. Stable	Dim. Stable	Fluct	Dim. Stable
ssd_inception_v2_coco	8.62		Fluct. Forte	Fluct. Forte	Dim. Stable	Fluct. Forte
ssd_inception_v2_pets	8.05		Fluct. Forte	Fluct. Forte	Dim. Stable	Fluct. Forte
ssd_inception_v3_pets	11.56		Fluct. Forte	Fluct. Forte	Dim. Stable	Fluct. Forte

Table 1 - Les valeurs de perte et l'évolution de la fonction perte

Les abréviations dans la table :

- | | |
|-------------------------------------|---------------------------------------|
| - Dim. Stable : Diminuer stablement | - Augm. Stable : Augmenter stablement |
| - Fluct : Fluctuer normalement | - Fluct. Forte : Fluctuer fortement |

3. Les critères de Précision et Rappel:

- a. Précisions de la boîte de détection (*DetectionBox_Precision*) avec la métrique mAP (*Mean Average Precision*) (table 2)
- b. Rappels de la boîte de détection (*DetectionBox_Recall*) avec la métrique AR (*Average Recall*) (table 3)

Précision/Valeur Prédictive Positive (*Positive Predictive Value – PPV*) et Rappel/Sensibilité/Taux de Vrai Positif (*True Positive Rate - TPR*) sont les 2 métriques basiques pour juger la performance d'un modèle de classification, pour lesquelles:

- La Précision indique l'exactitude de la prédiction, calculée par le ratio des échantillons corrects par rapport au total des échantillons classés dans une classe, dont la formule est $Precision (PPV) = \frac{TP}{TP + FP}$
- Le Rappel détermine le ratio des échantillons corrects par rapport au total des échantillons d'une classe, dont la formule est $Recall (TPR) = \frac{TP}{TP + FN}$
(avec TP, FP, FN indiquent dans l'ordre Vrai Positif, Faux Positif et Faux Négatif)
- La valeur de la Précision et du Rappel est toujours entre 0 et 1, et plus la valeur de Précision et Rappel est élevée, plus la performance du modèle est bonne car la qualité de classification et détection plus élevé

Le sigle AP, toujours entre 0 et 1, indique la relation entre les 2 métriques Précision et Rappel – présentée par la courbe de Précision-Rappel (*Precision-Recall curve*). L'objectif d'AP est évaluer la performance de détection et seulement appliqué pour une classe/catégorie, et mAP est la moyenne est AP – calculée sur toutes les classes/catégories. En outre, AR signifie la moyenne de Rappel – appliquée pour une image.

Basé sur mAP, COCO fournissent la métrique mAP@.50IOU et mAP@.75IOU – indiquant dans l'ordre la valeur de mAP appliquée pour 50% et 75% de la taille de IOU (*IOU : Intersection Overlap Union* mesure le mappage entre la boîte englobante pré définie et celle détectée). En plus, les valeurs de mAP small, mAP medium et mAP large sont respectivement mAP appliquées pour la taille de la boîte de détection, dans laquelle :

- “small” a une taille de moins de 32 x 32 pixels
- “medium” a une taille entre 32 x 32 pixels et 96 x 96 pixels
- “large” a une taille supérieure à 96 x 96 pixels

De plus, COCO définit également la métrique AR@1, AR @10 et AR@100 – qui respectivement indiquent la valeur de AP sur 1 fois, 10 fois et 100 fois de détection. Outrement, AR@100 avec 3 les paramètres “small”, “medium” et “large” sont pareils comme mAP small, mAP medium et mAP large ci-dessus.

Average Precision (AP):

AP	% AP at OKS=.50:.05:.95 (primary challenge metric)
AP ^{OKS=.50}	% AP at OKS=.50 (loose metric)
AP ^{OKS=.75}	% AP at OKS=.75 (strict metric)

AP Across Scales:

AP ^{medium}	% AP for medium objects: $32^2 < \text{area} < 96^2$
AP ^{large}	% AP for large objects: $\text{area} > 96^2$

Résumé des valeurs de mAP (Source : <http://cocodataset.org/#detection-leaderboard>)

Average Recall (AR):

