Reconnaissance d'organismes marins dans des images photographiques sous-marines

Mémoire de fin d'étude

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours DECOL

Auteur
Sébastien Villon

Superviseurs
Marc Chaumont
Jérôme Pasquet
Gérard Subsol
Thomas Claverie
David Mouillot
Sébastien Villéger

Lieu de stage
LIRMM UM5506 - CNRS, Université de Montpellier

soutenu publiquement le 11 Juin 2015

Résumé

Ce document présente l'étude de techniques d'imagerie pour la reconnaissance, la classification et le comptage de poissons dans des vidéos sous-marines. Plus particulièrement, nous y abordons les aspects particuliers de la reconnaissance d'objets dans ce contexte et nous développons une méthode peu ou pas utilisée dans la littérature. Nous terminons par une présentation de nos résultats expérimentaux et de nos apports à l'état de l'art ainsi que par une discussion sur de futures améliorations

---

Abstract

This document presents the study we have done on the thematic of recognizing, classifying and counting fishes on submarine videos. We will see the particular aspects of object recognition and we will process a way of recognizing picture either not used or underused. To conclude, we will show the results of our experiments as well as our contributions to the state of the art and a discussion on possibles futures upgrades of our system.

Table des matières

Remerciements

Je tiens tout d'abord à remercier Jérôme Pasquet et Marc Chaumont pour m'avoir proposé ce stage et pour m'avoir aidé et encadré tout au long de celui-ci. Je tiens aussi à remercier Gérard Subsol, Sébastien Villéger, David Mouillot et Thomas Claverie pour m'avoir apporté leur expertise tout au long de ce projet ainsi que Lionel Pibre avec lequel j'ai beaucoup partagé pendant ce stage. Je remercie finalement toute l'équipe ICAR pour l'environnement de travail plus qu'agréable dans lequel j'ai pu faire évoluer mon projet de recherche, et NVIDIA corporation associée au fournisseur CARRI pour nous avoir fourni les machines sans lesquelles il n'aurait pas été possible de travailler.

1

Introduction

1.1 Présentation générale

La quantification de l'impact des activités humaines sur l'environnement est un enjeu majeur pour la conservation de la biodiversité des écosystèmes sous-marins. La plus grande difficulté, pour la surveillance de ces écosystèmes, est l'analyse de données fiables à grande échelle.
Les techniques de surveillance par extraction, comme la pêche, ne fournissent que des informations limitées à certaines espèces et l'interprétation des données récupérées peut être biaisée par l'échantillonnage [10]. De plus, le recours à la pêche, même pour la surveillance, impacte la biodiversité étudiée. Il existe aujourd'hui de nombreuses façons de collecter les informations sous-marines, entre autre par comptage visuels, que nous pouvons diviser en deux catégories.
Les études "manuelles", nécessitant deux plongeurs prenant des annotations à la main, effectuées lors de la plongée. Cette catégorie est la plus coûteuse en temps et en moyens, et sa qualité dépend de nombreux facteurs (expérience du plongeur, poissons craintifs...).
Les nouvelles techniques en plein essor dans le cadre du comptage et de l'étude des espèces par vidéos sont les méthodes RUV (Remote Underwater Video) et DOV (Diver Operated Video)[10]. Les études vidéo enregistrent des quantités de données beaucoup plus importantes (112 teraoctets acquis dans le cadre du projet européen Fish4knowledge [5]) mais sont extrêmement longues à analyser manuellement. C'est dans le but de traiter efficacement ces nombreux enregistrements vidéo que sont envisagées les techniques de traitements de l'image.

La plupart de ces traitements se décomposent en trois phases [14] : la détection, le suivi et la reconnaissance. Notons que certains auteurs [13, 11] regroupent la phase de détection et de reconnaissance en cherchant à détecter directement les espèces de poissons, sans situer au préalable une zone précise de l'image à étudier.
Nous avons décidé d'orienter notre étude sur une détection image par image. En effet, si une détection sur l'ensemble de la vidéo peut être assistée par un système de suivi, ce système va utiliser des à priori (arrière plan fixe pour les caméras fixes par exemple). Nous souhaitons que notre système soit adaptable, aussi bien pour une étude en caméra fixe que pour une analyse de vidéos filmées par des ROV (remote operated video). De plus, les suivis par analyse de mouvement sont beaucoup plus efficaces *offline*, c'est-à-

dire une fois que le mouvement complet est terminé, que *online*. Une analyse par image permettrait donc d'analyser les poissons présents en temps réel.

1.2 Contexte du stage

Mon travail a été effectué en association avec le laboratoire MARBEC (MARine Biodiversity Exploitation & Conservation [1]), spécialisé dans l'étude des changements de la biodiversité dans les écosystèmes marins pour optimiser leur exploitation et leur gouvernance. Dans le cadre de notre étude et de cette collaboration, nous avons décidé d'utiliser les données du projet lifeCLEF [2]. LifeCLEF est une campagne d'évaluation organisée depuis 2003 par le laboratoire CLEF Initiative. Elle est divisée en quatre sous tâches dont l'objectif global est un "benchmarking" des techniques d'annotations et de classification automatique d'images : classification médicale, classification d'oiseaux, de poissons et de plantes. Le résultat des propositions des participants de cette année sera présenté à la conférence "Evaluation Forum (CLEF)" du 8 au 11 septembre 2015.

Nous nous servirons plus particulièrement de la tâche *Image Based Fish Identification and Species Recognition* pour laquelle l'organisation met à disposition 285 vidéos labellisées contenant plus de 20000 images de poissons annotées. L'utilisation de cette base de données nous permettra de comparer notre approche à celles des autres participants, avec une base de connaissance commune. Les données mises à disposition sont un ensemble d'imagettes servant pour l'apprentissage et 20 vidéos accompagnées d'annotations sous formes d'un formulaire XML, contenant les poissons à identifier.

Pour la détection d'images dans un contexte sous-marin, les difficultés majeures sont les changements de couleur, les différences de lumière d'une image à l'autre, les sédiments présents dans l'eau et les plantes sous-marines (arrière-plan changeant) créant beaucoup de bruit sur l'image, ainsi que la faible qualité des images due à la résolution de base et à la compression [1]. Pour la classification des poissons, des individus d'une même espèce peuvent présenter des différences de taille, d'orientation, de mouvement (positions) en plus des particularités individuelles (changement de forme ou de couleur entre sexe ou entre adultes et junéviles). Nous avons pris la décision de traiter ces difficultés inhérentes au cas d'étude en conditions réelles plutôt que de les éluder grâce à des conditions d'acquisition contrôlées et standardisées, en bassin, de profil, comme dans l'article [14].

[1] http ://www.umr-marbec.fr/fr/
[2] http ://www.imageclef.org/lifeclef/2015

2

Deep Learning et Réseaux Convolutionnels

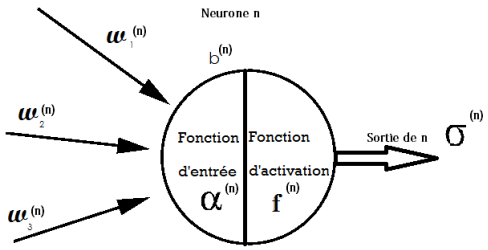
2.1 Introduction

L'article [14], cherchant lui aussi à reconnaître et classer des espèces de poissons dans des images, montre que l'utilisation des réseaux neuronaux a obtenu de meilleurs résultats que les calculs de distances euclidiennes (EDM, Euclidean Distance Matrix), en passant de 81% de résultats positifs à 99% dans des conditions identiques. Bien que l'ANN (Artificial Neural Networks) ait des temps de calcul bien supérieurs à l'EDM, de l'ordre de 6.3, la précision du classifieur neuronal est largement supérieure. Nous avons également vu dans notre analyse bibliographique que les réseaux neuronaux présentaient globalement de bonnes performances dans le cadre de la reconnaissance d'images.

