UNIVERSITÉ DE FRANCHE-COMTÉ

INRA SCIENCE & IMPACT

Projet Final

Détecter, identifier et localiser des oiseaux ravageurs des cultures

Khalid Bouabdallaoui

Maître de stage : Corentin Barbu
Encadrant : Professeur Célestin C. Kokonejdi
Université de Franche-Comté - Août 2019

"This work has been granted by Plant2Pro® Carnot Institute in the frame of its « 2016-2017-2018 » call for projects. Plant2Pro® is supported by ANR (#18-CARN-024-01)".

Table des matières

Table des figures

Chapitre 1

Remerciement

Tout d'abord, j'adresse mes remerciements à mon maître de stage, Mr Corentin BARBU, Directeur de recherches à l'INRA, pour son accueil, son encadrement, pour avoir donné les conseils et orientations nécessaires, et enfin pour m'avoir donné l'opportunité de découvrir la perspective professionnelle que pouvait offrir ma formation dans le domaine agronomique. Je choisis ce moment pour reconnaître sa contribution avec reconnaissance.

Je remercie aussi Jean-Marc TEULE et Gilles GRANDEAU, pour leurs conseils, et leurs investissements dans la partie «Matériels» de ce projet.

Je remercie également tout le personnel de l'INRA, pour leur accueil chaleureux et leur aide lors de ce stage.

D'autre part, je tiens à remercier l'ensemble des professeurs du master de modélisation statistique de l'université de Franche-Comté pour leurs enseignements, notamment Monsieur Kokonendji Célestin C, professeur en mathématiques appliquées, pour son accompagnement pédagogique.

Mes remerciements s'étendent également à mes camarades du master de modélisation statistique ainsi qu'à mes proches pour leur aide et leur soutien tout au long de ces deux années.

Chapitre 2

Introduction

Les dégâts causés par les attaques récurrentes d'oiseaux ravageurs, représentent une véritable cause d'incertitude pour les producteurs, en particulier pour les cultures de tournesol et de maïs. Environ un tiers des surfaces en tournesol seraient touchées chaque année[1], entraînant potentiellement des resemis ou des pertes de récolte. Ces attaques ont ainsi un impact direct sur la rentabilité de ces cultures. De plus, la crainte des dégâts engendrés et les frais qui en découlent remettent en question le maintien même de ces cultures en France, bien qu'elles présentent un intérêt agronomique et environnemental[2]. Les dégâts d'oiseaux réduisent ainsi les possibilités de diversification et les bienfaits que peut apporter cette culture comme précédent dans la rotation culturale.

Remédier à cette problématique est devenu un enjeu majeur pour la compétitivité de certaines filières de production[3]. Plusieurs méthodes classiques de lutte au champ existent déjà : effarouchage, tir, produits répulsifs, aménagement des pratiques culturales (semis plus tardifs,...). L'effarouchage reste la méthode la plus facilement mobilisable. Parmi les effaroucheurs, on citera notamment les canons à gaz qui font à intervalle régulier le bruit d'un coup de fusil, les épouvantails («*scaryman*») mobiles ou non et les bandes sonores de cris de détresse ou de prédateurs[4]. L'efficacité de ces techniques reste limitée dans le temps, les oiseaux s'habituant rapidement à l'effarouchage, des dispositifs capables de réagir en temps réel à la présence des oiseaux à leur insensibilité croissante pourraient allonger les durées d'efficacité. Par ailleurs, les capacités d'analyse et de quantification de ce fléau sont limitées. D'autant plus que les destructions sont très hétérogènes dans l'espace et dans le temps et que le recours à des observateurs humains pour les décrire avec précision est coûteux.

Les technologies numériques peuvent être d'un grand secours pour l'identification des oiseaux que ce soit à des fins d'effarouchage ou de recherche. Pour cela il est possible de combiner des capteurs existants avec des algorithmes entraînés pour analyser les images comme des observateurs humains pour démultiplier la capacité d'observation et de quantification. L'utilisation de réseaux de neurones profonds a permis ce type d'approche dans plusieurs domaines tels que :

- Reconnaissance faciale (Google, Facebook, Apple, Sony)[5]
- Reconnaissance optique de caractères (traitement automatique des chèques)[6]
- Reconnaissance vocale (Siri, Google translate, sous-titrage automatique) [7]
- Reconnaissance d'empreintes[8]

- Identification d'objets ou de personnes sur des photos ou dans des vidéos[9]

Ces techniques permettent la détection et l'identification en temps réel, souvent sur des dispositifs de calcul de taille réduite qui sont en association directe avec les capteurs (matériel embarqué). Parmi ces dispositifs, le raspberry Pi[10] s'est taillé une place de choix pour le prototypage. Le nombre d'images nécessaires pour entraîner les algorithmes étant très important, il est nécessaire de constituer des banques d'images. Dans notre cas, il n'est pas possible de se baser sur celles existantes car les images d'oiseaux sur fond de terre labourée juste semée n'abondent pas parmi celles recensées par les moteurs de recherche. L'annotation d'images est nécessaire à l'exploitation de telles bases de données, et étant donné la grande taille des jeux d'images considérés, une annotation semi-automatique peut permettre de ne présenter à l'observateur humain que les cas difficiles permettant de faire progresser les algorithmes d'entraînement[11]. Plusieurs systèmes téléchargeables gratuitement proposent ce type d'annotation d'images (Supervisely[12], Labelbox[13], labelimg[14]...). Il existe aussi des sites internet et des entreprises spécialisées dans l'annotation mais qui sont payants tel que Dataturks[15]. Une fois l'annotation terminée, l'étape suivante consiste à utiliser cette banque d'images annotées pour l'apprentissage d'algorithmes (Deep-learning). Les réseaux à convolution, malgré leur complexité, sont fortement conseillés pour la classification d'images ou la reconnaissance visuelle. Ce sont des domaines où ils surpassent toutes les autres méthodes existantes grâce à leur vitesse d'apprentissage[16] et à leur précision. La mise en place d'un tel procédé pour classifier des images complexes peut s'avérer difficile, car de nombreux paramètres sont à déterminer[17] : nombre d'images à utiliser pour l'apprentissage, taille des images, type de filtres, méthode de *Pooling*[18], nombre de couches de neurones, nombre de neurones par couche. La difficulté à trouver la bonne combinaison entre ces paramètres demande du temps pour la réalisation de réseaux de neurones performants. De plus, le temps d'entraînement pour chaque structure de réseau de neurone est important. C'est pour cette raison qu'il est utile d'utiliser des réseaux de neurones pré-entraînés qui garantissent à la fois une structure du réseau performante et limitent le temps nécessaire à l'apprentissage. En particulier, MobileNet SSD[19] a été utilisé sur Raspberry pi pour permettre de la reconnaissance en temps réel. Cependant ces réseaux de neurones pré-entraînés ne peuvent traiter que des images de taille limitée (300 px de côté pour MobileNet) en raison du coup exponentiel en termes de nombre de paramètres lorsque le nombre de pixels augmente. Pour permettre le traitement, sans perte d'information, des images à haute définition nécessaires pour surveiller un champ, il faut alors assurer une subdivision de l'image. Se focaliser sur les différences entre deux images successives pourrait permettre de se concentrer sur des fragments de l'image, de taille raisonnable, où des changements ont été observés et donc susceptible de contenir des oiseaux. Ces fragments peuvent alors être traités, sans perte d'information, par le réseau de neurones.

Dans le projet finançant le stage, nous développons un prototype complet de dispositif basé sur un *Raspberry Pi* permettant la prise d'image au champ et leur analyse en temps réel pour détecter, localiser et identifier des oiseaux déprédateurs se posant sur des parcelles juste semées.

L'année précédent le stage, une vaste campagne d'acquisition au champ et d'annotation d'images a été lancée. Le stage a permis de participer à cette acquisition et d'apporter des capacités d'annotation semi-automatique. Le cœur de ce stage consiste à adapter et comparer des algorithmes existants pour ce problème spécifique de détection avec des moyens de calcul embarqué et donc limités. Une aide à la réalisation de l'interface permettant de piloter l'acquisition et le traitement a aussi été apportée. Trois axes sont développés au centre de ces algorithmes de traitement : la mise

en évidence de différences entre deux images consécutives pour détecter des zones de changements (imagettes), le référencement des images par rapport à des cartes permet le positionnement des imagettes dans le champ et l'application d'algorithmes de deep learning aux imagettes doit permettre d'identifier les oiseaux, éventuellement en distinguant différentes espèces comme les corvidés et les colombidés.

Chapitre 3

Matériels et méthodes

3.1 Acquisition des images

Le stage a permis dans un premier temps de participer à l'acquisition d'images pour constituer la base de données de référence pour ce projet. Nous les avons acquis de deux manières :

1. Installation de caméras dans les champs
2. Utilisation d'un Raspberry

3.1.1 Caméras dans les champs

Plusieurs caméras sont installées dans différentes parcelles de chanvre et de tournesol au moment du semis.

Deux marques de caméras sont utilisées :

FIGURE 3.1 – Pièges photographiqies : *Bushnell* et *Berger & Shröter*

Les pièges photographiques ci-dessus sont robustes et peuvent supporter les conditions climatiques du terrain. Les caméras sont équipées de carte mémoire SD, et sont programmées pour faire une capture d'image toutes les quinze secondes entre 4h et 22h.