AR % AR at OKS=.50:.05:.95

AR^{OKS=.50} % AR at OKS=.50

AR^{OKS=.75} % AR at OKS=.75

AR Across Scales:

AR^{medium} % AR for medium objects: $32^2 < \text{area} < 96^2$

AR^{large} % AR for large objects: $\text{area} > 96^2$

Résumé des valeurs de AR (Source : <http://cocodataset.org/#detection-leaderboard>)

Modèle	mAP					
	mAP	mAP small	mAP medium	mAP large	mAP@.50IOU	mAP@.75IOU
embedded_ssd_mobilenet_v1_coco	3.41E-04	3.41E-04	0	0	1.14E-03	5.69E-05
ssdlite_mobilenet_v1_coco	1.40E-04	1.40E-04	0	0	5.62E-04	9.57E-05
ssdlite_mobilenet_v2_coco	6.81E-05	6.81E-05	0	0	2.69E-05	1.69E-06
ssd_mobilenet_v1_coco	9.03E-05	9.03E-05	0	0	3.86E-04	2.08E-05
ssd_mobilenet_v1_focal_loss_pets	2.55E-05	2.58E-05	0	0	8.86E-05	0
ssd_mobilenet_v1_pets	2.72E-04	2.72E-04	0	0	1.30E-03	3.71E-05
ssd_mobilenet_v2_coco	7.34E-05	7.34E-05	0	0	3.38E-04	1.36E-05
ssd_mobilenet_v2_focal_loss_pets_inference	9.93E-07	1.68E-06	0	0	4.05E-06	0
ssd_inception_v2_coco	1.25E-03	1.25E-03	0	0	4.43E-03	2.07E-05
ssd_inception_v2_pets	1.38E-03	1.39E-03	0	0	4.64E-03	7.59E-06
ssd_inception_v3_pets	1.74E-04	1.74E-04	0	0	7.86E-04	2.46E-06

Table 2 - Les valeurs de mAP du projet SUBIA

Modèle	AR					
	AR@1	AR@10	AR@100	AR@100 large	AR@100 medium	AR@100 small
embedded_ssd_mobilenet_v1_coco	1.14E-03	5.93E-03	0.01321	0.01321	0	0
ssdlite_mobilenet_v1_coco	8.12E-04	2.76E-03	6.57E-03	6.57E-03	0	0
ssdlite_mobilenet_v2_coco	2.44E-04	1.06E-03	5.52E-03	5.52E-03	0	0
ssd_mobilenet_v1_coco	0	5.56E-03	7.99E-03	7.99E-03	0	0
ssd_mobilenet_v1_focal_loss_pets	0	1.05E-03	1.14E-03	1.14E-03	0	0
ssd_mobilenet_v1_pets	1.20E-03	4.97E-03	7.79E-03	7.79E-03	0	0
ssd_mobilenet_v2_coco	0	4.53E-03	9.13E-03	9.13E-03	0	0
ssd_mobilenet_v2_focal_loss_pets_inference	0	0	1.54E-04	5.28E-04	0	0
ssd_inception_v2_coco	1.20E-03	5.51E-03	0.01275	0.01275	0	0
ssd_inception_v2_pets	1.10E-03	3.51E-03	9.16E-03	9.57E-03	0	0
ssd_inception_v3_pets	9.65E-04	4.49E-03	0.01647	0.01647	0	0

Table 3 - Les valeurs de AR du projet SUBIA

Les résultats de chaque modèles est représenté en échelle de 3 couleurs (rouge – jaune – vert), dans lesquelles :

- Le plus rouge est “mauvais” – c'est-à-dire, la qualité de modèle est inférieure à la moyenne
- Le jaune est “moyen”
- Le plus vert est “bon” – c'est-à-dire, la qualité de modèle est acceptable
- En plus, le gris indique la valeur de 0



Échelle de couleur avec 3 niveaux – Rouge, Jaune, Vert

Chaque critère ci-dessus sera visualisée par Tensorboard, un outil intégré dans Tensorflow pour suivre et évaluer les modèles via les outils graphiques – en temps réel.