2.2 Généralités

Les réseaux de neurones artificiels sont des modèles mathématiques qui tentent de reproduire le comportement du cerveau humain [2].Un ANN est composé d'un ensemble interconnecté de nœuds, appelés neurones. Chaque neurone n reçoit un ensemble de valeur des neurones précedents qui sont modifiés grâce à : une fonction d'entrée $\alpha^{(n)}$, une fonction d'activation $f^{(n)}$ qui transforme l'entrée du neurone et définit l'information qu'il transfèrera aux neurones suivants, une pondération (ou poids) $\mathbf{w}^{(n)} \in \mathbb{R}^c$ avec c le nombre de connexions en entrée, et un biais $b^{(n)}$ qui servira de seuil à la fonction d'entrée. Une fois l'entrée du neurone modifiée, elle est envoyée au neurones suivants

FIGURE 2.1 : Représentation détaillée d'un neurone



Par la suite, avec c le nombre de connexions en entrée, on définit $\alpha^{(n)}$ comme :

$$\alpha^{(n)}(\mathbf{x}^{(n)}) = \sum_{i=1}^{c} w_i^{(n)} x_i^{(n)} + b^{(n)}$$

Soit $\sigma^{(n)}$ la sortie du neurone n et $f^{(n)}$ sa fonction d'activation :

$$\sigma^{(n)}(\mathbf{x}^{(n)}) = f^{(n)}(\alpha^{(n)}(\mathbf{x}^{(n)}))$$

Les réseaux que nous employons utilisent une fonction d'activation ReLU, soit, pour un réel r :

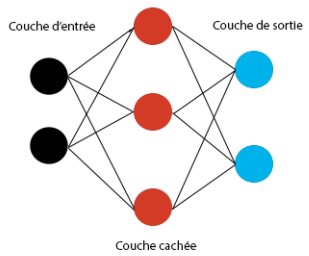
$$f(r) = max(0, r)$$

ou une fonction d'activation sigmoïde, soit

$f(r) = \frac{1}{1+e^{-r}}$

Ces neurones sont interconnectés en couches, c'est-à-dire que les neurones d'une couche ne sont connectés en entrée, qu'aux neurones de la couche précedente, et en sortie qu'aux neurones de la couche suivante. La première couche du réseau est la couche d'entrée, qui reçoit les images. La dernière couche est celle de sortie qui correspond aux classes retournées par le réseau, c'est-à-dire les classes que l'ont cherche à identifier. Les couches intermédiaires ou couches cachées contiennent des neurones connectés à tous les neurones de la couche précédente et de la couche suivante (exemple de réseau à une couche en figure 2.2). L'objectif du réseau est de modifier les paramètres de ses neurones par apprentissage grâce à des méthodes que nous verrons par la suite pour améliorer la classification.

FIGURE 2.2 : Réseau de neurones



Pendant l'entraînement d'un ANN, on utilise un mécanisme de rétro-propagation afin de corriger et d'améliorer le réseau, c'est-à-dire modifier les poids associés aux neurones afin de minimiser l'erreur de classification :
Soit un vecteur représentant un objet de classe $k \in \{1..K\}$ qui est passé en entrée du

réseau, on compare la valeur attendue (100% de probabilité d'appartenir à la classe k) à la valeur obtenue, puis on calcule l'erreur de la couche de sortie. On propage ensuite l'erreur des neurones des couches j aux neurones des couches $j-1$. On met ensuite à jour les poids de chacun des neurones, afin de faire converger le résultat de l'analyse du vecteur de classe k vers un meilleur résultat [16].

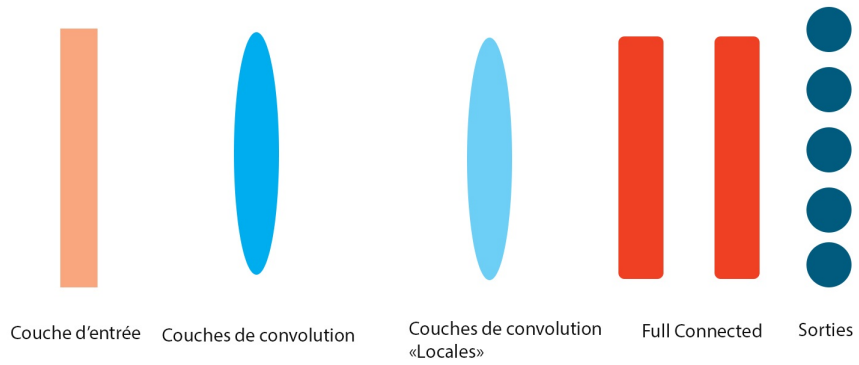
2.3 Deep learning et Réseaux Convolutifs

Les réseaux *Deep Learning* (Deep neural network ou DNN) sont des ANN possédant plusieurs couches cachées. Les réseaux neuronaux de convolution (CNN) sont utilisés pour permettre une abstraction supérieure. Pour ce faire, une ou plusieurs couches de *convolution* sont utilisées connectées à un ANN classique [9]. Chacune des couches de convolution effectue sur ses données d'entrée une transformation grâce à un noyau de convolution et une étape de *pooling*, qui rend l'apprentissage plus robuste , puis une étape de non linéarité. Une fois les n étapes de convolution effectuées, les sorties sont données à un réseau de neurones classique (ANN).

Détail d'un réseau convolutif

Un réseau d'apprentissage prend en entrée une base de couples $(I^i, l^i)_{i=1}^{i=N}$ avec I^i, l'imagette de numéro $i \in \{1, ..., N\}$. Dans notre cas, les imagettes sont des matrices de pixels carrées de taille 32*32. Le label $l^i \in \{1, ..., 16\}$correspond à l'espèce contenue dans l'imagette, avec une valeur de 1 à 15 si il y a un poisson et la valeur 16 si il contient autre chose.

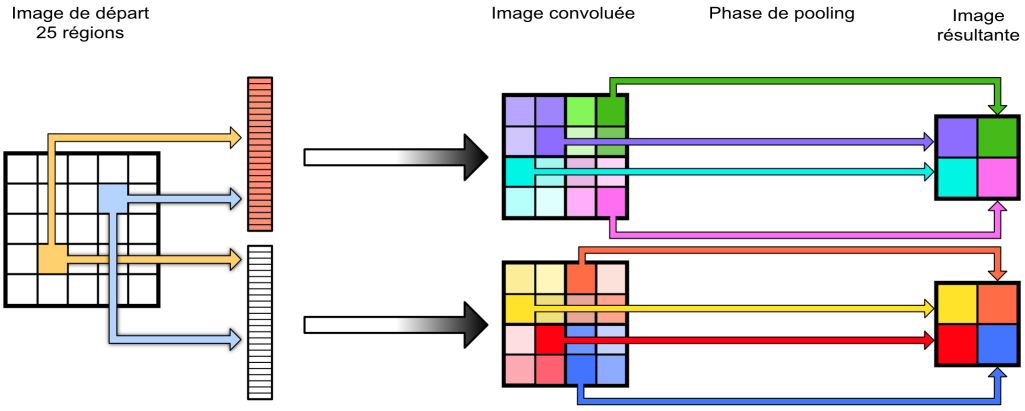
FIGURE 2.3 : Exemple simplifié d'un réseau convolutif de deep learning, avec deux couches cachées ou "Full connected"



Après la couche d'entré, l'image donnée en entrée passe par une première phase de convolution. Comme nous l'avons vu plus haut, cette phase est elle-même séparable en trois sous parties.

