Projet Fil Rouge

COMPTER, CARTOGRAPHIER ET CARACTÉRISER POUR MIEUX PRÉVENIR LES DÉGÂTS D'OISEAUX

IODAA - AMI2B

INRAE - AgroParisTech

Chaimae EL HOUJJAJI - Donatien LEGE - Anthony MOREAU - Pauline TURK - Dalyan VENTURA

25 Février 2022

Table des matières

1 Contexte

1.1 Problème étudié

TerresInovia est un institut technique agricole assurant des missions de recherche et développement en agriculture, dirigé par Laurent ROSSO et présidé par Gilles ROBILLARD. Elle est issue d'une fusion entre deux instituts, le CETIOM (Centre technique interprofessionnel des oléagineux métropolitain) et l'UNIP (Union nationale interprofessionnelle des plantes riches en protéine). Une des problématiques auxquelles sont confrontés les chercheurs est les dégâts causés par les corbeaux sur les semis. En effet, les cultures sont consommées par les corbeaux lorsqu'elles sont inférieures à une hauteur critique. De plus, la perte d'argent liée à ce problème se situe à environ 162 euros par hectare, et avec près de 120 000 ha de maïs détruits par les oiseaux chaque année, cela revient donc à une perte de 19 440 000 euros pour le maïs uniquement.

Une méthode qui a montré son efficacité est d'utiliser un effaroucheur, un répulsif sonore contre les espèces indésirables d'oiseaux. Il suffit d'activer l'effaroucheur à un intervalle de temps donné (par exemple, les moments de la journée où les oiseaux viennent picorer les champs). Mais cela implique deux problèmes :

- Les corbeaux s'adaptent facilement aux effaroucheurs et savent quand ils se déclenchent dans la journée, ce qui implique d'utiliser les effaroucheurs plus souvent
- Les effaroucheurs produisent des nuisances sonores pour le voisinage

1.2 Objectifs

Afin de pallier à ces problèmes, une solution de détection d'oiseaux sur des images prises par des capteurs positionnés dans des champs a été proposée. Les caméras utilisées sont associées à un Raspberry Pi 3 afin de faire tourner l'algorithme de détection. Si la caméra détecte un oiseau, un système effaroucheur associé à la carte Raspberry Pi 3 se déclenche afin de le faire fuir.

Ainsi, l'objectif de ce projet est de pouvoir mettre en place un outil d'aide à la reconnaissance d'oiseaux sur des images prises par ces capteurs. Ce travail a déjà en parti été réalisé par d'autres stagiaires. Nos objectifs s'inscrivent donc dans la continuité de leurs travaux.

Missions qui nous ont été confiées :

- Prise en main des codes Python déjà réalisés.
- Prise en main de l'outil électronique : connexion au serveur distant et execution des codes.
- Création de nouveaux codes pour obtenir les résultats attendus.
- Améliorer la détection des oiseaux en identifiant les mouvements de tête.
- Changer le traitement de l'image : passer d'un histogramme de couleur RGB au HSV.
- Faire de la prédiction sur les histogrammes de couleur.

Une autre idée a été proposée, celle d'améliorer la détection automatique des oiseaux en changeant la méthode de Machine Learning utilisée. Pour ce faire, nous avons proposé d'effectuer du co-apprentissage plutôt que du clustering classique qui ne semble pas très bien fonctionner (1% de vrai positifs obtenus). De plus, cela permettra d'utiliser des données non annotées pour entrainer et améliorer la détection de corbeaux.

1.3 Présentation des données

Nous disposons dans le cadre de ce projet des images obtenues en sortie du capteur. Les caractéristiques de celles-ci sont les suivantes :

- Les images sont issues de 20 parcelles.
- Nous disposons d'1 million d'images.
- Parmi ces images, 2600 ont été annotées.

— Ces images annotées ont été segmentées en un total de 6000 imagettes. annotées. *(fig 1)*

(a) Chevreuil (b) Corneille (c) Pigeon

FIGURE 1 – Exemple d'imagettes

1.4 Défis :

Plusieurs problèmes ont été repérés sur les différentes images récupérées :
- Le changement de luminosité au niveau des images.
- Modification des sols.
- La profondeur de champs qui est variable.
- Un seuil de détection jusqu'à 200 mètres.
- Certaines images présentent des gouttes/buée qui gènent la prise de vue.
- La plupart des images ne sont pas annotées, ce qui nous fait un faible échantillon d'apprentissage.

2 Outils et Méthodes

2.1 Organisation et répartition du travail

Nous nous sommes répartis en 2 sous-groupes afin de travailler sur un maximum d'objectifs possibles :
- Un sous-groupe, composé de Chaimae, Pauline et Donatien, s'est occupé de la partie "Analyses d'images". Ce sous-groupe a pris en charge l'amélioration du code en implémentant de nouvelles fonctionnalités dans le traitement des images (Obstruction de la ligne de vue, prédiction sur les histogrammes de couleur, détection des "fantômes" des oiseaux sur l'image).

- L'autre sous-groupe, composé d'Anthony et Dalyan, s'est occupé de la partie "Réseau de neurones". Ce sous-groupe s'est concentré sur l'amélioration de la détection d'oiseaux sur des images non annotées par des réseaux de neurones.

2.2 Outils

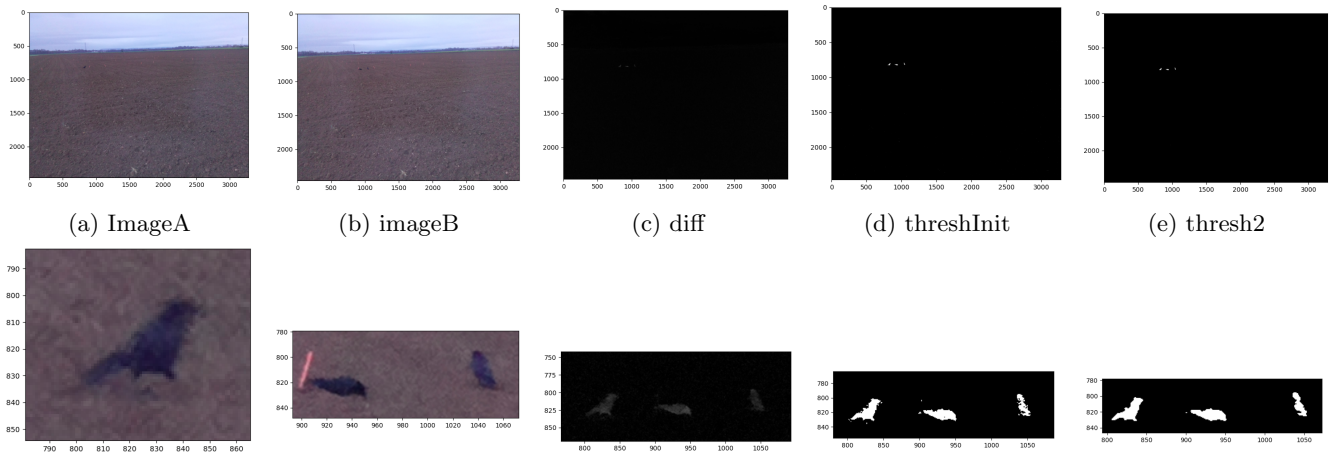
Pour nous permettre d'avoir accès au code ainsi qu'aux différentes images, un accès au serveur ssh distant nous a été donné, ainsi qu'à un repository BitBucket.
En plus de cela, un environnement virtuel a été créé par notre encadrant, avec plusieurs packages python installés dont voici une liste non exhaustive : Scipy, Scikit-image et Scikit-learn, OpenCV, TensorFlow. Pour le co-apprentissage, nous avons choisit d'utiliser non pas TensorFlow mais PyTorch.
Le langage de programmation R est aussi utilisé lors de l'analyse des images dans le prétraitement.