3.1.2 Raspberry Pi

Matériel

Le Raspberry Pi est un nano-ordinateur : de la taille d'une carte de crédit(3.2), il est équipé du strict nécessaire :

- Un microprocesseur ARM
- La mémoire RAM
- Une carte vidéo
- Une carte ethernet
- Un module ratio (Wi-fi et Bluetooth)

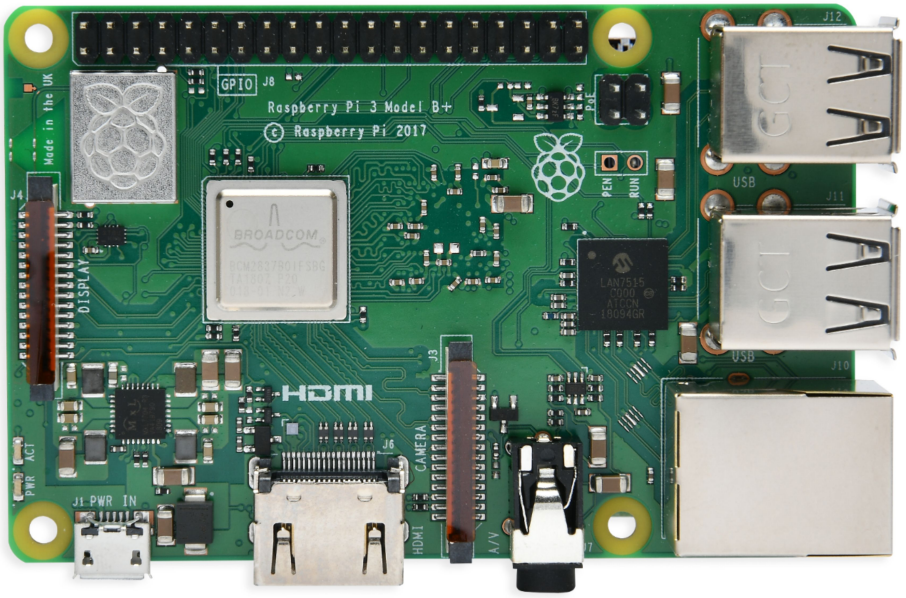


FIGURE 3.2 – La carte *Raspberry Pi* 3

A l'aide de ce nano-ordinateur et de la Raspberry Pi Camera V2, nous avons conçu un dispositif qui permet la prise d'images dans les champs (3.3).

Ce dernier est facilement transportable dans les parcelles. Un transformateur alimente le Raspberry assurant la transformation de l'énergie fournie par une batterie de voiture (3.4). Le dispositif a une autonomie supérieure à un mois pour la simple prise d'image toutes les 15 secondes.

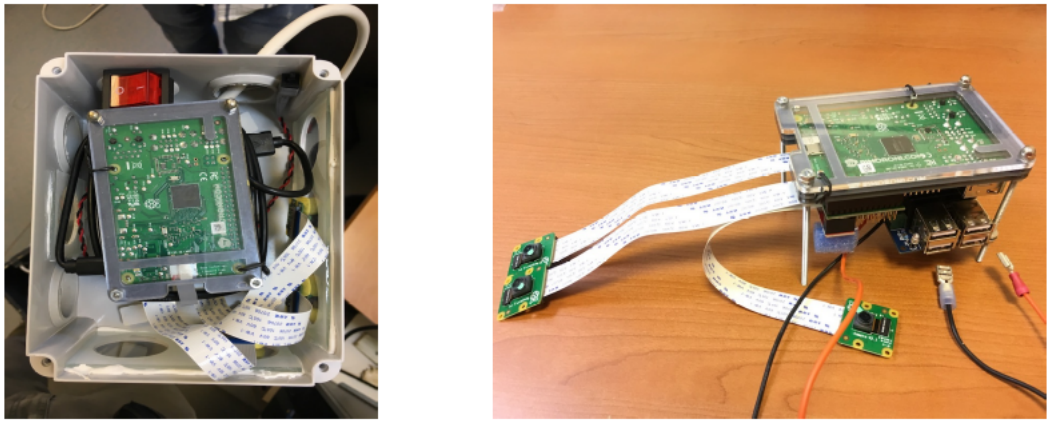


FIGURE 3.3 – Vue du prototype multi-caméra en cours d'assemblage

FIGURE 3.4 – Vue du prototype multi-caméra assemblé

Interface Web

Une interface web (3.5) est mise en place afin de faciliter le pilotage du dispositif, elle permet par ailleurs de récupérer plusieurs informations importantes telles que les données de géolocalisation

de la parcelle et à définir un "masque" délimitant la zone sur laquelle nous allons faire nos analyses (3.6).

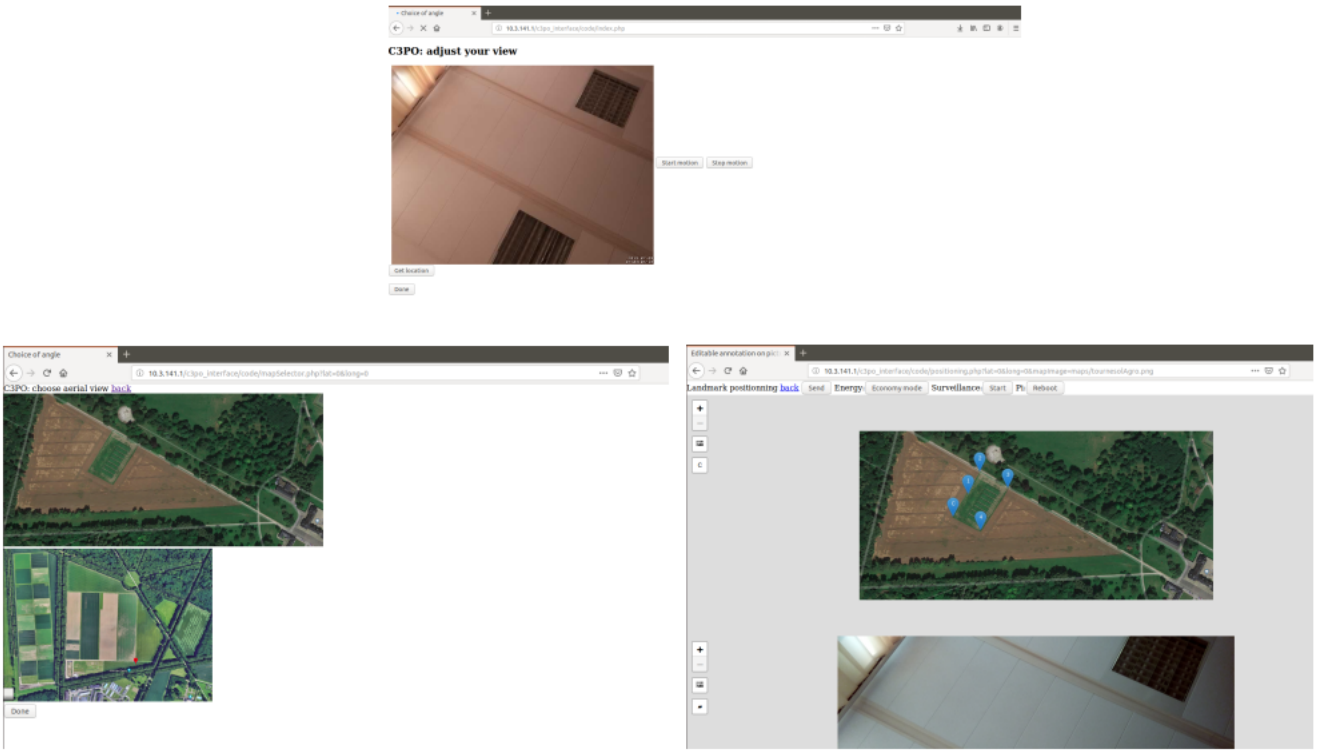


FIGURE 3.5 – Pages de l'interface de commande par wifi du raspberry

Plusieurs boutons sont présents dans la dernière page de l'interface et permettent de :

- Choisir la durée de capture
- Activer le mode economie d'énérgie (désactive les entrées USB, Ethernet,...)
- Lancer la boucle d'acquisition des images
- Arrêter la prise d'image
- Redémarrer le Raspberry Pi

3.2 Annotation des images

Assigner des labels aux images est une étape importante dans le projet. Nous avons fait le choix d'utiliser «Labelimg» (3.7) une application implémentée avec Python et qui permet de :