Premièrement, on déplace une fenêtre glissante sur l'imagette (voir figure 2.4) et pour chaque région de l'image prise par la fenêtre glissante, un neurone appartenant à un ensemble de neurones, applique un produit scalaire sur l'image pour la transformer grâce à une matrice de transformation d'image, ou *kernel de convolution*. Tous les ensembles de neurones transforment la même image, donnant ainsi plusieurs images transformées pour une même image d'entrée. On appelle une image transformé par un groupe de neurones partageant le même poids une $features\ map$.

FIGURE 2.4 : Convolution et pooling



Nous appliquons ensuite sur les images convoluées une phase de *pooling*, en général soit un *Average Pooling*, soit un *Max Pooling*, c'est-à-dire que l'on passe sur notre image transformée une nouvelle fenêtre glissante de taille fixe. Pour chaque sous échantillon selectionné par la fenêtre glissante, nous conservons une seule valeur de pixel. L'*Average Pooling* sélectionne, pour une région, la moyenne des valeurs des composantes de cette région, et le *max pooling* la composante de valeur maximale.

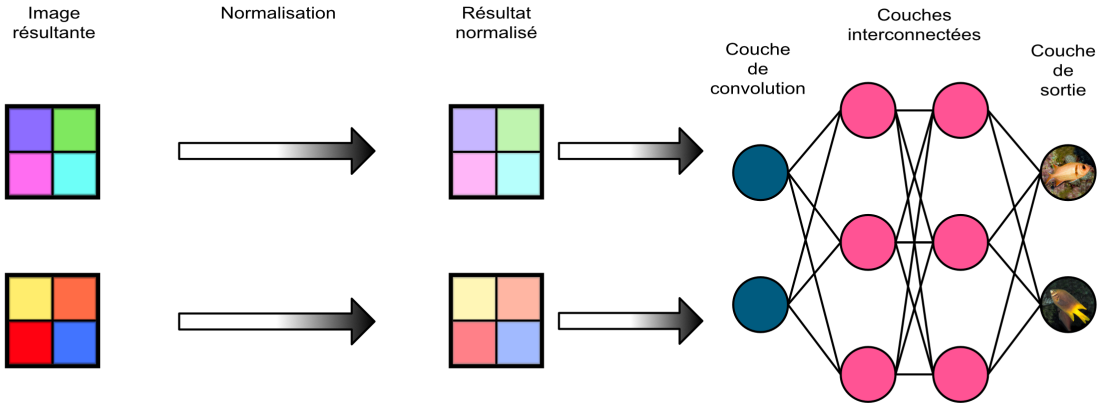
Après cette étape nous obtenons donc une nouvelle matrice de valeurs représentatives résultant de la phase de *pooling*. Celles-ci passent ensuite par une fonction de normalisation. Celle que nous utilisons est une fonction normalisant chaque sortie du neurone grâce aux sorties des neurones des autres ensemble de neurones à la même position.

La fonction de normalisation f^{norm} d'un neurone u d'un ensemble de neurone i avec $i \in \{1..F\}$que nous utilisons peut être formalisée ainsi, avec F le nombre de $features\ maps$, et α et β des paramètres du réseau :

$$f^{norm}(u_i^{x,y}) = \frac{u^{x,y}}{(1+\frac{\alpha}{N}\sum_{i'=max(0,i-\lfloor N/2 \rfloor)}^{min(F,i-\lfloor N/2 \rfloor+N)}(u_{i'}^{x,y})^2)^\beta}$$

Cette fonction possède un attribut de taille N qui permet de diviser la sortie du neurone actuelle par les sorties des autres neurones à la même place dans d'autres ensembles de neurones. Par exemple, avec une taille de normalisation de 5, le neurone à la première

FIGURE 2.5 : Pooling et normalisation



position de l'ensemble 7 sera normalisé par les neurones en position 1 des ensembles 5 à 9. Nous utilisons cette fonction pour la normalisation pixel par pixel qu'elle effectue.

Après cette phase de normalisation nous obtenons une version transformée de notre image d'origine (voir fig. 2.5). Il est ensuite possible de continuer à modifier cette image avec des couches de convolutions simples ou *locales*. Le principe de la couche de convolution locale est le même que la convolution simple, à la différence que les *kernels* à l'intérieur d'un même ensemble sont différents. Une fois toutes les transformations effectuées, les matrices obtenues à partir des images de bases, ou plus simplement $features$, sont envoyées aux couches de neurones interconnectées ou *couches cachées* telles que vues plus haut. Ces features sont ensuite classés dans une des classes disponibles grâce à une couche particulière comme la couche contenant la fonction $softmax$. Cette fonction permet de pondérer les sorties afin d'obtenir des probabilités $pr \in [0, 1]$ pour chaque classe et pour chaque image. La fonction softmax est définie comme :

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{16} e^{x_j}}$$

avec y_i la sortie pour une classe, et x_i les entrées du neurone i

Dans notre cas, chaque layer utilisant la back propagation possède trois paramètres :

- ϵ, le poids du taux d'apprentissage
- γ, le moment du poids
- wc, le $weigth - decay$ vu plus haut

A partir de ces valeurs, les formules de calcul de mise à jour des poids :

$$\Delta w_t^{(n)} := \gamma . w_{t-1}^{(n)} - wc.\epsilon.w_{t-1} + \epsilon.\delta w_t$$

$$w_t^{(n)} := w_{t-1}^{(n)} + \Delta w_t^{(n)}$$

Avec $\delta e w_i$ le gradient moyen obtenu par séquence d'apprentissage, et t le temps pour lequel on applique la modification.

Une fois l'ensemble des images d'entraînement analysé, le réseau utilise une base annexe fournie pour effectuer une validation. La fin de la validation marque la fin d'un cycle, ou *epoch*, et le commencement d'un nouveau cycle avec les mêmes images.

Tout comme les ANN, les CNN utilisent des fonctions de rétro-propagation pour se réajuster pendant l'apprentissage afin de converger. L'ajout de couches cachées d'abstraction soumet les CNN à des erreurs de surentraînement. Pour lutter contre ce surentraînement, ou sur-apprentissage, les CNN utilisent aussi les fonctions de régularisation comme les fonctions *weight decay* [3] et *dropout* [7]. Le *weight decay*, vu plus haut dans la formule de normalisation permet de pondérer la valeur de la modification au cours du temps, et le drop-out permet de ne pas activer tous les neurones à chaque apprentissage afin d'être certain que les neurones ne sur-apprennent pas.