3 Résultats actuels de la détection d'oiseaux

Ce projet de détection des oiseaux ravageurs des cultures agricoles est mené à l'INRAE depuis 3 ans. Notre premier objectif a donc été de prendre en main les codes Python déjà mis en place que nous allons brièvement résumer et exposer dans cette partie et montrer les résultats déjà obtenus.

Actuellement, le sytème mis en place prend en entrée deux images successives qu'il analyse ensuite.
La première étape de l'analyse vise à distinguer les élements caractéristiques comme les oiseaux, chevreuils, ... Elle comporte les étapes suivantes :

- `diff = CumulCanalDiff(imageB,imageA)`
  Fait un cumul des différences sur les 3 canaux de deux images pour éviter les compensations de passage bleu-rouge pas exemple.
- `threshInit = GausThresh(diff,threshMultFull)`
  Extrait la gaussienne ajustée à l'histogramme de la différence entre 2 images afin d'identifier les Pixels dont la valeur est complètement improbable avec un seuil très bas de threshMultFull. Ainsi ces Pixels correspondent très probablement à un objet détecté.
- `thresh2=MajorityOnNeigh(threshInit,5)`
  Seuille sur la moyenne des pixels voisins



(a) ImageA (b) imageB (c) diff (d) threshInit (e) thresh2

(f) zoom ImageA (g) zoom imageB (h) zoom diff (i) zoom threshInit (j) zoom thresh2

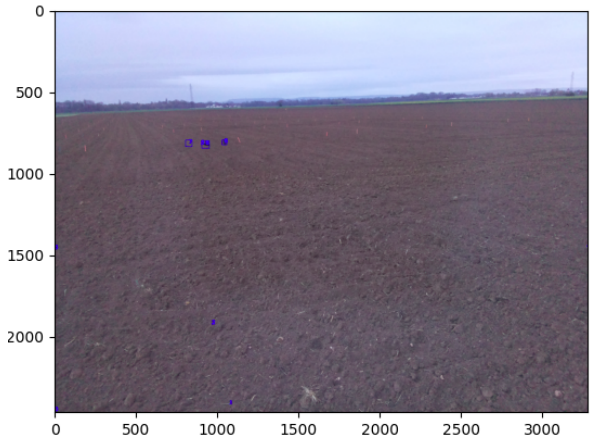
FIGURE 2 – Résultat de la visualisation des étapes d'analyse des images

La suite de l'analyse consiste en la récupération des contours des éléments identifiés dans la variable `cnts` à l'aide de la bibliothèque cv2, puis on trace sur l'image les élements identifiés. :

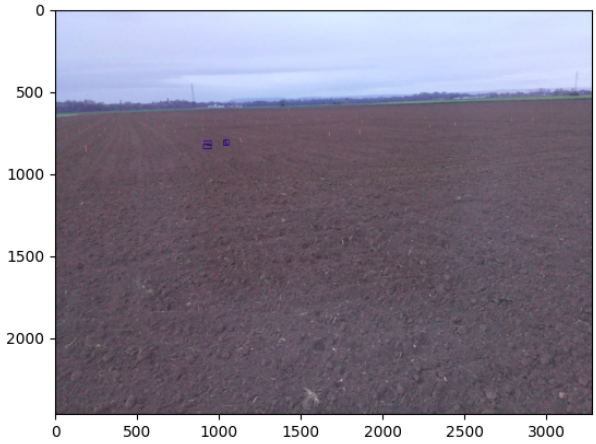
- `initDiff=WriteNumCnts(DrawContextFromCnts(imageB,cnts),cnts)`
  Permet de tracer les contours identifiés sur l'imageB ainsi que leur numéro d'index. Or on remarque que parmi les 9 élements identifiés, il y en a des évidents à enlever. Cette analyse a été effectuée pour à terme ne sélectionner que les 2 imagettes correspondant aux 2 pigeons réellement présents sur l'imageB.
- `finale = DrawContextFromCnts(imageB,cntsNext)`

Enfin, dans une dernière partie, on récupère les données qui nous interessent sur les zones identifiées et on essaie de déterminer leurs classes (oiseau,pigeon, ...) :

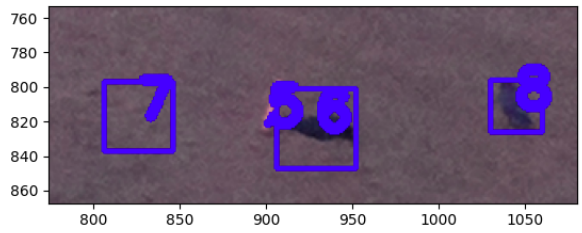
- `XYmmTable = XYMinMax(cnts)`
  Pour récupérer les sommets des carrés des imagettes
- `tableau2 = predict_ResNet(pathImagettes, classes, ctx, transform_test,label_desc, finetune_net, imageB, nameB, cnts)`



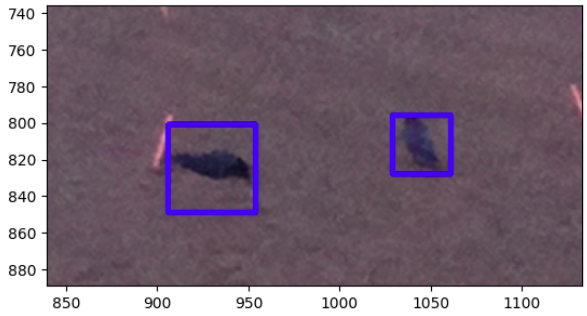
(a) initdiff



(b) finale



(c) zoom initdiff



(d) zoom finale

FIGURE 3 – Résultat de la visualisation des étapes de détection de contours.