- Créer un réctangle autour d'un élément identifié

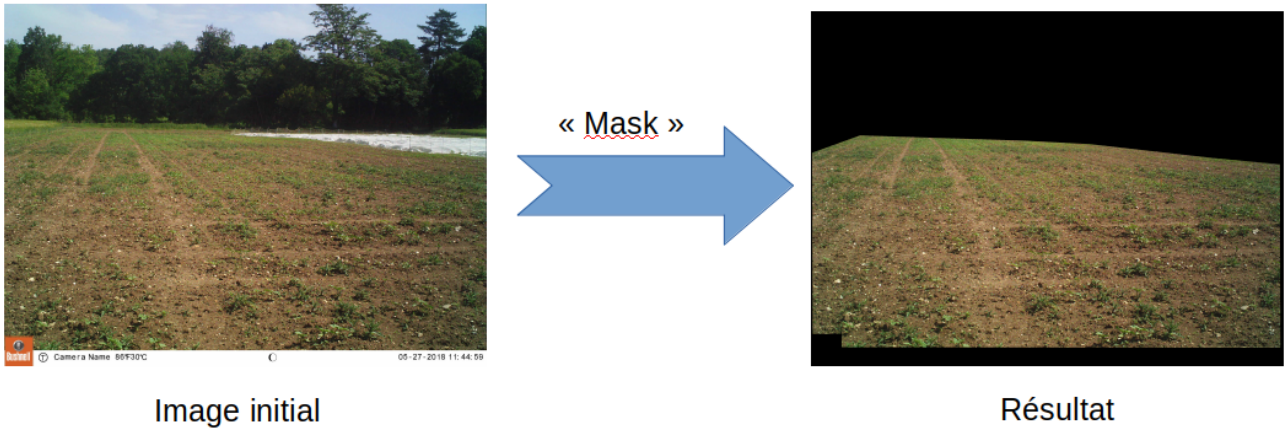


FIGURE 3.6 – Application du masque sur l'image

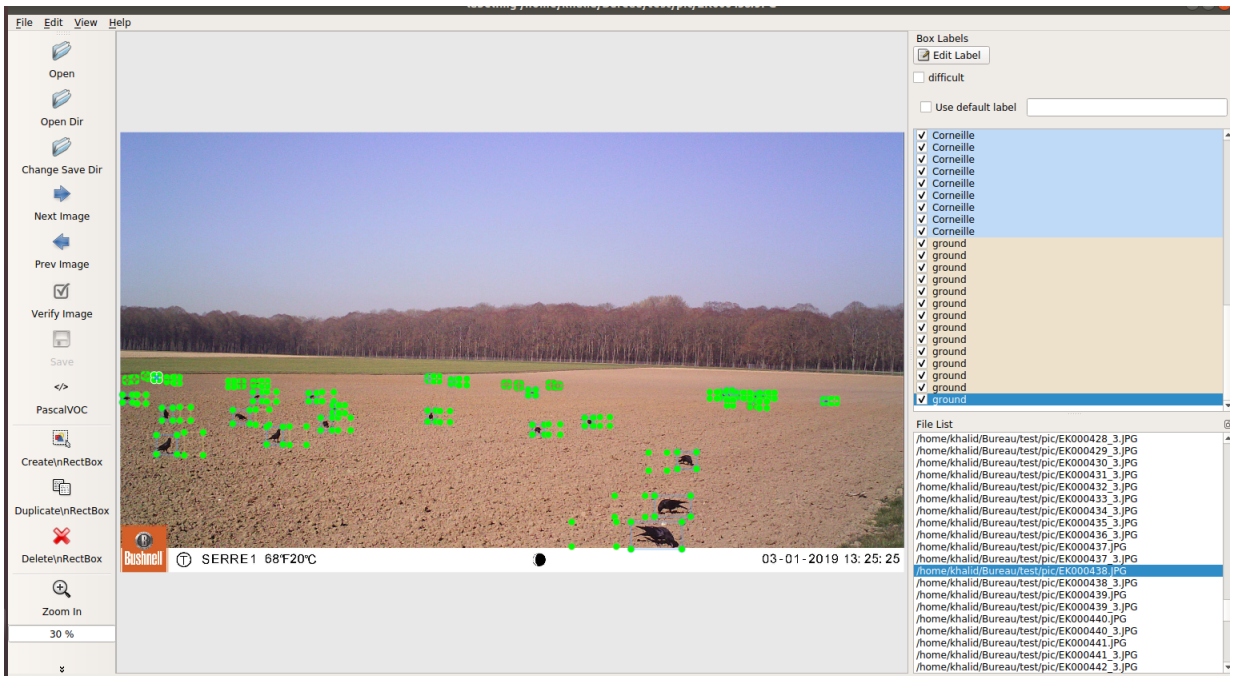


FIGURE 3.7 – Interface du logiciel LabelImg

- Donner un label à ce rectangle

Une telle annotation sur des images haute définition et avec une profondeur de champ telle que certains oiseaux peuvent apparaître très petits, prend beaucoup de temps par image. Il n'est donc pas possible d'annoter de cette manière la totalité des images acquises (des dizaines de milliers, en accroissement constant). Ainsi, nous avons décidé d'automatiser cette procédure comme nous le verrons dans la partie "Annotation semi-automatique" (5).

3.3 Différences entre deux images

3.3.1 Méthodes d'identification de différences entre images

Afin de trouver les différences nous avons utilisé plusieurs méthodes :

1. SSIM : fonction Compare-SSIM du package OpenCV spécialisé dans le traitement d'image.
2. Diff : différence entre deux images dans le sens mathématiques (imageA-imageB), en RGB
3. Absdiff : différence entre deux images avec la valeur absolue (|imageA-imageB|), en RGB
4. HSVabsdiff : procédure simple détectant les différences en termes de Teinte, Saturation et Luminance.

Le script que nous avons écrit permet de repérer les différences par l'un de ces algorithmes au choix et de dessiner un rectangle autour (3.8), ce qui constitue une imagette qui devra par la suite être analysée par le réseau de neurones.

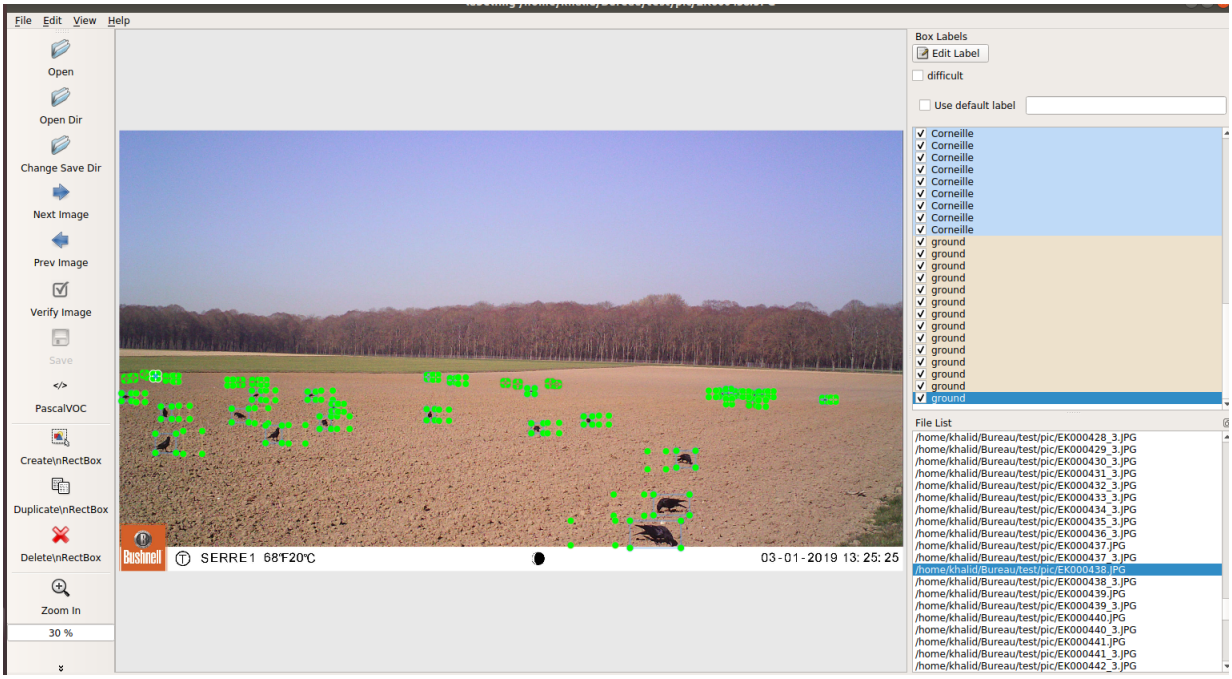


FIGURE 3.8 – Interface du logiciel LabelImg

En revanche nous pouvons remarquer que l'algorithme a tendance à sélectionner un très grand nombre de différences à cause des changements de luminosité. Le nombre d'imagettes correspondant à ces différences est trop grand pour pouvoir être analysé en temps réel par un réseau de neurones profond. Pour cette raison nous avons ajouté des filtres sur la taille des rectangles.

3.3.2 Filtres de différences pertinentes

Filtre par régions d'images

Pour cette méthode nous avons découpé l'image en trois régions selon la position de la cible sur l'image :

1. Hauteur maximale de l'imagette $<$ 1200 pixels
2. 1800 $>$ Hauteur maximale de l'imagette $>$ 1200 pixels
3. Hauteur maximale de l'imagette $>$ 1800 pixels

La taille du rectangle dessiné autour de la cible sera filtrée selon les régions citées ci-dessus. Par exemple, si la cible est dans la région 3, la taille du rectangle sera assez grande. Si elle est dans la région 1, elle sera au fond de l'image et par conséquent la taille du rectangle sera trop petite.

Filtre avec un modèle *Random Forest*(4.3.1)

Nous avons utilisé les images annotées manuellement pour construire une table de données «Image-labels» :

	0	1	2	3	4	5	6	7
0	filename	width	height	classe	xmin	ymin	xmax	ymax
1	EK000455_3.…	3264	2448	0	1310	1557	1649	1705
2	EK000455_3.…	3264	2448	1	932	1555	1277	1702
3	EK000550.JPG	3264	1832	0	699	1350	828	1459
4	EK000550.JPG	3264	1832	0	2917	1435	3093	1573
5	EK000550.JPG	3264	1832	0	1746	1223	1831	1300
6	EK000550.JPG	3264	1832	0	1705	1259	1787	1335
7	EK000550.JPG	3264	1832	0	1534	1070	1555	1089
8	EK000550.JPG	3264	1832	0	2755	1089	2780	1107
9	EK000549.JPG	3264	1832	0	364	1467	496	1591
10	EK000549.JPG	3264	1832	0	693	1335	840	1462
11	EK000549.JPG	3264	1832	0	469	1506	643	1653
12	EK000549.JPG	3264	1832	0	2493	1226	2567	1312
13	EK000549.JPG	3264	1832	0	2011	1256	2090	1314
14	EK000549.JPG	3264	1832	0	27	1066	55	1094

FIGURE 3.9 – Représentation de la table de données «Image-labels»

Cette table (3.9) regroupe les données de 1110 images et sert à entrainer un modèle de *Random Forest* sur les tailles d'imagettes susceptibles de comporter des oiseaux en fonction de la taille de l'imagette et de sa position dans l'image.

```
# Rapport de classification
print(classification_report(TestY, y_pred, target_names=le.classes_))
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(TestY, y_pred))
```

```
              precision    recall  f1-score   support

      Oiseau       0.76      1.00      0.86        16
   PasOiseau       1.00      0.78      0.88        23

   micro avg       0.87      0.87      0.87        39
   macro avg       0.88      0.89      0.87        39
weighted avg       0.90      0.87      0.87        39

Accuracy: 0.8717948717948718
```

```
col_0   0   P
row_0
0      16   0
P       5  18
```

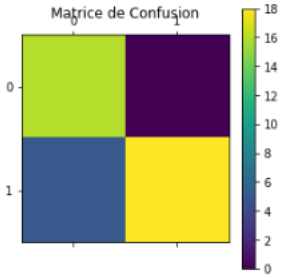


FIGURE 3.10 – La Matrice de confusion, et Le Score du modèle

```
# Coéf d'accord KAPPA
kappa = metrics.cohen_kappa_score(TestY, y_pred, labels=None, weights=None, sample_weight=None)
print("KAPPA:",kappa)
```

```
KAPPA: 0.7470817120622568
```

FIGURE 3.11 – Le Score *KAPPA* du modèle

On peut voir dans la figure (3.10) que la précision du modèle atteint 87,17%. De plus, pour vérifier le taux d'accord ou de «concordance» nous avons calculé le coefficient *KAPPA*[20]

Le score est dans l'intervalle [0.61; 0.80], donc nous avons un accord fort mais surtout la précision pour identifier les imagettes qui ne sont pas des oiseaux est de 1 ce qui en fait un filtre d'élimination fiable.