3

Données techniques

3.1 Choix de l'implémentation

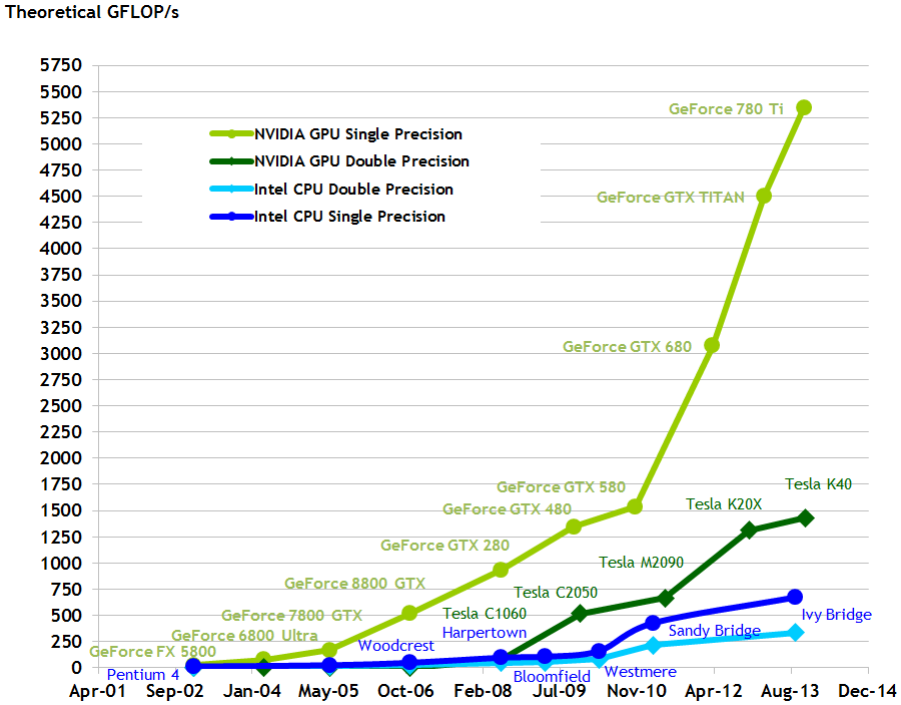
Pour notre réseau, nous avions la possibilité d'utiliser une implémentation existante, ou de développer notre propre structure. En raison des contraintes de temps, nous avons cherché un système pré-existant qui conviendrait à ce que nous cherchions. Les deux classifieurs que nous avons trouvés correspondant à nos attentes ont été EBlearn et CUDA-convnet. EBlearn propose un système d'apprentissage qui semble cependant moins performant en terme de temps de calcul. Il utilise une conversion en matrice matlab plutôt qu'en fichiers batch, mais propose moins de possibilités de réutilisation une fois l'apprentissage terminé. Nous avons finalement porté notre choix sur CUDA-convnet.

3.2 CUDA

Utiliser un réseau de type Deep Learning nécessite un nombre de calculs gigantesques, et que cela soit sur un simple cœur ou même un supercalculateur multi-cœur, le temps d'apprentissage est de plusieurs jours voire plusieurs semaines. Il est donc nécessaire d'utiliser des architectures spécialisées dans le calcul de scalaires. On peut effectuer de tels calculs sur une unité graphique. Les GPU sont en effet maintenant dotés de milliers de processeurs et certaines architectures proposent d'exploiter cette puissance à des fins calculatoires. CUDA a été développée par NVIDIA afin d'augmenter les performances d'une machine en utilisant les processeurs graphiques (GPU) (figure 3.1 [1]). C'est une implémentation exclusive NVIDIA permettant de dériver des calculs parallèles sur des centaines voir des milliers de processeurs GPU. Ce type de système est très utilisé dans de nombreux domaines nécessitant beaucoup de ressources tels que le traitement d'images et de vidéos ou encore la modélisation numérique en biologie. Dans le cas du traitement de données par réseaux neuronaux, cette possibilité nous permettait de grandement diminuer les temps d'apprentissage et de test et ainsi de pouvoir tester davantage de possibilités. Nous pouvons voir sur la figure 3.1 l'impact de l'utilisation de GPU en terme de données traitées. L'augmentation pour des calculs "simple précision" entre CPU et GPU représente bien l'utilité que peuvent avoir des architectures utilisant la puissance GPU pour des travaux nécessitant une grande puissance de calcul.

[1] http ://docs.nvidia.com/cuda

FIGURE 3.1 : Comparaison de la vitesse de calculs GPU/CPU



3.3 Convnet 2

CUDA-Convnet2 est une implémentation de réseau neuronal en langage CUDA permettant l'exploitation du GPU par le réseau. L'utilisation d'une telle technologie était nécessaire étant donné les demandes en temps de calculs des réseaux neuronaux. Cette implémentation dispose d'une surcouche python permettant la manipulation des différents *layers* d'un réseau. L'utilisation d'une implémentation existante induit cependant de nombreuses contraintes telles que l'utilisation du format utilisé et le temps d'adaptation et de prise en main.

L'architecture des réseaux convnet est celle présentée plus haut dans le chapitre 2.3. Les nombreux paramètres qui affectent les résultats seront détaillés dans la section suivante. En plus de l'entraînement de réseaux, CUDA-convnet permet de récupérer les $features$ créées au cours de l'évolution du réseau, ainsi que les poids de celui-ci afin de pouvoir le réutiliser. La récupération de $features$ permet d'utiliser un classifieur annexe pour la classification, ou la réutilisation de ceux-ci. Il est aussi possible de récupérer et d'afficher les différents filtres convolutionnels utilisés afin de les réutiliser ou d'observer leurs caractéristiques. De nombreuses autres options permettent, entre autres, d'afficher les filtres utilisés ou encore les images à différents niveaux du réseau.

3.4 Materiel utilisé

La principale particularité de CUDA-convnent2 est l'utilisation du langage NVIDIA-CUDA [2] exploitant l'architecture des carte GPU. Cette implémentation n'est disponible que sur certaines cartes graphiques spécifiques. N'ayant pas la possibilité d'utiliser CUDA sur nos machines personnelles, nous avons tout d'abord dû trouver une machine disponible avec une *GPU Capability* suffisante. Cependant, ces machines n'étaient pas suffisamment puissantes pour nos réseaux et les sorties de mémoires fréquentes nuirent grandement à nos avancées. Nous avons donc directement contacté NVIDIA par le biais duquel nous avons eu accès à une machine de haut-de-gamme grâce au fournisseur CARRI :

- CPU : 2x Intel Xeon E5-2670v2
- Memory : 64Go DDR3 1600MHz ECC
- Compute GPU : 2x nVidia Tesla k80m
- SSD 240Go, HDD 2To

C'est grâce à ce matériel que nous avons effectué la majeure partie de nos tests, et si l'implémentation CUDA permet d'obtenir des temps de calculs bien plus compétitifs, il est important de noter que cela a un coût matériel relativement important, ici la machine sur laquelle nous avons travaillé ayant approximativement un prix de 15000$.

[2] https ://developer.nvidia.com/cuda-gpus

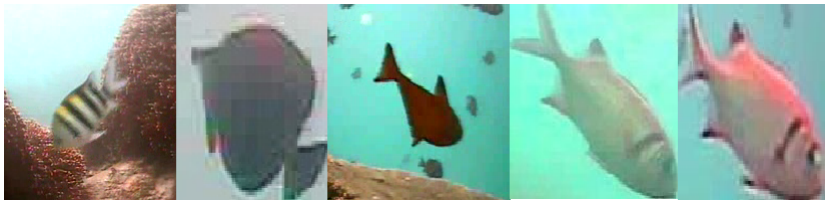
4

Expérimentation

4.1 Données

Nous avons décidé de traiter deux types de données. Premièrement les données fournies directement sous forme d'imagettes (voir figure 4.1), c'est-à-dire des parties d'image qui on été découpées et centrées autour du poisson. Dans les données 'FishCLEF' [1] que nous utilisons, ces imagettes sont de tailles et de qualités variables. Le deuxième jeu de données est composé de vidéos prises en caméra fixe, d'une durée d'environ 15 secondes, et contenant plusieurs espèces de poissons. Nous détaillerons dans cette partie les protocoles utilisés pour préparer ces données aux réseaux de neurones que nous allons utiliser.