Cette fonction permet de prédire la classe des imagettes en associant des probabilités à chacune des classes ('autre', 'cheval', 'chevreuil', 'corneille', 'dirt', 'faisan', 'ground', 'lapin', 'pigeon', 'tracteur', 'voiture').

```
(Pdb) print(XYmmTable)
   xmin  xmax  ymin  ymax
0   906   953   816   834
1  1035  1055   796   827
```

(a) XYmmTable

```
(Pdb) print(tableau2)
      autre    cheval  chevreuil  ...  tracteur   voiture     classe
0  0.284457  0.000358   0.000875  ...  0.001042  0.003848  corneille
1  0.279011  0.000033   0.000253  ...  0.000295  0.001041  corneille
```

(b) tableau2

FIGURE 4 – Visualisation de certaines données associés aux imagettes.

Plusieurs modèles de réseau de neurones ont déjà été entrainé pour la détection sur les images en fonction de divers classes : (chevreuil, corneille, faisan, lapin, pigeon, ...). Le problème actuel avec les modèles utilisés était systématiquement le sur-apprentissage.

4 Avancée du projet

Une fois les analyses précédentes bien maitrisées et les outils bien appréhendés, nous avons pu nous lancer dans les deux grandes thématiques évoquées. D'une part, continuer l'analyse statistique et d'autre part mettre en place des techniques d'entrainement de réseaux de neurones avec des problématiques de Data Augmentation et de Co-apprentissage.

4.1 Gestion des oiseaux qui bougent légèrement ou pas du tout

Suite au premier jeu d'analyse afin de détecter des oiseaux, un constat a été réalisé : une signature de la présence d'oiseaux sur une image était qu'à l'instant d'après il s'était déplacé. Cela a abouti à la détection basée sur la différence de deux images successives. Or un problème survient lorsque l'oiseau bouge peu ou ne bouge pas du tout ce qui arrive sur certaines images. Dans ce cas-ci l'imagette détectée est tronquée ou inexistante. Pour résoudre ce problème nous avons d'abord mis en place une fonction `hasFantome` qui permet de référencer dans chaque image quelle oiseau est considéré comme un fantôme d'un autre oiseau à un instant précédent. Cela signifie que l'oiseau a peu bougé d'une image à l'autre. Ensuite dans un second temps, nous avons voulu régler ce problème d'imagettes tronquées lorsque l'oiseau bouge une ou plusieurs parties de son corps (bec, queue, tête, ...).

4.1.1 Fonction `hasFantome`

`hasFantome` est une fonction qui prend en entrée deux tableaux qui contiennent les informations des imagettes contenant des oiseaux pour des images successives à un instant t et à instant t+1 (aire, perimetre, nom, ...). Son objectif est de pouvoir déterminer si une imagette d'un oiseau à un instant t+1 correspond à une imagette d'un oiseau à un instant t. Pour cela on calcul un taux de recouvrement entre les imagettes et si celui-ci est supérieur à 0% alors cela signifie que l'imagette B est un fantôme de l'imagette A. On peut retrouver sur le github un lien vers cette fonction hasFantome.py.

On obtient en sortie de cette fonction, un tableau qui indique, le chemin et les noms des images considérées à l'instant t et à l'instant t+1, l'index de l'imagette à l'instant t ainsi que l'index de l'imagette à l'instant t+1, la surperficie de recouvrement (en pixel) ainsi que le taux de recouvrement de A et le taux de recouvrement de B.

Sur la Figure 5, on donne un exemple de sortie de cette fonction sur deux images successives présentées en Figure 6 . On peut lire d'après le tableau que l'imagette 1 à l'instant `t+1` est un fantôme de l'imagette 0 à l'instant `t`. En d'autres termes, l'oiseau a légèrement bougé entre l'image à l'instant `t` et à l'instant `t+1` mais pas suffisament pour que l'aire de recouvrement soit nulle. De même les imagettes 0 et 2 à l'instant `t+1` sont des fantômes de l'imagette 1 à l'instant `t`.

Limites, améliorations et continuations

Cette fonction nécessite des ajustements dans le cas où les tableaux qu'elle prend en entrée contiendraient des valeurs manquantes 'NaN'.

	Path	indexA	nameA	indexB	nameB	AreaOverLap	RecouvrementA	RecouvrementB	verdict
0	/work/c3po_all/c3po_interface_chaimae/images_tests/	0	image_2021-04-10_07-04-59.JPG	1	image_2021-04-10_07-05-31.JPG	912	1	0.206101694915254	True
1	/work/c3po_all/c3po_interface_chaimae/images_tests/	1	image_2021-04-10_07-04-59.JPG	0	image_2021-04-10_07-05-31.JPG	504	0.75	0.508064516129032	True
2	/work/c3po_all/c3po_interface_chaimae/images_tests/	1	image_2021-04-10_07-04-59.JPG	2	image_2021-04-10_07-05-31.JPG	380	0.56547619047619	0.451843043995244	True

FIGURE 5 – Tableau de correspondance obtenu en sortie de hasFantome

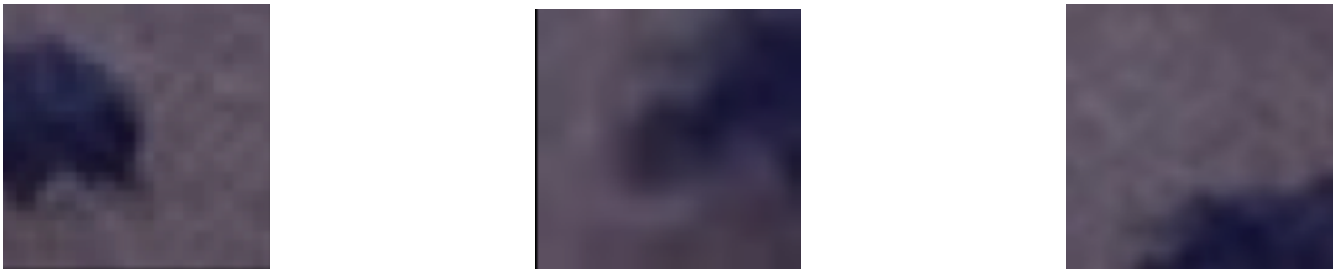
(a) Image instant `t`

(b) Image instant `t+1`

FIGURE 6 – Deux images d'oiseaux successives avec les contours détectés.