Commentaires Préliminaires

1. Basé sur l'évolution de la fonction de perte (table 1), nous allons temporairement classer les 11 modèles en les 3 groupes – *Mobilenet (groupe A)*, *Inception (groupe B)* et *Intégré (groupe C)*, parmi lesquels :
 - a. Les modèles du groupe A sont similaires à ceux du groupe B, mais la perte de régularisation fluctue légèrement. En termes de performance, ce groupe est le meilleur avec 2 modèles “*ssd_mobilenet_v1_focal_loss_pets*” et “*ssd_mobilenet_v2_focal_loss_pets_inference*”
 - b. Pour le groupe B, la perte de classification, localisation et totale sont instable tandis que celle de la régularisation diminue stablement. Généralement, la performance de groupe est relativement baisse (niveau ‘orange’ et ‘rouge’)
 - c. Pour le groupe C, la perte de tous les quatre événements évolue stablement, dans laquelle seulement la perte de régularisation augmente régulièrement. La performance de cette groupe, en générale, est également baisse (niveau “orange-rouge”)
2. En se basant sur les valeurs de mAP – présentées dans le table 2, les 2 modèles “*ssd_inception_v2_coco*” et “*ssd_inception_v2_pets*” peuvent être considérées les meilleurs – c'est-à-dire, la valeur de mAP est plus haute.
3. En se basant sur les valeurs de AR – présentées dans le table 3, les trois modèles basées sur Inception et “*embedded_ssd_mobilenet_v1_coco*” sont les quatre meilleures modèles.
4. Bref, après avoir comparé et considéré les 3 commentaires ci-dessus, nous pouvons temporairement affirmer que “*ssd_inception_v2_coco*”, “*ssd_inception_v2_pets*” sont les meilleures modèles pour continuer à déployer.

Charger le modèle entraîné

Le modèle de détection des espèces marines est maintenant prêt afin d'être appliqué pour toutes les données. Pour faire cela, en premier lieu, on peut directement invoquer le dernier point de contrôle à l'étape ci-dessus ou geler le tous les paramètres dans un graphe d'inférence et puis, l'invoquer. Le script de Python, fourni par Google, de cette étape est disponible [ici](#).

Problèmes techniques restants

Bien que lors de ce stage j'aie réussi à implémenter, déployer les différents modèles pour la détection des espèces marines, il reste également les problèmes techniques objectifs et subjectifs.

Premièrement, concernant la qualité des données, en raison de la nature biologique, les espèces marines étant distribuées densement dans une petite zone (comme un ponton) et la forme de ces espèces étant également très diversifiée, cela conduit à une variété de caractéristiques. Néanmoins, la taille de données est trop petite (seulement 350 images), ce qui peut causer le surajustement du modèle - le problème dans lequel le modèle s'adapte parfaitement avec les données d'entraînement, mais s'adapte mal aux données de test – à l'extérieur du jeux de

données existant. Plus les données sont volumineuses, plus la capacité d'extraire des caractéristiques est précise – par exemple, le jeu de données de COCO comprend plus de 200,000 images avec une taille de 19 Gigaoctets.

Par la suite, technologiquement, bien que Google Collaboratory avec la version Professionnelle soit une plateforme parfaite pour implémenter et aussi déployer les modèles d'Apprentissage en Profond, le temps d'exécution ainsi que les ressources (processeur, mémoire,...) sont limitées – notamment par le fait que la machine virtuelle de Google sera redémarré après 24 heures – causant la perte de configurations et modèles. Pour les grands modèles ou les modèles ayant un grand nombre d'étapes (plus de 50,000), Google Collaboratory n'est pas une bonne solution.

Tous les codes du projet SUBIA sont fournis en public sur mon Gitlab de l'Université de La Rochelle, disponible [ici](#).

Conclusion

La dégradation de la biodiversité, en particulier des espèces marines, est un grave problème, non seulement pour La Rochelle, mais aussi pour le monde. En suivant les occurrences de chaque espèce, on peut évaluer la biodiversité ainsi que le niveau de la dégradation de biodiversité pour laquelle les spécialistes peuvent chercher les meilleures solutions pour protéger l'écologie marine. Ce stage a présenté une approche dans laquelle Intelligence Artificielle, notamment la Vision Numérique basée sur les modèles SSD pré-entraînés, a été appliquée afin d'analyser l'apparition des espèces marines.