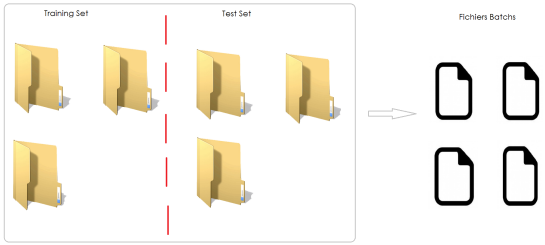
FIGURE 4.1 : Images disponibles dans la base de test LifeCLEF



Nous commençons le traitement avec 15 dossiers contenant 22000 imagettes de poissons. Le nombre d'image est très inégal entre les espèces, avec 24 images seulement pour la classe en contenant le moins le *Chaetodon speculum*, et plus de 3000 pour le *Dascyllus reticulatus*. La première étape de ce traitement est un redimensionnement des images afin de leur donner un format standard et de les rendre robustes au changement de taille. Nous obtenons après ce filtre un corpus C1 simple et peu volumineux. Nous verrons plus tard les autres corpus que nous avons utilisés. Une fois ces corpus obtenus, ils sont divisés aléatoirement en 5 dossiers d'apprentissage et 5 dossiers de test (voir figure 4.2), puis les images sont converties dans des fichiers batchs utilisables par CUDA-convnet.

[1] http ://www.imageclef.org/lifeclef/2015

FIGURE 4.2 : Traitement des images



4.2 Importance de la structure du réseau

Afin de s'assurer que les résultats présentés représentaient au mieux les résultats moyens qu'il était possible d'obtenir, nous avons, pour chaque expérimentation, divisé aléatoirement les images disponibles en un ensemble d'apprentissage, un ensemble de validation et un ensemble de test.
Pour des questions de temps, les réseaux ont tout d'abord été entraînés sur le corpus C1 afin de voir si des différences nous permettaient de sélectionner les réseaux les plus performants. En effet, il faut compter entre 3h et plus d'une journée pour qu'un réseau converge, selon la taille de celui-ci.
Nous allons présenter ici les trois paliers d'amélioration notables que nous avons pu extraire de nos expérimentations.

Premier palier

Nous avons commencé par tester le réseau $Cifar$ (voir figure 4.4)(un réseau utilisé pour de la reconnaissance de nombres calligraphiés) sur C1 afin d'avoir comme base un réseau fonctionnant sur la reconnaissance d'image (voir figure 4.3)

Les résultats sur notre jeu de données donnent des résultats peu fructueux avec environ 12% de précision en moyenne sur les 10 jeux de données. Le réseau converge rapidement, en environ une heure et demi.

Second palier

Nous avons ensuite ajouté un réseau interconnecté de deux couches de 2000 neurones aux couches de convolutions afin d'avoir un réseau d'apprentissage plus complet. Avec deux couches cachées à la suite du réseau précédent, les résultats d'évaluations passe de 12% à 40% de réussite, ce qui s'explique par l'augmentation de la spécificité des images. L'ajout de ces deux couches double cependant le temps de calcul (voir figure 4.4).

Troisième palier

Le dernier résultat obtenu avec C1 a atteint les 47% avec 3 réseaux différents (voir figure 4.5). Ces réseaux comportent trois couches de convolution, deux couches locales, et deux couches interconnectées. Les trois réseaux implementent aussi une méthode pour lutter contre le sur-apprentissage.

FIGURE 4.3 : Réseau Cifar - Deux couches de convolution de 32 filtres, Stride de 2 (c'est-à-dire décalage de deux pixel entre chaque région), une fonction de max pooling, et une fonction de normalisation cmrnorm. Toutes les fonctions d'activations des neurones sont des fonctions ReLU et la décision de classement est effectuée grâce à une couche softmax

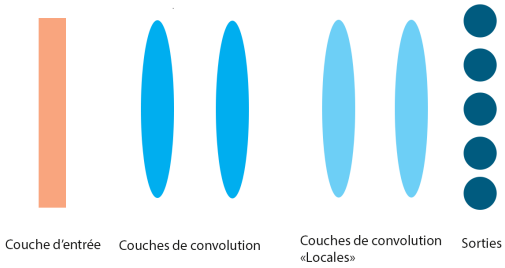


FIGURE 4.4 : Réseau 2 - Deux couches de convolution, identiques à CIFAR, une fonction de max pooling, et une fonction de normalisation cmrnorm. Toutes les fonctions d'activations des neurones sont des fonctions ReLU, y compris ceux des nouvelles couches cachées.

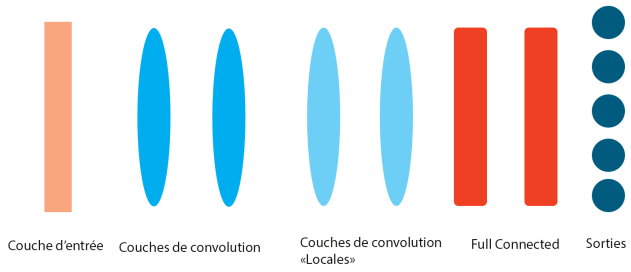
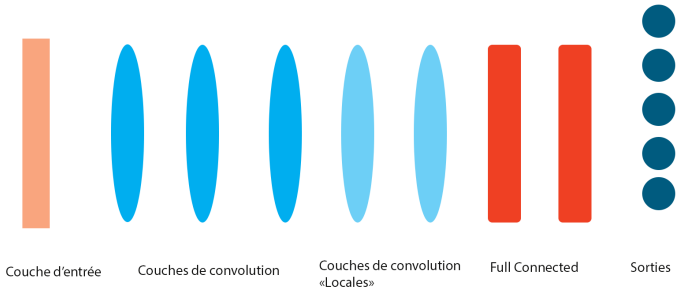


FIGURE 4.5 : Réseau 3,4,5 - Ces trois réseaux implémentent 3 couches de convolution, et deux couches *full connected* de 4000 neurones. La fonction de max pooling à été remplacée par une fonction average pooling pour les réseaux 4 et 5, et la fonction d'activation des couches cachées n'est plus une fonction ReLU mais une fonction logistique pour les réseaux 3 et 4.