4.1.2 Obtention d'imagettes plus complètes

Une des limites majeure des analyses qui avaient été mises en place concerne la détection des imagettes qui sont souvent incomplètes et tronquées. Des exemples sont fournis sur la Figure 7.



(a) Imagette 1

(b) Imagette 2

(c) Imagette 3

FIGURE 7 – Exemple d'imagettes tronquées.

Afin de palier à ce problème il a donc été décidé l'ajout d'une étape à l'analyse. Une fois que la soustraction entre deux images a été réalisée, un oiseau qui bouge que la tête et que la queue mais pas le corps sera identifié avec deux imagettes car le contour se fera sur les parties qui ont bougé. Ainsi pour prendre en compte également le corps de l'oiseau (qui lui ne bouge pas), une approche visant à sommer la soustraction des deux images ainsi que la soustraction finale à l'issu de l'étape précédente est mise en place. Une figure récapitulative est fournie en Figure 8. Ainsi cette technique implémentée a permis d'améliorer la détection de certains oiseaux sur des images comme illustré sur la Figure 9

Limites et continuations

Cette approche s'avère être très utile pour les premières images mais lorsque un élément autre qu'un oiseau est détecté, elle révèle des failles. Des améliorations sont à apporter dans le futur.

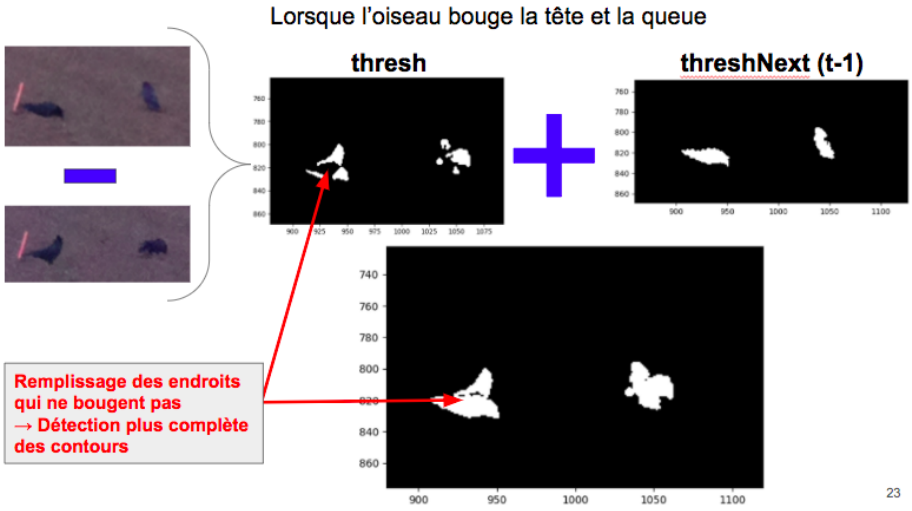


FIGURE 8 – Amélioration de la détection des oiseaux

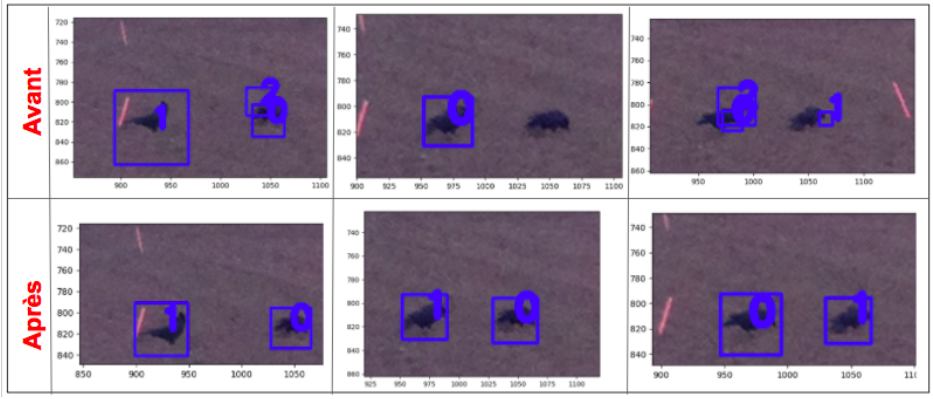


FIGURE 9 – Résultats

4.2 Classification par couleurs

Le réseau de neurones de la fonction `Res_predict` ne présentant pas des performances satisfaisantes quant à la reconnaissance des objets sur les imagettes, notamment du fait des variations de netteté et de taille, l'objectif est de trouver des critères plus simples et plus contrôlés, en particulier statistiques, pour compléter l'analyse. En particulier, on cherche à exploiter les histogrammes de couleur pour la reconnaissance d'oiseaux (un pigeon est plus gris qu'une corneille par exemple). Les essais ont été menés en deux temps : en premier lieu, dans un but exploratoire, seuls les histogrammes d'intensité sont exploités à des fins de classification de type *un contre tous*,sur certaines classes très représentées telles que 'corneilles', 'autre' et 'ground'. Des modèles de PLS-DA et de forêts aléatoires sont testés. Dans un second temps, dans un soucis de complémentarité avec la reconnaissance d'oiseaux par le nouveau réseau de neurones implémenté au cours du projet, seule la détection d'oiseaux et la différenciation de corneilles/pigeons par forêt aléatoire sont recherchées.

4.2.1 Analyse exploratoire

Pour cette première analyse, un jeu de données de 1488 imagettes, dont 432 sont labellisées corneilles. Aucune imagette n'est labellisée pigeon.

PLS-DA *(Partial Least Squares - Discriminant Analysis).*

La régression PLS peut être utilisée à des fins de classification. C'est un outil classique d'analyse spectrale (Ballabio et Consonni, 2013). Pour cela, le vecteur Y des variables quantitatives à expliquer est remplacé par un vecteur de variables indicatrices.On obtient pour chaque individu un résultat de régression compris entre 0 et 1. Arbitrairement, un seuil de 0.5 est appliqué pour l'attribuer, mais celui-ci peut être optimisé en approximant les distribution des classes par des gaussiennes et en appliquant un théorème de Bayes pour calculer une probabilité d'appartenance à posteriori. Dans le cas de plusieurs classes, le vecteur indicateur Y est remplacé par une matrice indicatrice constituée d'une colonne par classe. De la même manière qu'une régression PLS quantitative, les zones du spectre étudié ainsi que le nombre de variables latentes sont des paramètres à optimiser.
Après implémentation, une premier test est effectué sur 20 imagettes d'un même champ, sur lequel des personnes apparaissent ou non. Une projection des nouvelles variables sur les deux premiers axes permet de visualiser une séparation relativement nette à l'oeil nu (fig (3)). Deux prétraitements sont appliqués : une normalisation par la somme des intensités pour prendre en compte les différences de luminosité au cours de la journée, ainsi qu'une transformée logarithme afin de faire ressortir l'information contenue dans les composantes de faible intensité.
La régression PLS-DA sur un jeu complet d'imagettes annotées ne se révèle par la suite d'aucune utilité : les projections des individus des deux classes sont complètement superposées, et les estimations de classe par la PLS sont donc concentrées sur des valeurs proches de 0.5, rendant délicat l'utilisation d'un seuil. De manière générale, cette méthode semble trop complexe par rapport à l'interprétabilité recherchée.