Bien que les résultats soient encourageants, les modèles n'ont pas atteint les performances attendues en raison des limites concernant la taille ainsi que la qualité des données. En outre, les obstacles liés aux infrastructures (temps d'exécution du programme, capacité de la mémoire,...) empêchent d'obtenir la performance attendue des modèles.

En conséquence, dans l'avenir, ce projet se concentrera sur l'exploitation des données. Le premier pas sera l'amélioration de la qualité des données collectées, en augmentant la taille de l'ensemble de données et en essayant de distribuer plus uniformément les différentes données (réduire les écarts entre les représentations des différentes catégories de données). En outre, le nombre des labels sera également mis à jour pour s'adapter à la situation biologique actuelle.



En termes de la qualité des modèles, la suite de ce projet sera configuré de nombreuses étapes différentes pour déterminer la meilleure valeur tout en garantissant la valeur minimale possible de la fonction de perte ainsi que minimiser le surajustement (*overfitting*) et sous-ajustement (*underfitting*). Ensuite, en plus des 11 modèles déployés ci-dessus, ce projet va implémenter, déployer et évaluera les 50 modèles restants, basés sur le réseau Faster RCNN, Mask RCNN, RFCN-Resnet et ResNet. Les réseaux de YOLO seront les prochaines options de ce projet.

À la fin du stage, j'aurai acquis des connaissances et expériences sur le problème de la vision numérique ainsi que savoir comment implémenter et déployer une application de détection d'objet, du début à la fin. Technologiquement, j'ai également considérablement amélioré mes compétences en programmation Python et manipulé les bibliothèques et les frameworks associés, comme Tensorflow, Numpy, Matplotlib et Protocol Buffer. En termes de rédaction, j'ai également su comment préparer un document de recherche scientifique complet.

En termes de compétences non-techniques, bien que ce stage ait été réalisé entièrement à distance en raison de la pandémie, j'ai amélioré significativement mes compétences de communication via les visio-conférences avec mon maître de stage, ainsi que compétences en gestion du temps, talents pour la résolution des problèmes et pour la résistance au stress.

Références

- [1]. “UN Report : Nature’s Dangerous Decline ‘Unprecedented’ ; Species Extinction Rates ‘Accelerating’”, url : <https://www.un.org/sustainabledevelopment/blog/2019/05/nature-decline-unprecedented-report>
- [2]. “Facts and figures on marine biodiversity”, url : <http://www.unesco.org/new/en/natural-sciences/ioc-oceans/focus-areas/rio-20-ocean/blueprint-for-the-future-we-want/marine-biodiversity/facts-and-figures-on-marine-biodiversity/>
- [3]. D.Walther, D.R. Edgington, and C.Koch, “Detection and tracking of objects in underwater video”, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), vol.1, pp.544-549, June 2004
- [4]. L.Itti, C.Koch, and E.Niebur, “A model of saliency-based visual attention for rapid scene analysis”, IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI), vol. 20(11), pp.1254-1259, November 1998
- [5]. C.Spampinato, YH.Cheh-Burger, G.Nadarajan, and Robert B.Faher, “Detecting, tracking and counting fish in low quality unconstrained underwater video”, Proceedings of 3rd International Conference on Computer Vision Theory and Application (VISAPP), vol.2, pp.514-519, 2008
- [6]. P.L.Correia, P.Y.Lau, P.Fonseca, and A.Campos, “Underwater video analysis for Norway lobster stock quantification using multiple visual attention features”, Proceedings of 15th European Signal Processing Conference (EUSIPCO), Poznan, Poland, pp.1764-1768, September 2007
- [7]. K-H.Kim, S.Hong, B.Roh, Y.Cheon, and M.Park, “PVANET: Deep But Lightweight Neural Network for Real-Time Object Detection”, arXiv preprint ID 1611.08588, 2016
- [8]. E. Trucco and A. T. Olmos-Antillon, “Self-tuning underwater image restoration” IEEE Journal of Oceanic Engineering, vol. 31, no. 2, pp. 511–519, 2006.
- [9]. K. T. Mane and V. G. A. Pujari, “Novel approach for species detection from oceanographic video” in Fourth International Conference on Advanced Computing & Communication Technologies, IEEE, Piscataway, NJ, USA, 2014.
- [10]. C. Barat and R. Phlypo, “A fully automated method to detect and segment a manufactured object in an underwater color image” EURASIP Journal on Advances in Signal Processing, vol. 2010, no. 1, pp. 1–11, 2010.
- [11]. D. L. Rizzini, F. Kallasi, F. Oleari, and S. Caselli, “Investigation of vision-based underwater object detection with multiple datasets” International Journal of Advanced Robotic Systems, vol. 12, no. 6, pp. 1–13, 2015.
- [12]. A. Yamashita, M. Fujii, and T. Kaneko, “Color registration of underwater images for underwater sensing with consideration of light attenuation” in IEEE International Conference on Robotics & Automation, vol. 14, IEEE, Piscataway, NJ, USA, April 2007.
- [13]. A. Salman, A. Jalal, F. Shafait et al., “Fish species classification in unconstrained underwater environments based on deep learning: fish classification based on deep learning” Limnology and Oceanography: Methods, vol. 14, no. 9, pp. 570–585, 2016.
- [14]. G. Boussarie, N. Teichert, R. Lagarde, and D. Ponton, “BichiCAM, an Underwater Automated Video Tracking System for the Study of Migratory Dynamics of Benthic Diadromous Species in Streams” River Research and Applications, vol. 32, no. 6, pp. 1392–1401, 2016.
- [15]. P. Kannappan, J. H. Walker, A. Trembanis, H. G. Tanner, and O. Methods, “Identifying sea scallops from benthic camera images” Limnology and Oceanography: Methods, vol. 12, no. 10, pp. 680–693, 2014.
- [16]. J.Garcia, J.Fernandez, P.Sanz, and R.Marin, “Increasing autonomy within underwater intervention scenarios – the user interface approach”, Proceedings of the IEEE Systems Conference, pp.71-75, 2010