Bilan sur l'importance du réseau

Nous avons pu comparer l'importance de différentes variations des réseaux. Bien que nous n'ayons présenté plus haut que les réseaux qui apportaient une grande amélioration, nous avons testé un grand nombre de combinaisons de $features$. Une des surprises lors de nos tests fut la réussite de l'utilisation des couches de convolutions locales (voir figure 4.5), pourtant très peu utilisées dans la littérature. Lors de la suppression de cette couche de convolution locale, il était nécessaire de doubler le nombre de filtres de la couche de convolution pour arriver au même résultat, augmentant du même fait le temps de calcul de façon significative, mais aussi le nombre de cycles nécessaires pour permettre au réseau de converger. Nous avons aussi pu mettre en évidence l'impact de différentes fonctions de normalisation. Si la variation de celles-ci ne fait pas beaucoup de différences lorsque le réseau est de taille réduite, nous avons constaté que la fonction logistique était plus efficace dans notre cas. Les tailles des différents *kernels* pour les couches de convolutions et les opérations de pooling sont elles aussi importante, de mauvaises dimensions ayant fait chuter les résultats à 30% de précision ou même 8% lorsque le pooling n'est pas adapté à l'image. Avec dans notre cas un taux de réussite plus élevé avec des tailles respectives d'un quart ou d'un huitième de l'image à traiter pour la convolution, et une transformation de taille quatre pour le pooling. Les fonctions de robustesse au surentraînement ont aussi été testées avec différents paramètres, mais dans notre cas elles n'ont permis aucune amélioration, ou même dégradé les résultats du réseau.

Une fois les réseaux 3,4,5 à disposition nous avons pu envisager le passage à l'échelle avec des données plus importantes. Tout au long de ces tests nous avons continué à modifier nos réseaux ainsi que la façon de construire les batchs utilisés par CUDA-Convnet pour les adapter aux retours de nos tests.

4.3 Importance du jeu de données

La deuxième phase du traitement des données, après le redimensionnement, consiste à effectuer trois rotations de 90, 180 et 270 degrés afin de gagner en robustesse ([8])et d'augmenter la taille de la base . Nous obtenons un nouveau corpus C2 de 86 000 images. La troisième phase est une phase d'équilibrage, afin d'éviter que l'apprentissage des classes contenant beaucoup d'images n'écrase l'apprentissage des classes ne disposant pas de beaucoup d'images. A cette fin, nous dupliquons les images jusqu'à en obtenir le même nombre pour toutes les classes. Ainsi nous disposons d'un troisième corpus C3 de 220 000 images. Un dernier corpus C4 a été obtenu en utilisant aussi des images de fond afin d'introduire une classe utilisée lorsque le réseau ne reconnaît aucun poisson. Le dernier corpus C4 compte 300 000 images.

Le corpus C2 contenant les images de base et ses quatre rotations à 90,180 et 270 degrés a été testé avec les trois réseaux 3,4,5. Nous nous sommes aperçus que le nombre de filtres n'intervenait plus dans une certaine mesure car les trois convergent vers 70% de réussite en moyenne, mais ajoutent en temps de calcul. Nous avons aussi remarqué que le passage d'une fonction logistique à une fonction ReLU n'influençait que peu les

résultats, mais permettait une convergence plus rapide en terme de cycles d'apprentissage.

Le corpus C3 qui contient les classes équilibrées a été analysé de la même façon. Les résultats furent globalement moins bons, la meilleure composition ayant permis d'attendre 55% d'espèces détectées et la pire n'ayant que 24% de réussite. Nous avons réalisé que les espèces avec le moins d'échantillons présents montraient aussi la moins bonne qualité d'image, ce qui pourrait expliquer les résultats obtenus.

Finalement, en ajoutant des images de fond à C3, les résultats de la détection sont sensiblement remontés jusqu'à atteindre une moyenne de 31% de réussite.

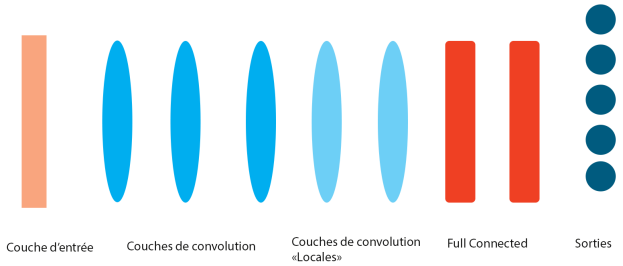
Bilan sur l'importance des données utilisés

Nous avons pu comparer l'importance des données utilisées dans le cadre de l'apprentissage pour la reconnaissance d'images. Tout d'abord, la transformation du corpus C1 en C2 montre que la quantité d'informations servant à l'apprentissage influe grandement sur la qualité des résultats du réseau. Nous avons aussi pu étudier grâce au corpus C3 qu'au-delà de la quantité, la qualité de l'image est aussi très importante puisque même en augmentant le nombre d'images, si la qualité n'est pas exploitable le réseau échoue plus fréquemment. La taille des images est elle aussi importante, puisque de plus grandes images permettent d'appliquer plus de couches de convolutions, et potentiellement des images transformées plus représentatives.

4.4 Résultats finaux

Vu la chute des résultats lors du passage de C2 à C3 nous avons décidé de créer un 5ème corpus c5, qui contient les images de fond ajoutées cette fois directement à C2, sans passer par la duplication d'images et donc sans équilibrage du corpus. Nous avons ainsi obtenu de très loin nos meilleurs résultats, avec 80% de réussite en moyenne, et jusqu'à 86% dans le meilleur des cas. Nous avons obtenu ces résultats en utilisant le réseau présenté en figure 4.6 sur le nouveau corpus c5.

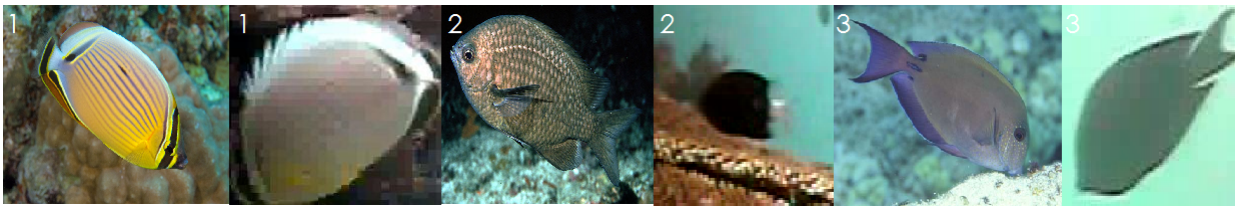
FIGURE 4.6 : Réseau final - 3 couches de convolutions de 128 filtres, 2 couches de convolution locales de 64 filtres et deux couches *full connected* de 4000 neurones. Les fonctions de pooling sont des fonctions "average pooling", et la fonction d'activation des couches cachées est une fonction logistique. La classification est réalisée par un soft-max



4.5 Problèmes rencontrés

La première difficulté à mettre en avant est la qualité des images utilisées 4.7. En effet la qualité de la base des échantillons utilisés influencera non seulement l'ensemble des résultats mais aussi la construction du réseau car celui-ci s'adapte aux informations qu'il traite. D'après les échanges avec nos partenaires du laboratoire MARBEC, il est possible d'obtenir des images de bien meilleure qualité, et potentiellement d'augmenter la qualité du résultat, celui-ci pouvant être plus précis dans ses fonctions. De plus il est important d'enseigner au réseau à reconnaître des images du même type que celles qui vont lui être demandé de reconnaître. Il est probable que la qualité des images récupérées aille en s'améliorant, et il est donc important d'entraîner le réseau dans ce sens. Un nombre plus important d'imagettes par espèce permettrait d'avoir plus de positions et d'exemples, et donc d'améliorer les résultats.

FIGURE 4.7 : Comparaison entre les images disponibles et des images de bonne qualité. Liste des espèces dans l'ordre : 1) Chaetodon lunulatus, 2) Chromis chrysura, 3) Acanthurus nigrofuscus



La seconde difficulté est inhérente à l'utilisation d'une plateforme déjà existante. Le format batch utilisé par CUDA-convnet2 ainsi que la restriction du format des images induisirent de nombreuses erreurs de conversion et de formalisme. Il est important de noter que ces erreurs ne se révèlent pas nécessairement immédiatement, mais parfois pendant l'apprentissage, voire au bout de plusieurs heures. Il est alors nécessaire de reprendre toutes les opérations de création de corpus et d'apprentissage depuis le début.