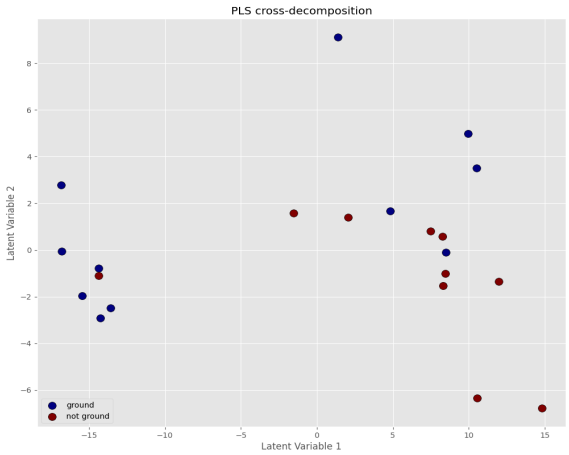


FIGURE 10 – projection préalable de 20 individus sur le deux premières variables latentes - Classe "ground" contre le reste

Essais de forêts aléatoires. Les forêts aléatoires consistent à classifier les échantillons à l'aide d'un vote majoritaire de différents arbres simples de décision, construits sur un nombre limités de variables et à partir d'échantillons tirés aléatoirement, de manière à limiter leur dépendance mutuelle. Cette technique présente l'avantage d'utiliser des critères de décision simple, tout en étant plus robuste au surapprentissage que la PLS-DA. Dans le cas présent, le système d'apprentissage est constitué de 100 arbres, et les performances de classification *un contre tous* sont établies suite à une validation croisée à 50 plis, avec k= 30% des échantillons utilisés en apprentissage. Les données, comme précédemment, ont subi deux prétraitements : une normalisation par la somme des intensités et une transformée logarithmique. Pour les classes *ground, autre+humain et corneille*, suffisamment représentées dans le jeu de données (au moins 100 parmi les 1488 individus), on obtient les précisions moyennes suivantes :

classe	ground	autre+humain	corneille
précision	0.50	0.78	0.67

Les simples histogrammes d'intensité ne permettent donc pas d'identifier de façon fiable les formes présentes sur les imagettes. Cependant, les forêts aléatoires proposant des performances meilleures que le hasard, ces essais prouvent qu'il y a effectivement de l'information à exploiter dans les histogrammes de couleur.

4.2.2 Détection et reconnaissance d'oiseaux

Pour cette section, 827 images supplémentaire de pigeons sont ajoutées au jeu de données. Les histogrammes d'intensité s'étant révélés inefficaces pour des fins de classification, les images sont converties au format HSV afin d'obtenir des histogrammes de teinte sur les 360° du cercle chromatique encodés dans le canal H. De plus, une détection de contours est effectuée avec un filtre d'OTSU. Ces deux traitements permettent de limiter à la fois les biais liés à la couleur du sol et à l'exposition. Il est cette fois décidé de ne classifier les images qu'au moyen de forêts aléatoires, cette méthode ayant donné de meilleurs résultats au cours de l'analyse d'exploratoire. Il s'agit maintenant d'optimiser trois paramètres : la résolution des histogrammes, le nombre de classifieurs et la section du cercle chromatique exploitée.

Obtention des histogrammes de teinte L'image convertie au format hsv, un filtre d'Otsu est appliqué sur le canal H de manière à ne considérer que les pixels présents à l'intérieur des formes se détachant de l'arrière-plan, comme dans l'exemple figure 11. Un histogramme du canal H est ensuite réalisé. Celui-ci contient les quantités de

(a) photo originale

(b) filtre d'Otsu

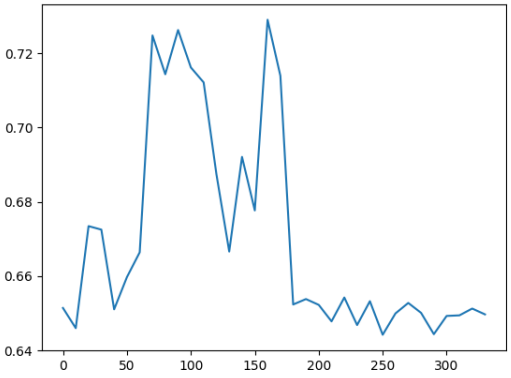
FIGURE 11 – Application du filtre de contour sur une photo de corneille. Seuls les pixels blancs sont considérés par la suite.

pixels dont la teinte h correspond à chacun des 360° du cercle chromatique. Comme pour l'analyse exploratoire, une normalisation par la somme du nombre de pixel est effectuée afin de compenser les différences possibles de taille entre les imagettes.

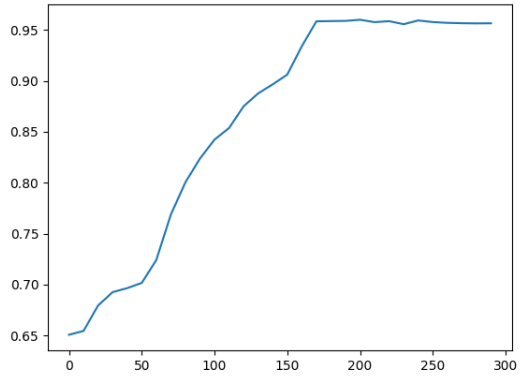
Optimisation des paramètres de la forêt aléatoire. Le principal intérêt de l'analyse d'histogrammes est de pouvoir classifier les différentes imagettes avec un coût calculatoire faible, en limitant le surapprentissage. Il est donc pertinent de limiter à la fois l'information en entrée du système, ainsi que ses dimensions. Pour faciliter l'implémentation, les mêmes paramètres du modèle seront appliqués pour la détection et la reconnaissance. Les résultats présentés par la suite sont ceux ayant permis la prise de décision, et peuvent donc porter donc sur l'étude de l'une ou l'autre des deux tâches.
Une première stratégie consiste à réduire la résolution des histogrammes en remplaçant chaque groupe de 2 ou 3 bins par leur moyenne respective. Le tableau 4.2.2 présente la précision moyenne des classifieurs (corneille ou pigeon)/(autre) suite à une validation croisée à 10 plis, associée l'écart type correspondant :

résolution (bins)	120	180	360
précision	74.2 ± 0.1	74.5 ± 0.1	75.2 ± 0.1

La perte de précision étant jugée trop importante par rapport à la faible compression proposée, il est plutôt proposé de ne sélectionner qu'une certaine fraction du cercle chromatique de résolution maximale. Pour estimer l'utilité de l'information discriminante portée par chaque section, la précision d'un classifieur corneille/pigeon est évaluée à l'aide d'une validation croisée à 10 plis effectuée pour chaque segment de 10° (figure 12(a)). La majorité de l'information utile étant regroupée sur la première moitié du cercle, le même protocole est appliqué pour une forêt aléatoire recevant un segment [0,n] du cercle, pour n allant de 10 à 300° (figure 12(b)).