- [17]. F.Sun, J.Yu, S.Chen, and D.Xu, “*Active Visual Tracking of the Free-Swimming Robotic Fish-based on automatic recognition*”, Proceedings of the IEEE Conference on Intelligent Control and Automation, pp.2879-2884, 2014
- [18]. P.Sermanet, D.Eigen, X.Zhang, M.Mathieu, R.Fergus, and Y.LeCun, “*Overfeat – Integrated Recognition, localization and detection using convolutional networks*”, arXiv preprint ID 1312.6229, 2013
- [19]. S.Ren, K.He, R.Girshick. and J.Sun, “*Faster-RCNN – Towards Real-time Object Detection with region proposal networks*”, Proceedings of the Advances in Neural Information Processing Systems, pp.91-99, 2015
- [20]. B.Alshaوا, F.Maussang, R.Garello, and A.Chevallier, “*Marine Life Airbone Observation using HOG and SVM Classifier*”, Proceedings of the OCEAN 2016 MTS/IEEE Monterrey
- [21]. S.Kassin, “*Essentials of Psychology*”, © 2004 Publisher Prentice Hall
- [22]. D.H.Hubel and T.N.Wisel, “*Receptive fields and functional architecture of monkey striate cortex*”, Journal of Physiology, pp.215-243, March 1968
- [23]. K.Fukushima, “*Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*”, Biological Cybernetics, pp.193-202, 1980
- [24]. Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner, “*Gradient-Based Learning Applied to Document Recognition*”, Proceedings of the IEEE, Vol.86, Issue.11, pp.2278-2324, 1998
- [25]. A.Krizhevsky, I.Sutskever, and G.Hinton, “*ImageNet Classification with Deep Convolutional Neural Networks*”, Communication of the ACM, Vol.60, Issue.6, pp.84-90, 2017
- [26]. K.Simonyan and A.Zisserman, “*Very Deep Convolutional Networks For Large-Scale Image Recognition*”, Proceedings of International Conference on Learning Representation (ICLR 2015), arXiv preprint ID: 1409.1556
- [27]. K.He, X.Zhang, S.Ren, and J.Sun, “*Deep Residual Learning for Image Recognition*”, Proceedings of Computer Vision and Pattern Recognition (CVPR), pp.770-778, 2016
- [28]. C.Szegedy, W.Liu, Y.Jia, P.Sermanet, S.Reed, D.Anguelov, D.Erhan, V.Vanhoucke, and A.Rabinovich, “*Going Deeper with Convolutions*”, Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), arXiv preprint ID: 1409.4842, 2015
- [29]. C.Szegedy, V.Vanhoucke, S.Ioffe, and J.Shlens, “*Rethinking the Inception Architecture for Computer Vision*”, Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2818-2826, 2016, arXiv preprint ID: 1512.00567v3
- [30]. C.Szegedy, S.Ioffe, and V.Vanhoucke, “*Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*”, Proceedings of Thirty-First AAAI Conference on Artificial Intelligence, pp.4278-4284, 2017, arXiv preprint ID: 1602.07261
- [31]. A.G.Howard, M.Zhu, B.Chen, D.Kalenichenko, W.Wang, T.Weynard, M.Andreetto, and H.Adam, “*MobileNets : Efficient Convolutional Neural Networks for Mobile Visisons Applications*”, arXiv preprint ID: 1704.04861
- [32]. F.Chollet, “*Xception : Deep Learning with Depthwise Separable Convolutions*”, arXiv preprint ID: 1610.02357
- [33]. R.Girshick, J.Donahue, T.Darell, and J.Malik, “*Region-based Convolutional Networks for Accurate Object Detection and Segmentation*”, Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.38, Issue.1, pp.142-158, 2016
- [34]. R.Girshick, “*Fast-RCNN*”, Proceedings of 2015 IEEE International Conference on Computer Vision, pp.1440-1448, 2015, arXiv preprint ID: 1504.08083