Tableau comparatif

Le tableau suivant (3.12) permet de comparer les différents filtres en utilisant la méthode SSIM. 65 images sont soumises aux différents filtres. Les images ont étées annotées manuellement et par conséquent, nous savons d'avance qu'elles contiennent exactement 110 «imagettes».

Filtre	Nombre d'image	Temps de calcul	Nombre d'imagettes	Nombre total d'oiseaux	Nombre d'oiseaux retenu	Nombre d'oiseaux perdu	Score KAPPA
Région d'image	65	7 min 4 s	1692	110	91	19	65,32 %
Random Forest	65	8 min 31 s	1214	110	104	6	73,80 %
No Filtre	65	26 min 27 s	4240	110	110	0	

FIGURE 3.12 – Tableau comparatif des différents filtres sur un jeu de données de 65 images contenant 110 «imagettes» de "Oiseaux"

On peut voir dans le tableau (3.12) que le filtre avec le modèle *Random Forest* fournit le meilleur résultat en termes de nombre d'imagettes enregistrées : sur 110 Oiseaux présents sur les images intiales il n'a pas réussi à identifier 6 Oiseaux ce qui représente 5% de perte. Le filtre utilisant les régions d'images en perd 17,27%. De plus, le score *KAPPA* du *Random Forest* est nettement supérieur à celui des régions.

En revanche, en termes de temps de calcul, le filtre avec régions d'images est meilleur par rapport au deux autres.

3.4 Identification des imagettes

3.4.1 Structure des *convolutional neural networks* (CNN)

Le nombre total d'imagettes obtenu pour le moment est de 1110 dont 521 représentant la classe "Pas Oiseau" et 589 la classe "Oiseau". Ces images ont été annotées manuellement avec "Labelimg" lors de leur acquisition. Notre rôle est d'entraîner un modèle de machine learning sur le jeu d'entraînement, puis d'utiliser ce modèle pour prédire les comportements sur les images de test.

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais *Convolutional Neural Network*, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

La première partie d'un CNN est la partie convolutive à proprement parler (la partie gauche de la figure (3.13). Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de couches, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions.

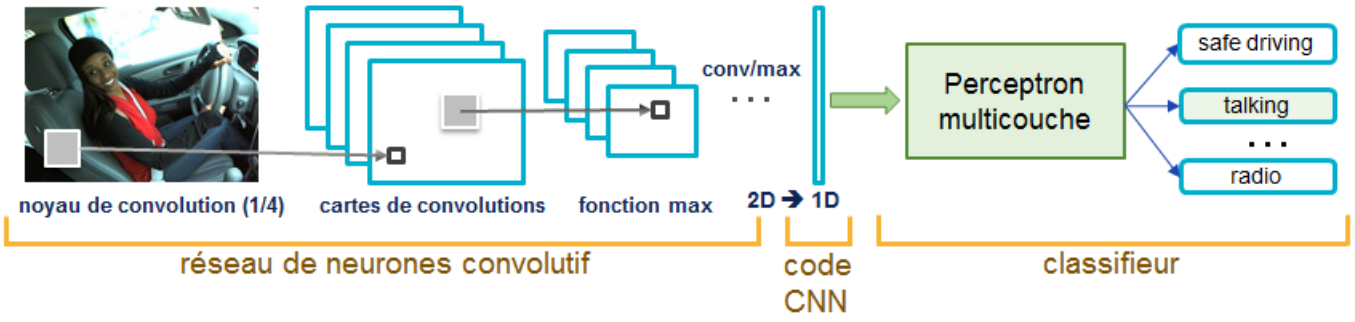


FIGURE 3.13 – Figure représentant les deux parties d'un CNN[21]

Ce code CNN en sortie de la partie convolutive est ensuite transmis à une deuxième partie(la partie droite de la figure (3.13), constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

Cependant créer un nouveau réseau de neurones convolutif est coûteux en termes d'expertise, de matériel et de quantité de données annotées nécessaires.

La complexité de la création de CNN et de l'entraînement des CNN peut être évitée en adaptant des réseaux pré-entraînés disponibles publiquement. Ces techniques sont appelées "transfert learning", car on exploite la connaissance acquise sur un problème de classification général pour l'appliquer à un problème particulier.

La “connaissance” sur la classification d’images contenue dans un tel réseau peut être exploitée de deux façons :

- ☐ Comme un extracteur automatique de caractéristiques des images, matérialisé par le code CNN, et ré-exploitée avec un autre classifieur.(3.14)
- ☐ Comme une initialisation du modèle, qui est ensuite ré-entraîné plus finement (*Fine Tuning*) pour traiter le nouveau problème de classification.(3.14)

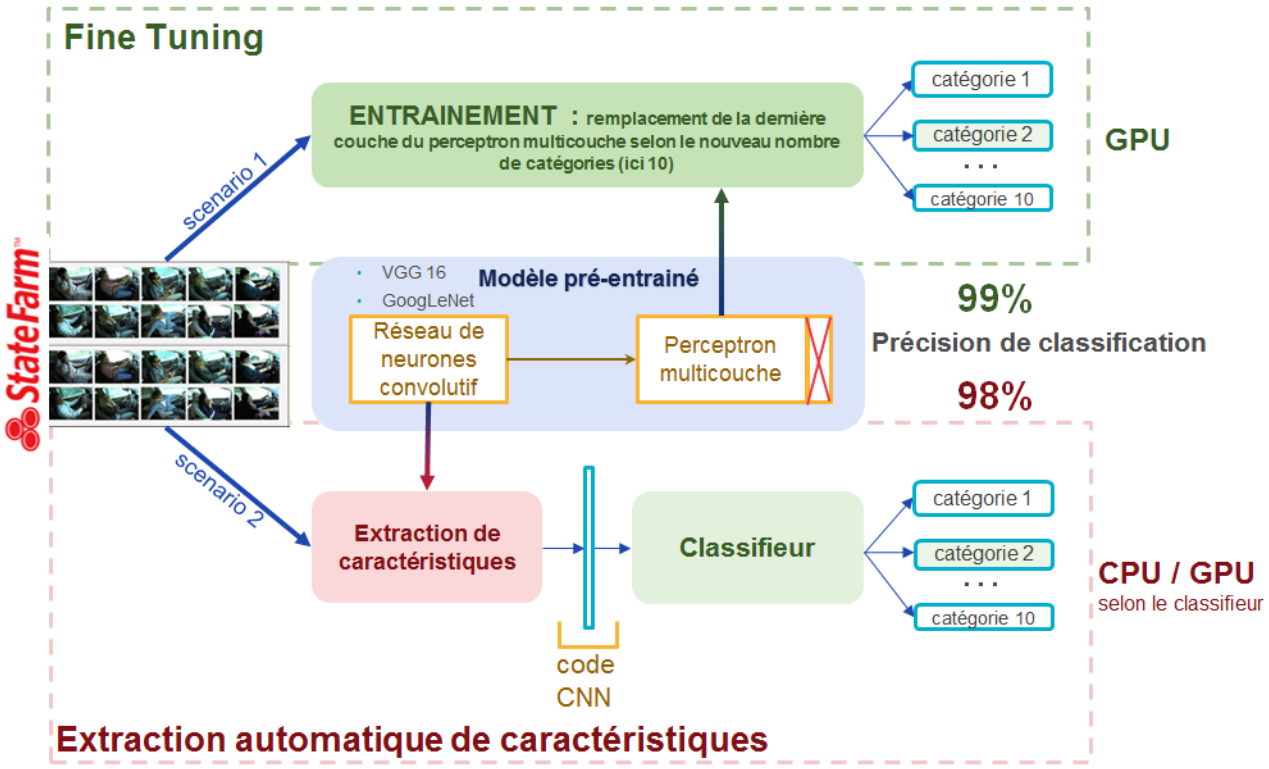


FIGURE 3.14 – Illustration des deux méthodes pour exploiter un CNN[22]

Dans notre projet nous avons choisi d’utiliser la deuxième méthode avec la structure VGG16, une version du réseau de neurones convolutif très connu appelé VGG-Net[23].

3.4.2 Spécificités de la structure VGG16

VGG16 est un réseau Covnet à 16 couches utilisé par le groupe de géométrie visuelle (VGG) de l’Université d’Oxford dans le cadre du concours ILSVRC (ImageNet) en 2014. Le modèle atteint un taux de succès de 92,5% dans le top 5 sur l’ensemble de validation[24].