(a) Précision moyenne du classifieur pigeon/corneille par segment de 10°



(b) Précision moyenne du classifieur pigeon/corneille par taille du segment (°)

FIGURE 12 – Sélection de la zone d'information utile du cercle chromatique

La précision du classifieur pigeon/corneille se stabilisant à partir de 180°, il est décidé de ne considérer que la section [0,200°] par la suite. La réduction de dimensionnalité est modérée, mais ne réduit pas la précision des forêts aléatoires et contribue à conserver des modèles les plus parcimonieux possible.
Enfin, il convient de choisir un nombre minimal de classifieurs au sein des modèles. Pour ce faire, la précision moyenne d'une classification (pigeon ou corneille)/autre est estimée sur une validation croisée à 10 plis, avec en entrée la section [0,200°] du cercle chromatique (figure 13).

Le compromis entre la complexité des calculs et la précision de la classification optimale est jugé optimal pour 40 classifieurs. Les modèles sont finalement implémentés avec 40 classifieurs, et reçoivent le segment [0, 200°] du cercle chromatique. Les précisions finales des modèles retenues sont les suivantes :

tâche	Détection d'oiseaux	Identification corneille/pigeon
précision	75.27%	95.27 %

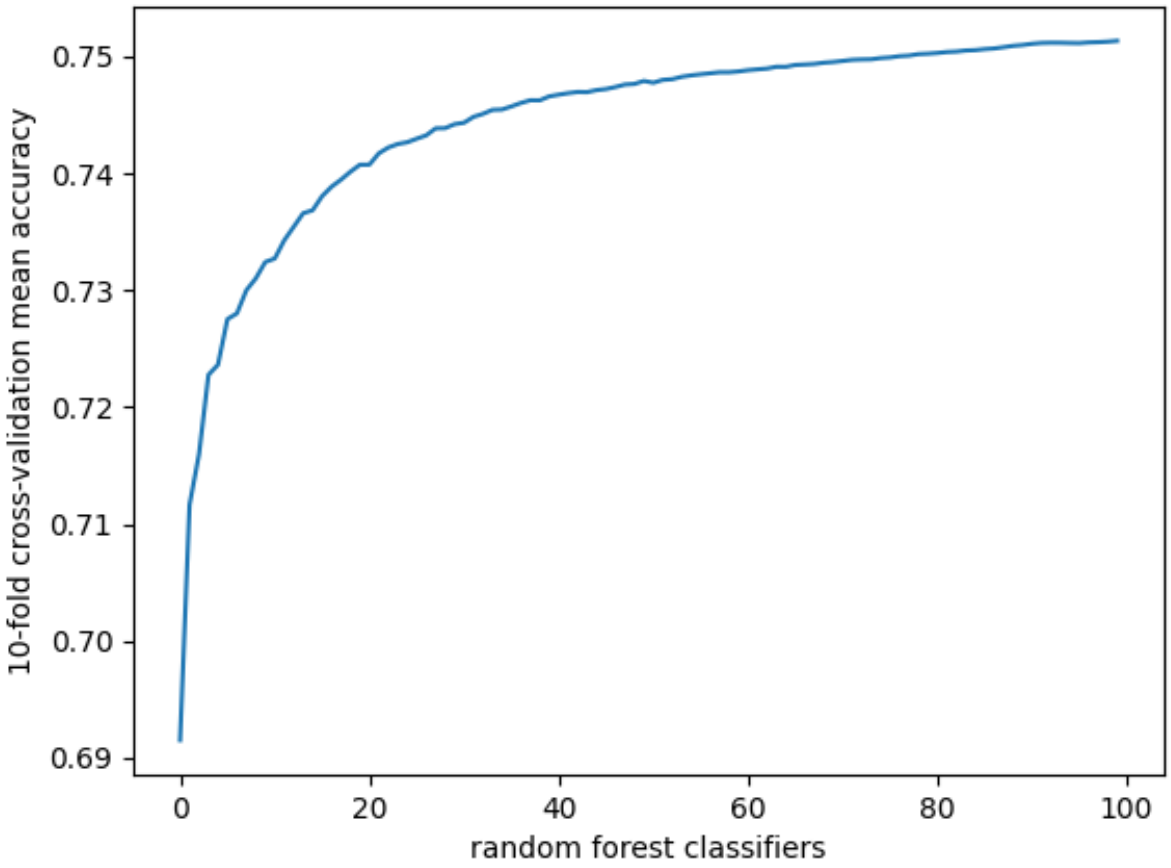


FIGURE 13 – Précision moyenne en fonction du nombre de classifieurs

Du fait de sa simplicité et de sa bonne performances sur l'identification d'oiseaux, la classification par histogramme de couleur agit en complément du réseau de neurones implémenté au cours du projet. L'utilisation des histogrammes de couleurs ne semble cependant pas satisfaisante pour des fins de détection, d'autant plus qu'une comparaison avec un classifieur plus complexe sur les mêmes données prétraitées (Boosting implémenté avec l'algorithme AdaBoost sur 100 arbres de décision de profondeur 3 et un taux d'apprentissage de 0.5) donne lieu à une précision comparable (75,2%). Cela laisse supposer que toute l'information utile à la classification est captée par des forêts aléatoires relativement simples. Si l'identification présente *a contrario* des résultats impressionnants, il est à noter qu'un biais d'environnement est envisageable, dans la mesure où toutes les images du jeu d'entraînement ont été acquises sur deux jours et dans le même champ.

4.3 Problème des gouttes

Problème et objectif. Des gouttes (pluie, buée, rosée) peuvent se former sur la lentille des caméras ce qui peut altérer la définition des contrastes d'une image et potentiellement la détection d'oiseaux. L'objectif est alors de pouvoir identifier ces images.