- [35]. S.Ren, K.He, J.Girshick, and J.Sun, “*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*”, Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.39, Issue.6, pp.1137-1149, 2016, arXiv preprint ID: 1506.01497
- [36]. J.Redmon, S.Divvala, R.Girshick, and A.Farhadi, “*You Only Look Once : Unified, Real-Time Object Detection*”, Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.779-788, 2016, arXiv preprint ID: 1506.02640
- [37]. W.Liu, D.Anguelov, D.Erhan, C.Szegedy, S.Reed, C-Y.Fu, and A.C.Berg, “*SSD: Single Shot MultiBox Detector*”, European Conference on Computer Vision (ECCV) 2016, pp.21-37, arXiv preprint ID: 1512.02325
- [38]. M.Everingham, S.M.A.Eslami, L.V.Gool, C.K.I.Williams, J.Winn, and A.Zisserman, “*The PASCAL Visual Object Classes Challenge: A Retrospective*”, International Journal of Computer Vision, Vol.111, pp.98-136, 2015
- [39]. T.Gamauf, “*Tensorflow Records? What they are and how to use them?*”, url : <https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564>
- [40]. “*Developer Guide*”, url <https://developers.google.com/protocol-buffers/docs/overview>
- [41]. S.Popić, D.Pezer, B.Mrazovac, and N.Teslić, “*Performance Evaluation of Using Protocol Buffers in the Internet of Things Communication*”, Proceedings of 2016 International Conference on Smart Systems and Technologies (SST), pp.261-265, 2016
- [42]. T-Y.Lin, M.Maire, S.Belongie, L.Bourdev, R.Girshick, J.Hays, P.Perona, D.Ramanan, C.L.Zitnick, and P.Dollar, “*Microsoft COCO: Common Objects in Context*”, arXiv preprint ID: 1405.0312
- [43]. O.M.Parkhi, A.Vedaldi, A.Zisserman, and C.V.Jawahar, “*Cats and Dogs*”, Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp.3498-3505, 2012
- [44]. B.Zoph and Q.V.Le, “*Neural Architecture Search with Reinforcement Learning*”, arXiv preprint ID: 1611.01578
- [45]. K.He, G.Gkioxari, P.Dollar, and R.Girshick, “*Mask R-CNN*”, Proceedings of International Conference on Computer Vision (ICCV), pp.2980-2988, arXiv preprint ID: 1703.06870
- [46]. J.Dai, Y.Li, K.He, and J.Sun, “*R-FCN: Object Detection via Region-based Fully Convolutional Networks*”, Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 16), pp.379-387, 2016
- [47]. X.Glorot and Y.Bengio, “*Understanding the difficulty of training deep feedforward neural networks*”, Proceeding of AISTATS conference, vol.9, p.249-256, 2010
- [48]. C.Cortes, M.Mohri, and A.Rostamizadeh, “*L2 Regularization for Learning Kernels*”, UAI 2009, arXiv preprint ID 1205.2653
- [49]. D.P.Kingma and J.Ba (2014), “*Adam: A Method for Stochastic Optimization*”, arXiv preprint ID : 1412.6980v9

Annexe A: Implémentation du projet SUBIA

Cette annexe va présenter comment implémenter complètement le projet SUBIA en étape-par-étape.