Il prend en entrée une image en couleurs de taille 224*224 px et la classifie dans une des 1000 classes. Il renvoie donc un vecteur de taille 1000, qui contient les probabilités d’appartenance à chacune des classes.

L’architecture de VGG-16 est illustrée par les schémas ci-dessous (3.15) :

FIGURE 3.15 – Architecture de VGG-16[25]

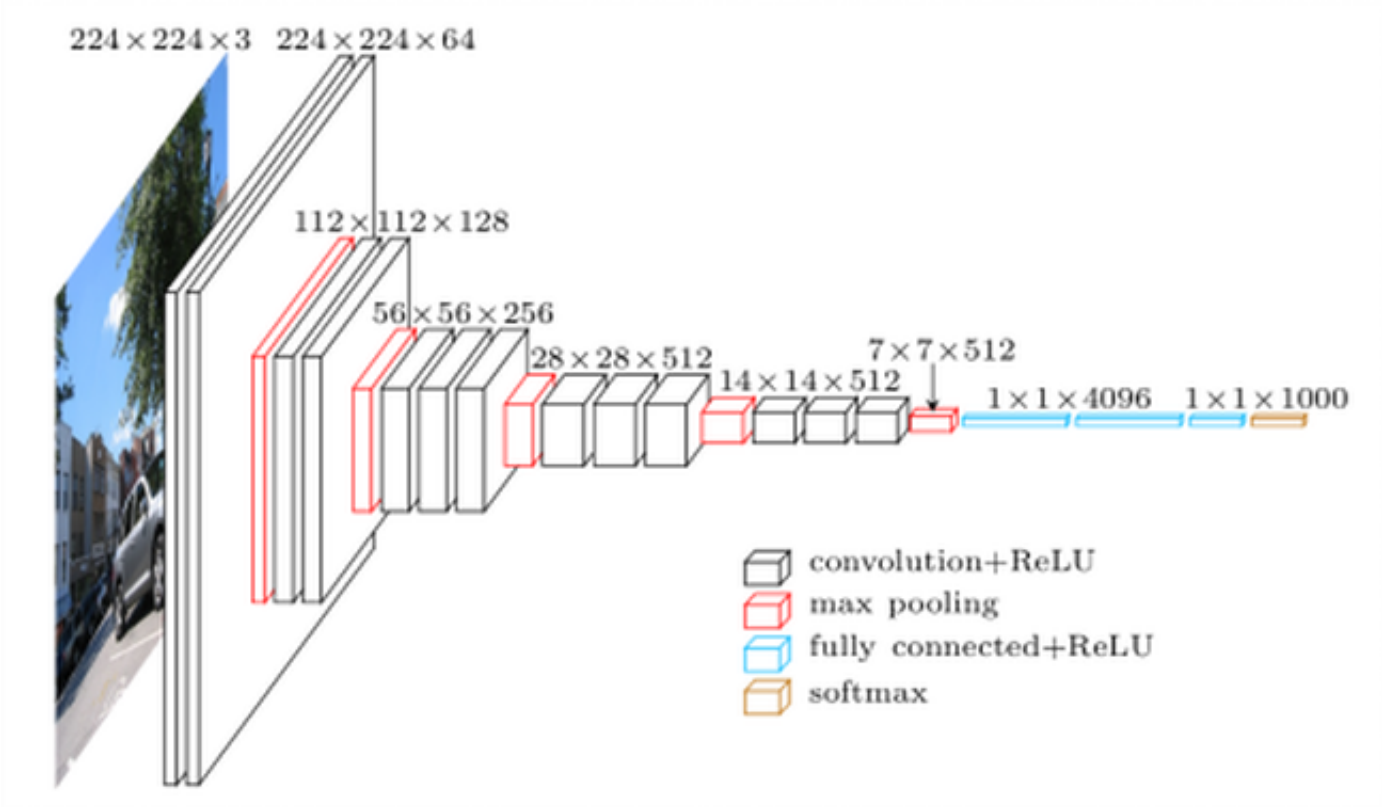


FIGURE 3.16 – Représentation 3D de l'architecture de VGG-16[26]

Il s'agit actuellement du choix le plus populaire pour l'extraction de fonctionnalités à partir d'images [27]. La configuration de poids du VGGNet est disponible publiquement et a été utilisée dans de nombreuses autres applications et défis en tant qu'extracteur de fonctionnalités de base.

3.4.3 Extraction automatique de caractéristiques avec *VGG16*

L'extraction automatique de caractéristiques exploite uniquement la partie convolutive d'un réseau pré-entraîné. Elle l'utilise comme extracteur de caractéristiques des images, pour alimenter le classifieur de votre choix.

En pratique, le VGG 16 est tronqué pour ne garder que la partie convolutive. Cette partie est dite gelée, pour exprimer l'absence d'entraînement. Ce réseau prend en entrée une image au

bon format (224, 224, 3) et produit en sortie le code CNN. Chaque image du dataset est ainsi transformée en un vecteur de caractéristiques, qui est utilisé pour entraîner un nouveau classifieur.

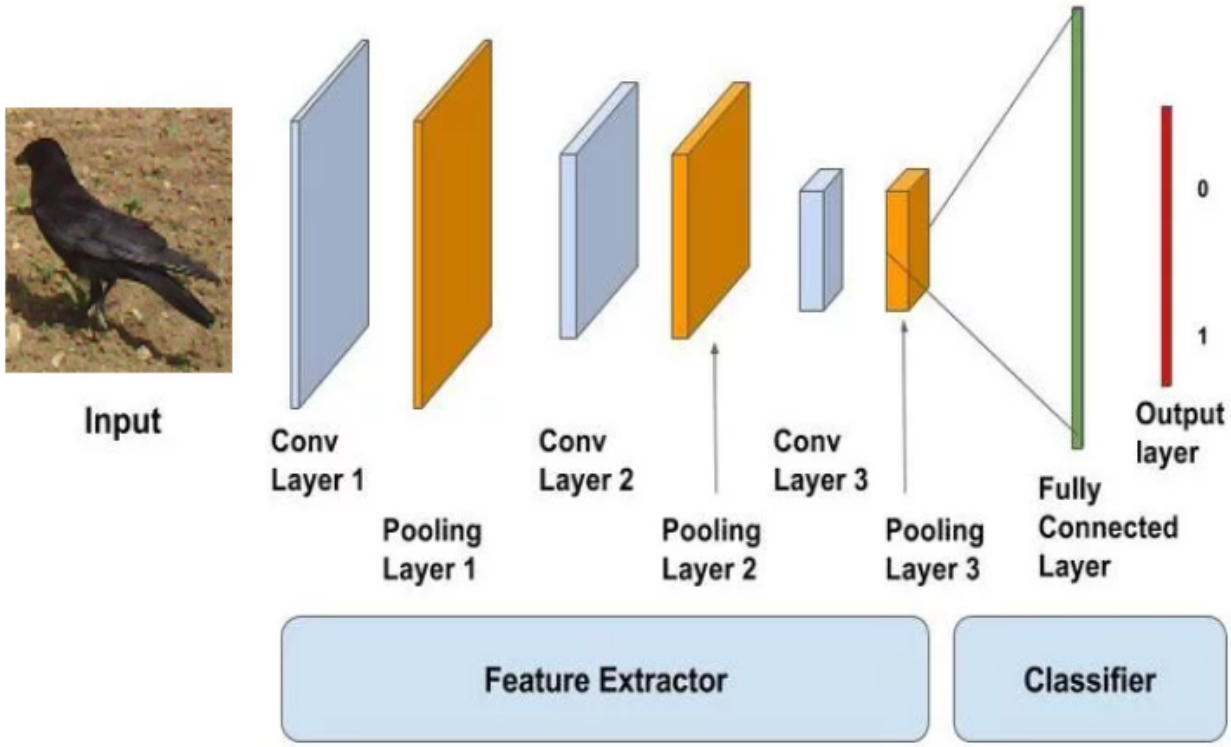


FIGURE 3.17 – Représentation de l'extraction automatique de caractéristiques avec *VGG-16* [28]

Chapitre 4

Modèles et résultats

On entraîne plusieurs classifieurs Régression logistique, Arbre de décision, *Random Forest* à partir des codes CNN de VGG 16 sur des échantillons d'entraînement de taille croissante.

La figure suivante illustre la table de données utilisée pour l'entrainement des différents modèles (sur la base des caractéristiques extraites par VGG-16).

Out[54]:

	Name	Label	2	3	4	5	6	7	8	9	...	25080	25081	25082	25083	25084	25085	25086	25087	25088	25089
	000039_Oiseau_765.JPG	0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	5.013870	-0.0	-0.0	-0.000000	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
	000439_Oiseau_813.JPG	0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	4.387150	-0.0	-0.0	-0.000000	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
	51_PasOiseau_1109.JPG	1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	4.586602	-0.0	-0.0	0.377037	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
	0314_3_Oiseau_896.JPG	0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	1.873481	-0.0	-0.0	-0.000000	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
	552_PasOiseau_177.JPG	1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	6.150038	-0.0	-0.0	-0.000000	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0

columns

FIGURE 4.1 – Représentation de la table de données finale

Description de la table :

- "Name" : Le nom des images
- Les variables "2" ; ... ;"25091" : des variables quantitatifs représentants les "Features" ou caractéristiques extraites des images avec VGG16
- "Label" (4.2) :
  - 0 = Oiseau
  - 1 = PasOiseau