4.3.1 Solution exploratoire : Variance de Laplacien sur image entière

Intuitivement, on peut s'attendre à ce que des gouttes qui se forment sur l'objectif de la caméra induisent une baisse de netteté des constrasts des objets détectés.
Cela peut être quantifié par l'opérateur convolutif dit du `Laplacien`[1] disponible dans la bibliothèque Python OpenCV.

Test préliminaire et résultats. Le Laplacien a été appliqué aux images prises le 01/05/2021 avec le kernel par défaut en 3x3. Ce test préliminaire à mis en évidence les points suivants :

- Le calcul de Laplacien sur une image entière ce fait en temps raisonnable, de l'ordre de 0.3s/image.
- Le calcul de variance semble plus explicatif que celui de la moyenne.
- La variance de Laplacien sur image entière est principalement influencée par la luminosité. L'apparition de gouttes, d'animaux ou d'objets a un impact relativement minime sur cette variance.

Conclusion et Redéfinition de l'objectif. Ainsi pour la suite, il a été décidé de ne travailler qu'avec la variance de Laplacien et de restreindre ce calcul à des zones bien définies beaucoup plus petites que l'image entière.

4.3.2 Solution proposée : Variance de Laplacien sur couronne de cnts

Une méthode plus fine pour quantifier cette variation de Laplacien entre deux images successvies a été mise en place functions_analysis_add_pauline.py, BoucleResNet_Pauline_cnts.py. Elle se base sur la restriction des zones d'intéret aux couronnes des cnts. `ThreshNext` donnée par le système expert déjà mis en place est une image binarisée contenant les cnts identifiés comme pertients. Les couronnes ont été obtenues en prenant la différences de l'image de `ThreshNext` dilatée n fois par l'image de `ThreshNext` érodée n fois *(fig 14)*.

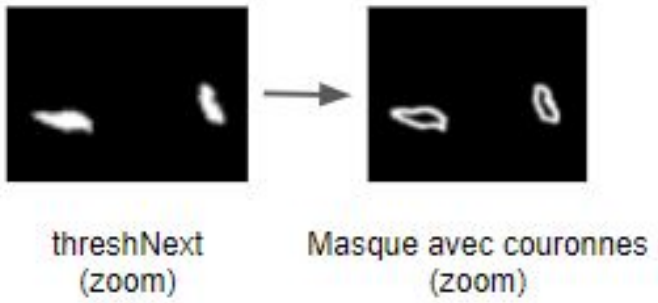


FIGURE 14 – Obtention du masque contenant les couronnes des cnts par soustraction de threshNext dilatée 2 fois avec threshNext érodée 2 fois

L'intéret de travailler sur des couronnes de cnts et non sur les cnts directement permet de pallier des cas où l'intérieur d'un oiseau est plus homogène que le sol sur lequel il se pose, impliquant que l'arrivée d'un oiseau diminue le laplacien au lieu de l'augmenter comme attendu.

La méthode proposée comprend les étapes suivantes *(fig 15)* :

- **Etape 1** : Calculer le Laplacien de chaque image successives A et B
- **Etape 2** : A partir de `ThreshNext`, calculer le masque contenant la couronne de chaque cnts
- **Etape 3** : A partir de ce masque, obtenir pour chaque cnts un sous-masque de la meme taille ne contenant que la couronne du cnts associé. Puis appliquer ce sous-masque à chaque Laplacien des images A et B. Enfin calculer pour chacune des images obtenues la variance sur l'ensemble des pixels dont la valeur est non nulle i.e ceux appartenant à la couronne.

1. https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html

— **Etape 4** : Pour chaque cnts, calculer le rapport de la variance de Laplacien de B sur celle de A. On s'attend à ce que ce rapport soit inférieur à 1 dans le cas d'objets qui sortent de la couronnes ou de formation de gouttes et une valeur supérieure à 1, on s'attend au contraire, à l'apparition d'objets dans la couronne.

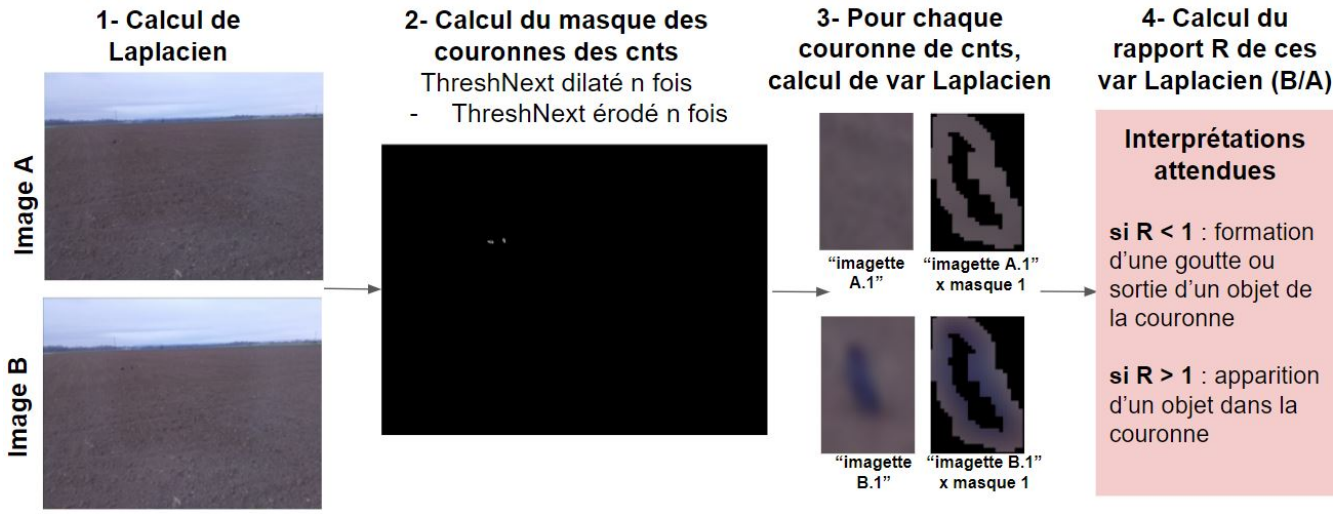


FIGURE 15 – Etapes de la méthode de calcul de variance de Laplacien dans la couronne de cnts sur deux images successives A et B (le 1 à l'étape 3 est relatif au cnts 1)

Résultats et discussion. Afin d'évaluer la méthode, elle a été appliquée à plusieurs couples d'images successives. La figure 16 résume les principaux résultats observés :

— **Cas 1** : Le R-Laplacien est supérieur à 1. On s'attend à ce qu'un objet soit arrivé dans la couronne au passage de l'image A à B.
Remarque : il ne s'agit pas tout le temps d'oiseaux, des cas de voitures refletant la lumière du soleil on pu etre observés avec un R-Laplacien de 10. Mais dans cette situation, s'il s'agit d'oiseaux qui apparaissent, la couronne associée dessine bien les contours de l'animal.