La troisième difficulté est d'ailleurs le temps. En effet, travaillant avec des volumes relativement importants de données et surtout un nombre de calculs considérable, la moindre opération prend beaucoup de temps. Les opérations de transformations d'images en fichiers batchs peuvent durer plusieurs heures, les migrations entre les machines locales et les serveurs distants aussi, et les réseaux neuronaux ont besoin de plusieurs heures ou même d'une journée complète pour converger. De plus la perte de connexion entre la machine locale et la machine distante, une erreur sur le serveur, ou comme vu plus haut, une erreur non détectée dans le corpus, demande plusieurs heures de calculs additionnels.

Ces temps sont calculés sur la machine haut-de-gamme décrite plus haut, sans laquelle il ne nous auraient pas été possible de travailler. Cependant la fin du contrat de location avec le fournisseur a ajouté un autre délai pendant lequel nous ne pouvions pas travailler.

La dernière difficulté notable est le manque de documentation *technique*, bien que la documentation théorique foisonne. Nous pensions qu'en utilisant l'implémentation la plus courante de réseaux neuronaux le problème ne se poserait pas, mais ce manque a induit un nombre d'heures de prise en main beaucoup plus conséquent.

Bien que ces points techniques ne soient pas des difficultés complexes à résoudre, ils augmentent drastiquement le facteur de temps, et malheureusement dans le contexte d'un stage de trois mois, ce facteur s'est montré plus crucial que la complexité théorique de l'apprentissage par le deep learning et de l'analyse d'images.

5

Extensions et perspectives

5.1 Amélioration du réseau

Nous avons cherché à améliorer l'apprentissage du réseau. Pour le moment, celui-ci utilise un vecteur caractéristique représentant l'image, transformé grâce à une couche de convolution. Nous pensons qu'il pourrait être intéressant d'intégrer à ce vecteur caractéristique d'autres informations représentatives, comme l'ajout des données d'un histogramme de gradient orienté (HOG).

Le principe de l'histogramme de gradient orienté est de décrire un objet dans une image grâce à ses contours et aux formes qui le composent en utilisant la distribution d'orientation et d'intensité du gradient. L'article [12] montre que l'utilisation de vecteurs caractéristiques HOG permet de meilleurs résultats dans le cadre de la détection dans des situations peu favorables (objets incomplets, difficiles à reconnaître), ce qui correspond aux difficultés rencontrées lors de la détection et de la reconnaissance en milieu sous-marin : poissons cachés ou partiellement présents, bruit important sur l'image.
La plupart des algorithmes de détection utilisent des vecteurs représentatifs particuliers (tailles, ratios de dimensions, teintes), ou des représentations SIFT [11, 17, 4] ou *shape context* (SC [15]). La représentation de données par des vecteurs HOG est plus efficace [6] que les représentations SIFT et SC [18] dans le cadre de la détection de personnes. Pour ce stage, nous avons choisi d'utiliser une représentation HOG.

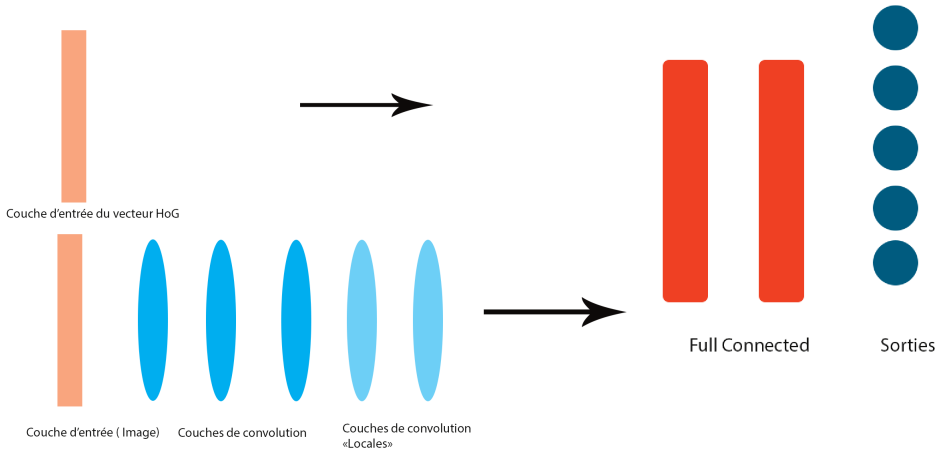
Une fois le vecteur caractéristique HOG otenu, nous comptons l'intégrer au réseau en le concaténant au vecteur obtenu, pour une même image, après le réseau de convolution 5.1.

5.2 Détection de poissons dans des vidéos sous-marines

Une seconde amélioration de notre outil est l'application de la détection, non plus à des imagettes préformatées, mais à des vidéos.

Pour ce faire, nous commençons par découper les vidéos disponibles. Dans notre cas, nous possédons 76 vidéos issues du challenge fishCLEF, d'une durée de 12 à 15 secondes et en format 320*240. Nous commençons par extraire de cette vidéo un certain nombre de frames, en considérant que trop d'images par seconde ajoutent inutilement du temps de calcul mais que trop peu peuvent diminuer la réussite de la détection. Dans notre cas nous avons estimé que 25 images par secondes étaient suffisantes, ce qui nous donne un total de 26390 frames.

FIGURE 5.1 : Concaténation du nouveau vecteur caractéristique au réseau pré-existant



À l'intérieur de ces frames, nous faisons ensuite passer une fenêtre glissante de taille variable. Les dimensions de cette fenêtre sont tout d'abord celles de la frame, puis diminuent progressivement jusqu'à récupérer l'image sous forme de petites parcelles. La taille minimum de la fenêtre glissante est estimée comme *la plus petite taille possible pour laquelle il est possible d'avoir un vecteur caractéristique exploitable*. Nous pensons qu'une taille minimum de 32*32 est nécessaire afin de pouvoir utiliser un réseau de $deep - learning$. À la fin de ces découpages nous obtenons un total de plus de 35 millions d'images. Ces images seront ensuite transmises au réseau afin que celui-ci évalue, pour chacune, les probabilités qu'elles appartiennent à une espèce déjà apprise ou, le cas échéant, reconnaître le fond marin.

Le facteur de temps a été plus que limitant dans notre travail, et une prolongation d'un an en tant qu'ingénieur d'étude a été accepté par le labex NUMEV afin de continuer le développement de mes recherches et de créer un prototype utilisable par le laboratoire MARBEC.

Nous comptons aussi au cours de l'année prochaine finaliser les extensions que nous avions prévues tel que l'intégration d'autres données caractéristiques au réseau et la détection de poissons dans des vidéos.

6

Conclusion

Nous avons réalisé une étude sur l'apport que l'utilisation de réseaux convolutionnels de neurones pourrait avoir à l'état de l'art sur la reconnaissance et l'identification d'espèces sous-marines dans des images, et nos expériences ont permis de clarifier le potentiel d'un tel type de classifieur.