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe8192b5128>
```

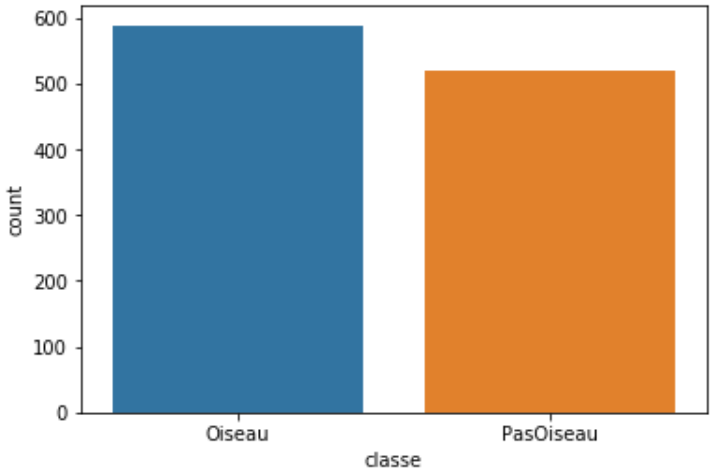


FIGURE 4.2 – Barplot de la variable "Label"

4.1 *VGG16* & Régression logistique

4.1.1 Modèle

La régression logistique[29] est une méthode statistique d'analyse d'un ensemble de données dans laquelle une ou plusieurs variables indépendantes déterminent un résultat. Le résultat est mesuré avec une variable dichotomique.

La régression logistique a pour objectif de trouver le modèle le mieux adapté pour décrire la relation entre la caractéristique d'intérêt dichotomique et un ensemble de variables indépendantes (prédicteur ou explicatif). La régression logistique génère les coefficients d'une formule permettant de prédire une transformation *logit* de la probabilité de présence de la caractéristique d'intérêt

$$Logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_kX_k$$

où p est la probabilité de présence de la caractéristique d'intérêt. La transformation *logit* est définie comme la cote enregistrée :

$$odds = \frac{p}{1-p} = \frac{Probabilité\ de\ la\ présence\ du\ caractéristique}{Probabilité\ de\ l'absence\ du\ caractéristique}$$

et

$$logit(p) = ln(\frac{p}{1-p})$$

Plutôt que de choisir des paramètres qui minimisent la somme des erreurs au carré (comme dans la régression ordinaire), l'estimation dans la régression logistique choisit des paramètres qui maximisent la probabilité d'observer les valeurs d'échantillon.

4.1.2 Résultats

Nous obtenons les sorties ci-dessous (4.3) en utilisant la fonction du logiciel *Python* : *LogisticRegression* de la librairie *Sklearn* :

```
# Rapport de classification
print(classification_report(testY, preds, target_names=le.classes_))
# Score
print("Accuracy", model_logit.score(testX, testY), sep = " : ")
```

```
              precision    recall  f1-score   support

      Oiseau       0.93      0.84      0.88        50
   PasOiseau       0.85      0.94      0.90        50

   micro avg       0.89      0.89      0.89       100
   macro avg       0.89      0.89      0.89       100
weighted avg       0.89      0.89      0.89       100

Accuracy : 0.89
```

```
col_0   0   1
row_0
0      42   8
1       3  47
```

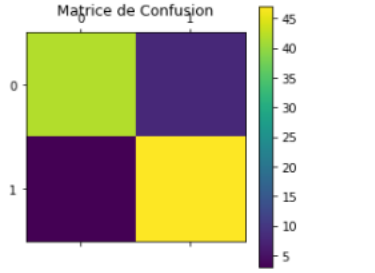
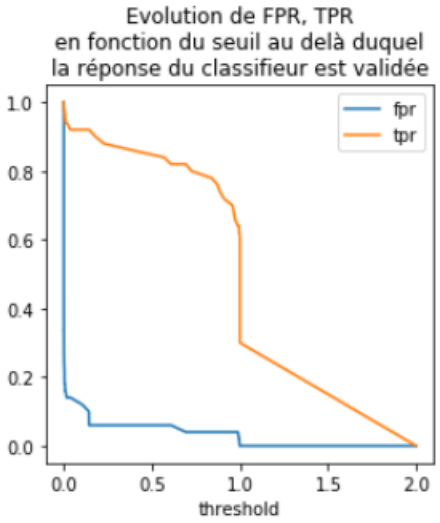


FIGURE 4.3 – Rapport de classification et Matrice de confusion pour le modèle de régression logistique

Après entraînement du modèle sur 1010 images, nous obtenons une précision de 89% sur un jeu de test de 100 images.



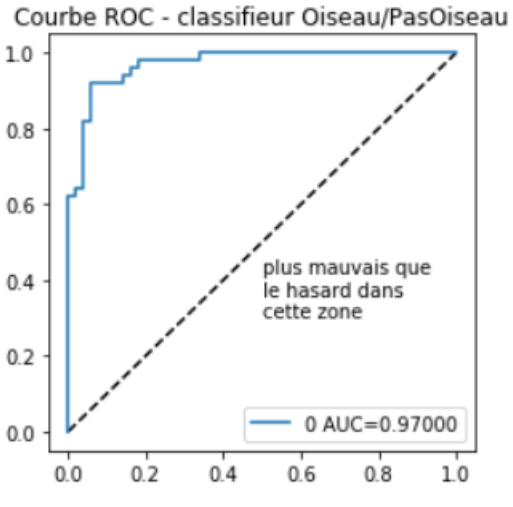


FIGURE 4.4 – Représentation de la courbe *ROC*, et de l'evolution des FPR, TPR en fonction du seuil de validité de la réponse

- *FPR* : Taux de faux positif (*False Positive Rate*)
- *TPR* : Taux de vrai positif (*True Positive Rate*)

Le score de la mesure *AUC*[30] de 97% que nous obtenons (4.4) est plutôt très bon.

4.2 VGG16 & Arbre de décision

4.2.1 Modèle

Les arbres de décision[31] sont une méthode d'apprentissage supervisé non paramétrique utilisée pour la classification et la régression. L'objectif est de créer un modèle qui prédit la valeur d'une variable cible en apprenant des règles de décision simples déduites des entités de données.

Les arbres binaires de décision (*CART : classification and regression trees*) s'appliquent à tous types de variables. La complexité du modèle est gérée par deux paramètres : *max_depth*, qui détermine le nombre max de feuilles dans l'arbre,et le nombre minimales *min_samples_split* d'observations requises pour rechercher une dichotomie. Nous avons optimisé ces paramètres dans notre étude avec la fonction *Python* : *GridSearchCV* (Validation Croisée) de la librairie *Sklearn*.

Une fois l'arbre construit, classer un nouveau candidat se fait par une descente dans l'arbre, depuis la racine aboutissant à l'une des feuilles (qui indique la décision ou la classe). A chaque niveau de la descente on passe un nœud intermédiaire où une variable x_i est testée pour décider du chemin (ou sous-arbre) à choisir pour continuer la descente.

4.2.2 Résultat

Nous obtenons les sorties ci-dessous (4.5) en utilisant la fonction du logiciel *Python* : *DecisionTreeClassifier* de la librairie *Sklearn* :

```
# Rapport de classification
print(classification_report(testY, y_pred, target_names=le.classes_))
# Accuracy
print("Accuracy : ",tree.score(testX,testY))
```

```
              precision    recall  f1-score   support

      Oiseau       0.84      0.62      0.71        50
   PasOiseau       0.70      0.88      0.78        50

   micro avg       0.75      0.75      0.75       100
   macro avg       0.77      0.75      0.75       100
weighted avg       0.77      0.75      0.75       100

Accuracy :  0.75
```

```
col_0   0   1
row_0
0      31  19
1       6  44
```

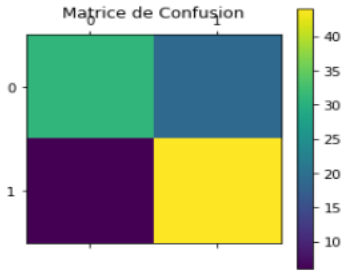


FIGURE 4.5 – Rapport de classification et Matrice de confusion pour le modèle Arbre de décision

Après entraînement du modèle sur 1010 images, nous obtenons une précision de 75% (4.5) sur un jeu de test de 100 images.

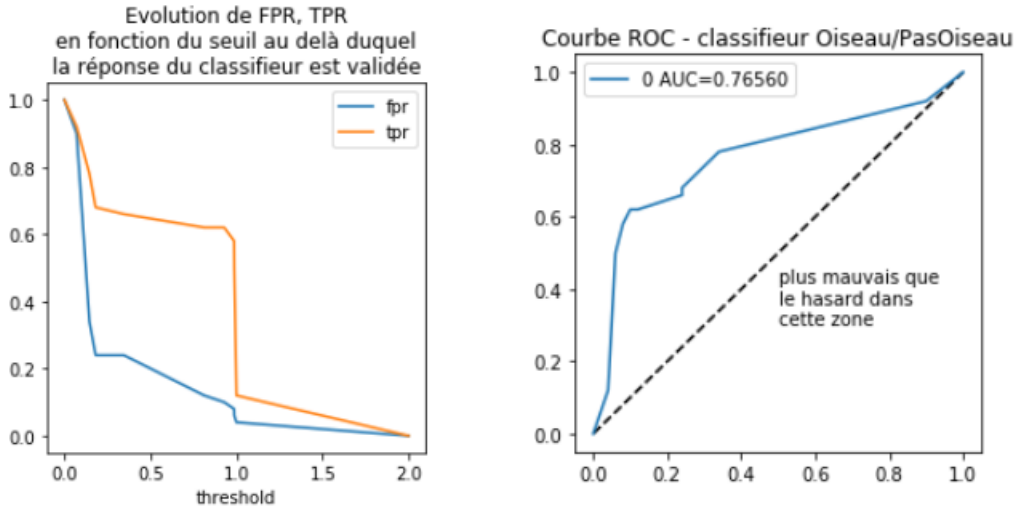


FIGURE 4.6 – Représentation de la courbe *ROC*, et de l'evolution des FPR, TPR en fonction du seuil de validité de la réponse

On peut voir sur la courbe *ROC*[32] (4.6) quelques fluctuations qui ont tendance à s'approcher de la diagonale, cela confirme la précision moyenne obtenue au-dessus. Avec une *AUC* égale à $76,56\%$ ce modèle est beaucoup moins précis que le précédent.

4.3 VGG16 & RandomForest

4.3.1 Modèle

L'algorithme du *Random Forest*[33] appartient à la famille des agrégations de modèles, c'est en fait un cas particulier de *bagging (bootstrap aggregating)* appliqué aux arbres de décision de type CART*classification and regression trees*. Le principe des méthodes de Bagging, et donc en particulier des *Random Forest* c'est de faire la moyenne des prévisions de plusieurs modèles indépendants pour réduire la variance et donc l'erreur de prévision. Pour construire ces différents modèles, on sélectionne plusieurs échantillons bootstrap, c'est-à-dire des tirages avec remises.

En plus du principe de *Bagging*, les *Random Forest* ajoutent de l'aléa au niveau des variables. Pour chaque arbre on sélectionne un échantillon bootstrat d'individus et à chaque étape, la construction d'un nœud de l'arbre se fait sur un sous-ensemble de variables tirées aléatoirement.

On se retrouve avec plusieurs arbres et donc des prédictions différentes pour chaque individu. Cependant l'estimation finale est obtenue :

Dans le cas d'une classification : on choisit la catégorie la plus fréquente

Dans le cas d'une régression : on fait la moyenne des valeurs prédites

Dans notre cas d'étude, on va appliquer l'algorithme de *Random Forest* pour créer un puissant modèle prédictif. Cet algorithme fournit de bonnes performances en prédiction, et ne crée pas de problème d'*overfitting* si l'ensemble de ses hyperparamètres sont bien ajustés. Nous utilisons la même fonction que pour l'Arbre de Décision (*GridSearchCV*) pour trouver les hyperparamètres optimaux afin d'obtenir le meilleur modèle.

4.3.2 Résultat

Nous obtenons les sorties ci-dessous (4.7) en utilisant la fonction du logiciel *Python* : *RandomForestClassifier* de la librairie *Sklearn* :