Étape 0 : Globaliser les variables d'environnements avec [ce fichier](#).

Étape 1.1 : Préparation de l'environnement

- Installer les paquets et outils nécessaires

```
apt-get install -y python3-pip python3-lxml python3-tk protobuf-compiler git
```

- Installer les librairies et frameworks nécessaires via pip3 (ou pip avec Python 2)

```
pip3 install Cython Pandas Matplotlib Pillow Pathlib Tqdm Tf_slim Contextlib2
```

- Installer les APIs de COCO via pip3 (ou pip avec Python 2)

```
pip3 install pycocotools
```

- Déinstaller Tensorflow, Tensorflow GPU et Numpy (s'ils existent)

```
pip3 uninstall tensorflow tensorflow_gpu numpy
```

- Réinstaller Tensorflow GPU (avec la version spécifique 1.14.0) et Numpy (avec la version spécifique 1.15.4)

```
pip3 install tensorflow_gpu==1.14.0
```

```
pip3 install numpy==1.15.4
```

Étape 1.2 : Compilation de Protobuf Compiler pour modulariser le module “object_detection”

- Cloner la dernière version de modèle de Tensorflow en Github

```
git clone https://github.com/tensorflow/models.git
```

- Modulariser le module de OBJET_DETECTION avec 6 sous-étapes suivantes :

1. cd [tensorflow_model]/models/research
2. protoc object_detection/protos/*.proto --python_out=.
3. export PYTHONPATH=\$PYTHONPATH :`pwd` :`pwd`/slim
4. source ~/.bashrc
5. python3 setup.py build
6. python3 setup.py install

Étape 2 : Préparation des données

Les images et les labels bruts sont fournis par L3I et ils doivent être prétraités avant à continuer. En premier lieu, le script de nettoyage des données (sous-étape 1) seront exécuté avec la commande `python3 clean-dataset.py`

```
ttduy@ubuntu:~/Desktop/subia$ python3 clean-dataset.py
rm: refusing to remove '.' or '..' directory: skipping '.'
rm: refusing to remove '.' or '..' directory: skipping '..'
rm: refusing to remove '.' or '..' directory: skipping '.'
rm: refusing to remove '.' or '..' directory: skipping '..'
--- Finishing after 0.016696929931640625 seconds ---
--- Dataset is clean ---
```

Nettoyage des données

Ensuite, on va éliminer tous les données en format incorrects puis synchroniser le nombre des images et labels en lançant la commande `python3 sync-dataset.py`

```
ttduy@ubuntu:~/Desktop/subia$ python3 sync-dataset.py
--- Finishing after 0.2852766513824463 seconds ---
--- Dataset is synchronized ---
```

Balancer les données

Étape 3 : Partitionnement des données

Suivant, le jeu de données original est répartis en 2 parties, l'un pour l'apprentissage et l'autre pour le test. Puis, l'ensemble des données d'apprentissage sera divisé en 2 sous-ensembles : entraînement et évaluation, en se basant sur une proportion fixée. Exécuter la commande `python3 data-partition.py` et puis fournir les paramètres pertinentes :

- “Choose mode” indique où le partitionnement sera appliqué
 - o “all” or “*ALL*” pour les images et labels
 - o “image” or “*IMAGE*” pour seulement les images
 - o “annotation” or “*ANNOTATION*” pour seulement les labels
- “How many images for training model” indique le nombre exacte images pour l'apprentissage
- “Your training rate” indique la proportion de données pour l'entraînement. Cette ratio est entre 0 et 1

```
ttduy@ubuntu:~/Desktop/subia$ python3 data-partition.py
Choose mode :all
We have 359 images. Please choose a number < 359
How many images for training model :350
--- Generate dataset ---: 21%|███████████| 73/350 [00:06<00:23, 11.64it/s]
```