Cette étude nous a permis de démontrer la viabilité de l'utilisation de réseaux de neurones pour la reconnaissance d'images sous-marines. Nous avons obtenu des résultats équivalents voire supérieurs aux articles présentés dans notre étude bibliographique et démontré l'impact des types de réseaux et de données utilisés pour améliorer d'avantage les taux de réussite.

Nous sommes confiants dans l'apport que de nouvelles données de meilleure qualité pourraient apporter au réseau et de la réussite de l'utilisation de ce type de classification par le deep-learning dans le contexte étudié, et comptons au cours de l'année prochaine améliorer notre approche grâce à la continuation de notre collaboration avec l'équipe MARBEC dans le cadre d'une année en tant qu'ingénieur d'étude.

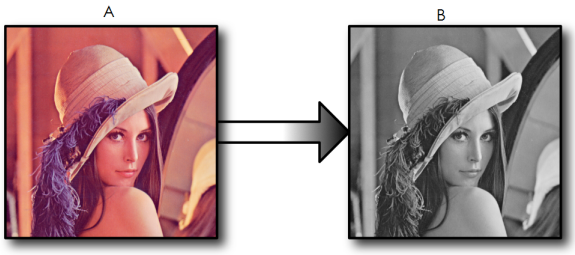
7

Annexe

7.1 Méthodologie d'extraction des HOG

Pour extraire l'histogramme de gradient orienté d'une image avant de l'intégrer à notre réseau, plusieurs étapes sont nécessaires. On commence par passer l'image en niveau de gris (ref1). On utilise ensuite un filtre vertical et un filtre horizontal pour générer deux image dérivées G_x et G_y (ref2). A l'aide de ces deux images, on génère deux matrices V et W, la première représentant l'orientation des gradients et la seconde contenant la norme de ces vecteurs (ref3).

FIGURE 7.1 : Passage de l'image de base en niveau de gris



On obtient l'orientation θ du gradient d'un pixel p : $\theta_p = \arctan(\frac{G_y}{G_x})$

La norme du gradient pour un pixel p est obtenu ainsi : $N_p = \sqrt{G_x(p)^2 + G_y(p)^2}$

La fusion des deux matrices permet donc de connaître pour chaque pixel la direction de son gradient ainsi que sa norme. On divise ensuite l'image en région, puis on calcule pour chaque région la distribution des gradients pondérée par leur norme(ref4). La concaténation des histogrammes représentant la distribution de chaque région permet d'obtenir un vecteur caractéristique représentant l'image.

FIGURE 7.2 : Transformation des images et générations des matrices

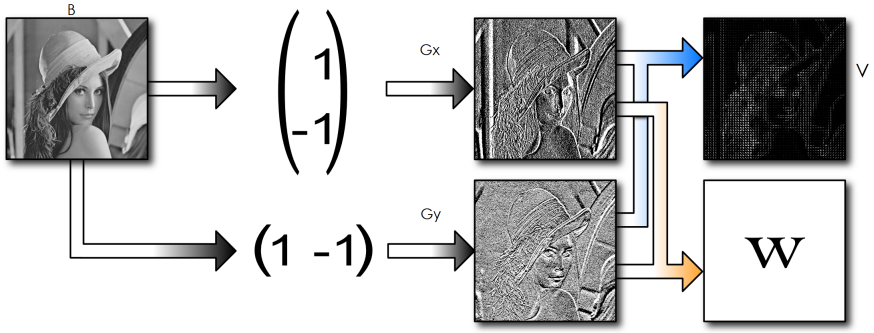
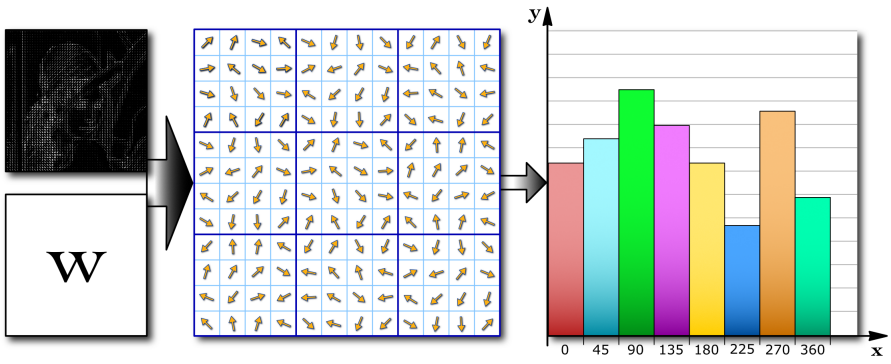


FIGURE 7.3 : Obtention de l'Histogramme orienté de gradient



Bibliographie

[1] Mutasem Khalil Sari Alsmadi, Khairuddin Bin Omar, Shahrul Azman Noah, and Ibrahim Almarashdah. Fish recognition based on the combination between robust feature selection, image segmentation and geometrical parameter techniques using artificial neural network and decision tree. *International Journal of Computer Science and Information Security, IJCSIS November 2009, ISSN 1947 5500*, 2009.

[2] Peter M Atkinson and ARL Tatnall. Introduction neural networks in remote sensing. *International Journal of remote sensing*, 18(4) :699–709, 1997.

[3] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE, 2013.

[4] Katy Blanc, Diane Lingrand, and Frédéric Precioso. Fish species recognition from video using svm classifier. In *Working Notes of CLEF 2014 Conference*, 2014.

[5] Bastiaan J Boom, Phoenix X Huang, Cigdem Beyan, Concetto Spampinato, Simone Palazzo, Jiyin He, Emmanuelle Beauxis-Aussalet, Sun-In Lin, Hsiu-Mei Chou, Gayathri Nadarajan, et al. Long-term underwater camera surveillance for monitoring and analysis of fish populations. *VAIB12*, 2012.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*, 2012.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[10] Delphine Mallet and Dominique Pelletier. Underwater video techniques for observing coastal marine biodiversity : A review of sixty years of publications (1952–2012). *Fisheries Research*, 154 :44–62, 2014.

[11] J Matai, R Kastner, GR Cutter Jr, and DA Demer. Automated techniques for detection and recognition of fishes using computer vision algorithms. In *NOAA Technical Memorandum NMFS-F/SPO-121, Report of the National Marine Fisheries Service Automated Image Processing Workshop, Williams K., Rooper C., Harms J., Eds., Seattle, Washington (September 4–7 2010)*, 2012.

[12] Jerôme Pasquet, Marc Chaumont, and Gérard Subsol. Comparaison de la segmentation pixel et segmentation objet pour la détection d'objets multiples et variables dans les images. *Colloque compression et représentation des signaux Audiovisuels*, 2014.

[13] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on lie algebra. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 728–735. IEEE, 2006.

[14] C. Pornpanomchai, B Lurstwut, P. Leerasakultham, and W Kitiyanan. Shape- and texture-based fish image recognition system. In *Kasetsart J*, 2013.

[15] Andrew Rova, Greg Mori, and Lawrence M Dill. One fish, two fish, butterfish, trumpeter : Recognizing fish in underwater video. In *MVA*, pages 404–407, 2007.

[16] Jürgen Schmidhuber. Deep learning in neural networks : An overview. *Neural Networks*, 61 :85–117, 2015.

[17] Yi-Haur Shiau, Sun-In Lin, Yi-Hsuan Chen, Shi-Wei Lo, and Chaur-Chin Chen. Fish observation, detection, recognition and verification in the real world. In *International Conference on Image Processing, Computer Vision and Pattern Recognition*, 2012.

[18] Qiang Zhu, M-C Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.