```
# Rapport de classification
print(classification_report(testY, y_pred, target_names=le.classes_))
# Accuracy
print("Accuracy : ",RandomForest.score(testX,testY))
```

```
              precision    recall  f1-score   support

      Oiseau       0.93      0.76      0.84        50
   PasOiseau       0.80      0.94      0.86        50

   micro avg       0.85      0.85      0.85       100
   macro avg       0.86      0.85      0.85       100
weighted avg       0.86      0.85      0.85       100

Accuracy :  0.85
```

```
col_0   0   1
row_0
0      38  12
1       3  47
```

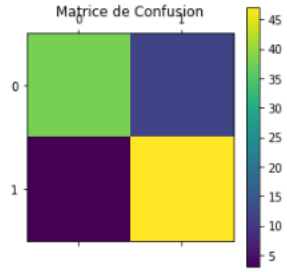
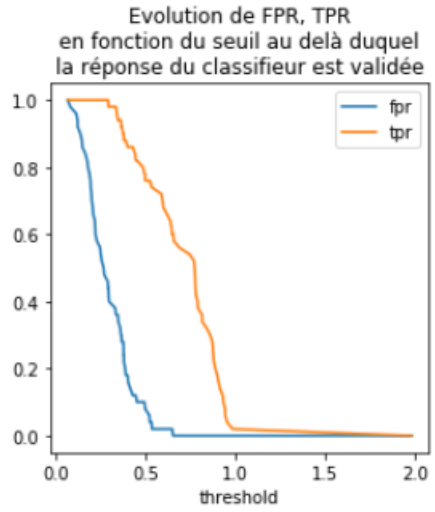


FIGURE 4.7 – Rapport de classification et Matrice de confusion pour le modèle Arbre de décision

Après entraînement du modèle sur 1010 images, nous obtenons une précision de 85% (4.7) sur un jeu de test de 100 images.



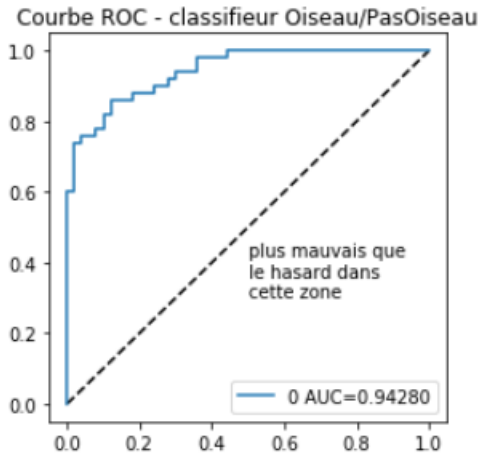


FIGURE 4.8 – Représentation de la courbe *ROC*, et de l'evolution des FPR, TPR en fonction du seuil de validité de la réponse

On peut voir là aussi la présence de quelques fluctuations (4.8), mais avec une *AUC* de l'ordre de 94,28% ce modèle reste intéressant.

```
# Coéficcient KAPPA du test de Cohen (accord)
kappa = metrics.cohen_kappa_score(testY, y_pred, labels=None, weights=None, sample_weight=None)
print("KAPPA:",kappa)
```

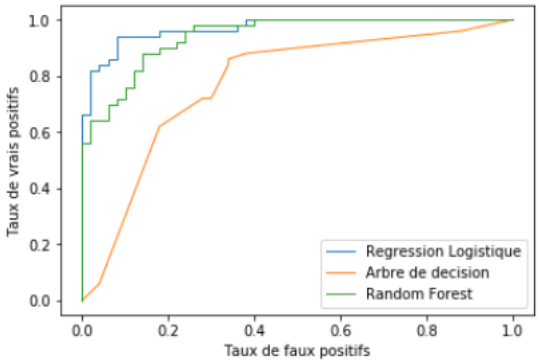
```
KAPPA: 0.7
```

FIGURE 4.9 – Test statistique de *KAPPA*

On obtient un coefficient de *KAPPA* de l'ordre de 0,7 (4.9) pour notre modèle *Random Forest* cela représente un accord fort entre les prédictions issues du modèle et les labels déjà connus du jeu de données fournit pour le test.

4.4 Comparaison des résultats

Dans toute méthode, la prévision de dépassement, ou non, est associée au choix d'un seuil qui est par défaut 0.5. L'optimisaiton de ce seuil dépend des coûts respectifs associés aux faux positifs et aux faux négatifs qui ne sont pas nécessairement égaux. La courbe *ROC* (4.10) permet de représenter l'influence de ce seuil sur les taux de faux positifs et vrais positifs.




```
In [146]: dataframeErreur.mean()