Division du jeu de donnée original

```
Your training rate :0.85
--- Generate training data ---: 100%|██████████| 298/298 [00:30<00:00, 9.85it/s]
--- Finishing after 71.5540783405304 seconds ---
--- Dataset is READY---
```

Organisation en deux sous-ensemble : entraînement et évaluation

Étape 4 : Conversion des labels au standard PASCAL VOC

Les labels bruts, fournis par L3I, sera convertis au standard PASCAL VOC en utilisant le script de Python `text2xml.py`

```
ttduy@ubuntu:~/Desktop/subia$ python3 text2xml.py
--- Train data ---: 100%|██████████| 298/298 [00:09<00:00, 32.68it/s]
--- Eval data ---: 100%|██████████| 52/52 [00:01<00:00, 34.78it/s]
--- Finishing after 10.623854398727417
--- XML converting is DONE---
```

Conversion des labels bruts au standard PASCAL VOC

Étape 5 : Générer les fichiers CSV à partir des fichiers XML

```
ttduy@ubuntu:~/Desktop/subia$ python3 xml2csv.py
---CSV Converting is DONE---
---CSV Converting is DONE---
```

Étape 6 : Crée la carte de label

Étape 7 : Générer TFRecord à partir des fichiers CSV. Il y a 3 fichiers de TFRecord qu'on doit le générer, l'un pour l'entraînement, et l'autre pour l'évaluation, en utilisant le script de Python `generate_tfrecord.py` avec l'utilisation de pipeline pour faire la liaison entre les images et le fichier CSV correspondant en utilisant 2 commandes suivantes :

```
python3 generate_tfrecord.py      -csv_input=[train_csv]
                                  --image_dir=[train_images]
                                  --output-path=XXX.record
```

```
python3 generate_tfrecord.py  
    --csv_input=[eval_csv]  
    --image_dir=[eval_images]  
    --output_path=YYY.record
```

Dans laquelle,

- “train_csv” et “eval_csv” sont au fur et à mesure le chemin absolu des fichiers CSV d’entraînement et d’évaluation
- “train_images” et “eval_images” respectivement indiquent où les images d’entraînement et d’évaluation sont stockés
- XXX.record et YYY.record indiquent au fur et à mesure les *TFRecord* d’entraînement et évaluation

Étape 7 : Configurer les paramètres de modèle, en modifiant ou ajoutant les paramètres, comme le nombre des étapes, le chemin absolu de la carte de label et de TFRecord, et les autres paramètres (si nécessaires)

Étape 8 : Entrainer le modèle en utilisant pipeline pour lier la configuration et la carte de label via pipeline avec le script Python – pré-intégré en Tensorflow

```
python3 model_main.py --logtosterr --model_dir=[logs] --pipeline_config_path=[config_file]
```

Dans laquelle,

- “logs” indique exactement où le graph-méta, le point de contrôle, et les fichiers de logs sont stockés
- “config_file” indique exactement le chemin absolu du fichier de configuration (avec l’extension .CONFIG)

Étape 9 : Visualiser la distribution des espèces marines avec le script Python *data-analyse.py*

Étape 10 : Visualiser les résultats du modèle en utilisant l’outil Tensorboard

```
tensorboard --logdir=[logs]
```

Dans laquelle,

- “logs” indique exactement où le graph-méta, le point de contrôle, et les fichiers de logs sont stockés

Étape 11 : Modulariser le modèle entraîné sous forme d’un graph inféré gelé – en utilisant le script Python pré-intégré en Tensorflow

```
python3 export_inference.py --input_type image_tensor --pipeline_config_path [config_file]  
                            --trained_checkpoint_prefix [logs]/model.ckpt-XXXX --output_directory [output_dir]
```

Dans laquelle,

- “config_file” indique exactement le chemin absolu du fichier de configuration (avec l’extension .CONFIG)
- “logs” indique exactement où le graph-méta, le point de contrôle, et les fichiers de logs sont stockés
- XXXX est le nombre maximal du point de contrôle, il est aussi le nombre d’étape maximum
- “output_dir” indique où le graph inféré gelé seront stocké