Out[146]: Regression Logistique    0.11
          Arbre de decision        0.25
          Random Forest            0.15
          dtype: float64
```

FIGURE 4.10 – Courbe *ROC* et pourcentage d'erreur pour les trois modèles

Le modèle de régression logistique est le plus performant par rapport aux autres avec un pourcentage d'erreur de 11%, suivi du modèle Random Forest avec 15% et enfin le modèle avec l'arbre de décision avec 25%.

Chapitre 5

Annotation semi-automatique

L'objectif de l'annotation semi-automatique est de minimiser le travail manuel, et de constituer une grande base de données très rapidement. Nous avons utilisé deux méthodes :

1. Avec un filtre sur les régions d'image
2. En utilisant un modèle *Random Forest*

Les probabilités issues de ce modèle seront utilisées de la manière suivante :

1. Si la probabilité de prédiction est supérieure ou égale à 80%, l'image sera annotée par le modèle.
2. Si la probabilité de prédiction est inférieure à 90%, une interface *Python* (5.1) dédiée permettra à l'utilisateur d'annoter comme on peut le voir sur la figure suivante.

Par ailleurs, un tirage aléatoire des images annotées automatiquement par le modèle, sélectionnés via un tirage aléatoire, sont présentés à l'annotateur pour évaluer la précision de l'annotation automatique et s'assurer que la qualité continue de s'améliorer.

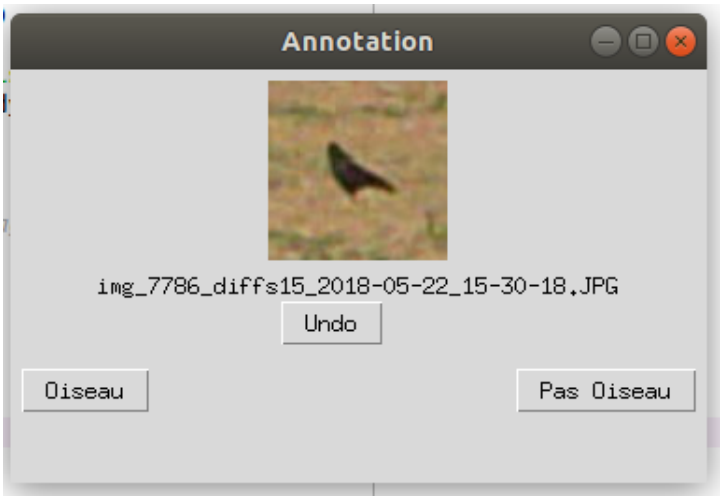


FIGURE 5.1 – Représentation de l'interface d'annotation semi-automatique conçu avec *Python*

Cette interface d'annotation sera reliée directement au processus d'acquisition et annotation des images, dans la partie «annotation par l'utilisateur» (5.2).

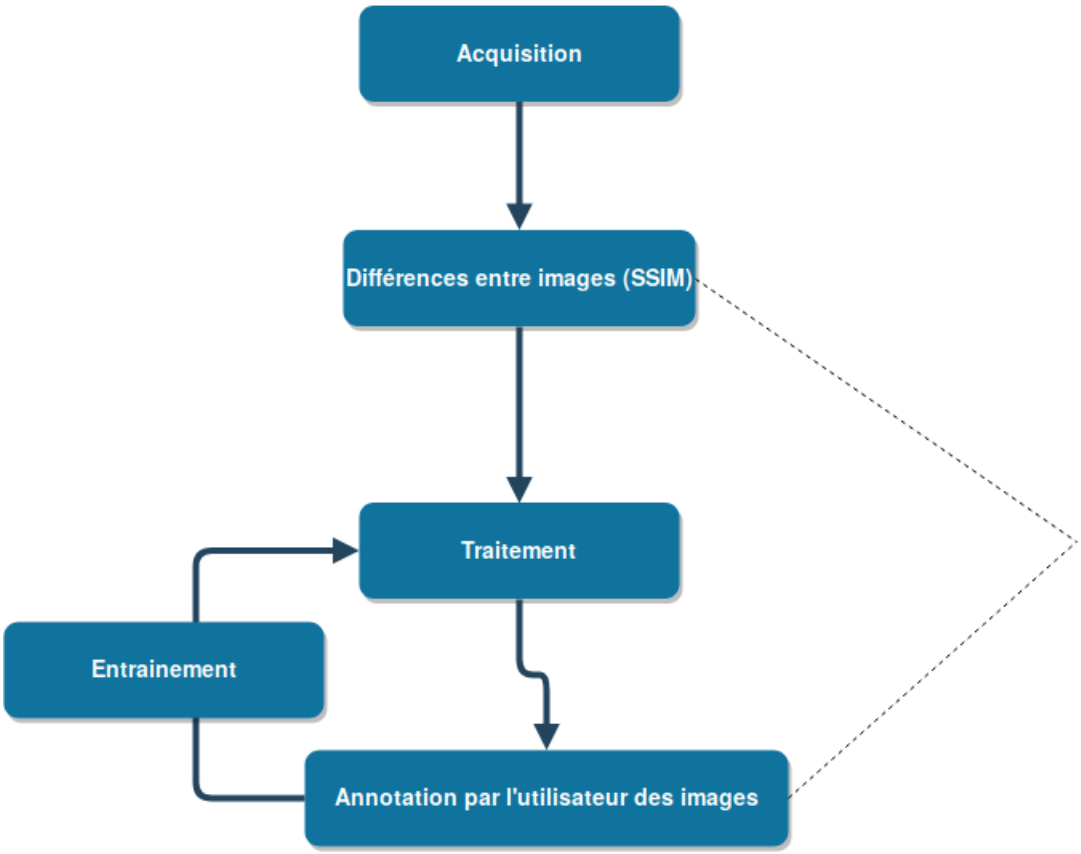


FIGURE 5.2 – Schéma représentant les différentes étapes d'acquisition et d'annotation des images

Chapitre 6

Conclusion

Tout d'abord, la partie acquisition des images a permis de mettre en place un dispositif assez maniable et facilement transportable dans les parcelles. Il reste encore des améliorations à apporter au dispositif mais, le travail éffectué sur l'interface qui le gère, ainsi que sur les scripts assurants le fonctionnement du *Raspberry Pi*, permet déjà d'avoir un outil fonctionnel pour accomplir cette étape importante d'acquisition de données.

L'extraction automatique de caractéristiques avec *VGG16* suivi par entrainement/test d'un classifieur a permis d'abord de contourner la création d'un nouveau *CNN*, ensuite de travailler avec trois modèles. Le modèle avec la régression logistique apparaît comme le meilleur avec une précision de 87%, en deuxième position le modèle avec *Random Forest* avec 85% de précision et enfin le modèle avec l'arbre de décision avec 75%. D'autres modèles sont envisagés dans la suite du projet (*K-NN[34], Lasso[35], BasicNet[36]*).

En associant, le modèle de prédiction obtenu et les filtres de différences entre images, nous avons automatisé la procédure d'annotation des images à l'aide d'une interface *Python*. Cela permet une annotation plus facile et beaucoup plus rapide, ainsi qu' une augmentation continue . Toutefois certaines améliorations pourraient être apportées à la précision de ce processus et à son temps de calcul.

Enfin, le stage a permis d'avancer sur deux grands axes, le premier est de concevoire un outil de suivis des dégâts d'oiseaux au semis et le deuxième relier cet outil aux différents modèles réalisés pour un traitement en temps réel. Cela permettra de détecter, localiser et identifier des oiseaux déprédateurs se posant sur des parcelles juste semées.

Bibliographie

[1] *Enquête Terres Inovia sur les dégâts d'oiseaux et petitsgibiers*. URL : https://www.terresinovia.fr/documents/20126/156330/Rapport_r%5C%C3%5C%A9sultats-2018-d%5C%C3%5C%A9clarations-de-d%5C%C3%5C%A9g%5C%C3%5C%A2ts_oiseaux.pdf/91cbb837-3031-3640-2aac-dcc8accfee9a?t=1554128951682.

[2] *Info - Tournesol : un atout pour vos assolements*. URL : https://www.terre-net.fr/observatoire-technique-culturale/strategie-technique-culturale/article/le-tournesol-une-culture-rentable-217-133684.html.

[3] *Dégâts d'oiseaux dans les cultures Oléo-protéagineuses – Synthèse des résultats d'enquête*. URL : https://www.terresinovia.fr/documents/20126/156330/Degats_oisx_synth_enqu2018_TerresInovia.pdf/64ea9e57-cad7-faaf-2082-e08f2787a0fd?t=1554128972736.

[4] *Dispositifs de lutte physique contre les oiseaux*. URL : http://www.ecophytopic.fr/tr/m%5C%C3%5C%A9thodes-de-lutte/m%5C%C3%5C%A9thodes-physiques/dispositifs-de-lutte-physique-contre-les-oiseaux-0.

[5] *Reconnaissance faciale : 7 tendances à suivre*. URL : https://www.gemalto.com/france/gouv/biometrie/reconnaissance-faciale.

[6] *RECONNAISSANCE DES ÉCRITURES MANUSCRITES*. URL : http://www.archives-nationales.culture.gouv.fr/documents/10157/180590/TEKLIA_kermorvant3.compressed.pdf/13d57645-b5a7-4eae-bd77-61b07f702797.

[7] *Reconnaissance vocale : quand l'intelligence artificielle nous parle*. URL : https://fr.theepochtimes.com/reconnaissance-vocale-lintelligence-artificielle-parle-100042.html.

[8] *Empreintes digitales - Un bref historique des systèmes d'identification criminelle*. URL : https://www.gemalto.com/france/gouv/biometrie/empreintes-digitales-et-identification.

[9] *DETECTION ET CATEGORISATION D'OBJETS EN MOUVEMENT DANS UNE VIDEO*. URL : https://hal.archives-ouvertes.fr/tel-01075040/document.

[10] *Raspberry Pi*. URL : https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[11] *What is annotation in machine learning ?* URL : https://www.quora.com/What-is-annotation-in-machine-learning.

[12] *Advanced annotation tools in Deep Learning : training data for computer vision with Supervisely*. URL : https://hackernoon.com/%5C%EF%5C%B8%5C%8F-advanced-annotation-tools-in-deep-learning-training-data-for-computer-vision-with-supervisely-847f8699a9cb.

[13] *Introducing best-in-class image annotation tools for computer vision applications*. URL : https://hackernoon.com/introducing-best-in-class-image-annotation-tool-for-computer-vision-applications-276116cfc500.

[14] *Graphical image annotation tool*. URL : https://github.com/tzutalin/labelImg.

[15] *Dataturks Product Try Demo Pricing API Signup*. URL : https://dataturks.com/help/image-rectangle-bounding-box-annotation.php.

[16] *Le deep learning apprend deux fois plus vite à reconnaîtres les images*. URL : https://www.usinenouvelle.com/editorial/le-deep-learning-apprend-deux-fois-plus-vite-a-reconnaitres-les-images.N829080.

[17] *Deep Learning, le réseau à convolution*. URL : https://www.supinfo.com/articles/single/8037-deep-learning-reseau-convolution.

[18] *Pooling Layers*. URL : https://keras.io/layers/pooling/.

[19] *Pensee artificielle mobilenet*. URL : http://penseeartificielle.fr/mobilenet-reconnaissance-images-temps-reel-embarque/.

[20] *Le Coefficient Kappa*. URL : http://kappa.chez-alice.fr/Kappa_2juges_Def.htm.

[21] *Classification d'images : les réseaux de neurones convolutifs en toute simplicité*. URL : https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/.

[22] *Classification d'images*. URL : https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/.

[23] *CNN Architectures*. URL : https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.

[24] *Introduction en douceur au défi de reconnaissance visuelle à grande échelle d'ImageNet (ILSVRC)*. URL : https://si62.be/introduction-en-douceur-au-defi-de-reconnaissance-visuelle-a-grande-echelle-dimagenet-ilsvrc/.

[25] *Classez et segmentez des données visuelles*. URL : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec-keras.

[26] *Segmentez des données visuelles*. URL : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec-keras.

[27] *Classification d'images*. URL : https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/.

[28] *Transfer Learning using pre-trained models*. URL : https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/.

[29] *Comprendre la régression logistique*. URL : https://www.em-consulte.com/en/article/842576.

[30] *Classification : AUC*. URL : https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=fr.

[31] *Cours - Arbres de décision*. URL : http://cedric.cnam.fr/vertigo/Cours/ml2/coursArbresDecision.html.

[32] *Classification : ROC*. URL : https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=fr.

[33] *Techniques ensemblistes pour l'analyse prédictive*. URL : http://eric.univ-lyon2.fr/~ricco/cours/slides/bagging_boosting.pdf.

[34] *K Plus prochesvoisins*. URL : http://webia.lip6.fr/~rifqi/COURS2001-2002/IA/knn.pdf.

[35] *Régression régularisée*. URL : http://eric.univ-lyon2.fr/~ricco/cours/slides/regularized_regression.pdf.

[36] *Train your first neural network : basic classification*. URL : https://www.tensorflow.org/tutorials/keras/basic_classification.