— **Cas 2** : Le R-Laplacien est inférieur à 1. On s'attend à ce qu'un objet sorte de la couronne au passage de l'image A à B ou que cette baisse de variance de Laplacien soit liée à la formation d'une goutte. Toutefois, les R-laplacien ne permettent pas de distinguer ses deux cas. Notons tout de même que suite à la formation d'une goutte, beaucoup de cnts sont sous le critère de taille minimale imposée par le système expert et donc ne sont pas retenus pour la suite de l'analyse. Cette information pourrait aussi etre exploitée conjointement à ce R-Laplacien inférieur à 1 pour caractériser assez probablement la formation d'une goutte.

— **Cas 3** : Le R-Laplacien est proche de 1 (seuil à définir). Ce cas n'avait pas été anticipé lors de l'élaboration de la méthode décrite dans cette section. Mais il est très intéressant. Ce cas met en évidence un objet qui a peu bougé entre A et B. On voit d'ailleurs que la couronne obtenue semble composite, témoignant de mouvements résiduels.

Conclusion. Ainsi le travail effectué sur le problème de la goutte a dérivé après discussion et reflexion vers une problématique plus large. Le travail exploratoire sur les couronnes des cnts a permis de mettre en évidence la possibilité de gain d'informations quand au mouvement d'objets. Cependant, les observations faites doivent etre prises avec précaution et ne doivent pas être généralisées à ce stade. Il conviendrait notamment de tester différentes combinaisons des paramètres comme le kernel utilisé pour le Laplacien ou encore le nombre de dilations/érosions choisi. Aussi un travail de recherche de seuil devrait être réalisé. Une dernière remarque repose sur la grande dépendance de ce travail à la qualité des cnts générés par le système expert.

Cas	Image A	Image B	R-Laplacien (B/A)	Interprétation
1	1.92 	2.79 	1.45	R > 1 → **Arrivée d'un objet**
2	3.02 	1.58 	0.52	R < 1 → **formation d'une goutte ou sortie d'un objet**
3	3.06 	2.77 	0.90	R~1 → **L'objet a fait du sur place**

FIGURE 16 – Cas illustrant la majorités des résultats observés par application de la méthode (les valeurs indiquées en haut à gauche de chaque photo est la valeur de la variance de Laplacien dans le couronne. R-Laplacien(B/A) est le rapport de la variance de Laplacien de B sur A. Les termes "arrivée" et "sortie" dans la dernière colonne sont relatif à la couronne).

4.4 Amélioration de la classification par Deep learning déjà existante

4.4.1 Premier entrainement d'un réseau de neurone sur les données existantes

Pour ce première entrainement nous avons décidé d'éviter d'être trop ambitieux. Dans l'idéal, le projet C3P0 voudrait créer à terme un système de détection et de classification des oiseaux. Le système devrait donc être en mesure non seulement de dire s'il y a un oiseau ou pas sur une prise de vue mais également de classifier l'oiseau dans les différentes espèces classiques qu'on retrouve dans les champs (pigeon, corneille, faisan...). Cependant au vu du relatif faible nombre de données annotées il a été décidé de commencer par entrainer un système capable simplement de dire si une image contient un oiseau ou pas. De plus l'architecture choisie est une architecture ResNet18, la plus petite des architectures ResNet. Elle a été sélectionné car plus l'architecture est grande et plus il faut de données pour l'entrainer. De plus contrairement à une base de données comme ImageNet qui compte 1000 classes à distinguer, le problème ici est une classification binaire donc on peut supposer qu'il est plus simple.

De la data augmentation a également été appliquée aux données. Cette pratique qui à la base permettait un meilleur apprentissage sur des petits jeux de données est désormais utlisée quelle que soit la taille du jeu de données. En effet, en forçant le réseau de neurone à être invariant à un certain nombre de petits changements, on le force à mieux isoler le concept cible ce qui permet une meilleure généralisation. De plus au vue des capacité d'adaptation énorme de ces systèmes, les nouvelles images générées agissent comme de nouveaux exemples. Pour ce première entrainement, des variation aléatoires d'angle, de translation et de taille ont été appliquées aux exemples.

Comme l'objectif était de faire une classification binaire entre les images d'oiseaux et les images contenant autre choses que des oiseaux, cela implique de ranger les différentes annotations sur les images dans ces deux

catégories. Pendant cette phase il s'est avéré qu'une des catégorie nommée incertain contenait des images contenant probablement des oiseaux et d'autre contenant certainement des objets qui n'étaient pas des oiseaux. Comme il était impossible de ranger cette catégorie selon si elle contenant des oiseaux ou pas, il a été décidé d'exclure cette catégorie des données d'entrainement.

Une fois cette travaille de réflexion en amont achevé, l'entrainement a été implémenté en python en utilisant le bibliothèque PyTorch et l'optimiseur Adam pour la descente de gradient. 90% des données ont été utilisées pour l'entrainement et 10% pour tester les performances du réseau de neurones. A l'issue de l'entrainement, le réseau de neurone avait un taux d'erreur de l'ordre de 4 à 5% en considérant que le jeu de donné global est composé à 37% d'images d'oiseaux et à 63% d'images n'étant pas des oiseaux. Ce premier résultat sans optimisation particulière de l'optimiseur ou de l'architecture est donc encourageant et laisse penser que la reconnaissance des oiseaux par un réseau de neurone peut fonctionner.

Dans un second temps, du transfert learning a été effectué en utilisant les premières couches d'un ResNet18 pré-entrainé sur la base de données ImageNet. L'apprentissage par transfert est populaire car il permet généralement d'obtenir de meilleur performances. Dans le cas actuel, cette méthode permet d'améliorer les performance en ayant un taux d'erreur de l'ordre de 3 à 4% et augmente la vitesse de convergence car il faut moins d'epoch pour que le modèle atteigne sont taux d'erreur maximal.

4.4.2 Entrainement sur des données non pré-traité

Les données existantes n'ont pas été directement découpées à partir des images sources, un prétraitement a été réalisé. Pour extraire de la photo la zone annotée, son centre et sa longueur la plus grande sont déterminée puis un carré de côté 1.2 fois la largeur centrée sur le centre de la zone est découpé. Cette zone découpé est ensuite redimensionnée en 96 par 96 pour fabriquer les images d'entrainement.

Cette procédure est cohérente avec le but de l'entrainement du réseau de neurone : prendre en entrée les zones contenant potentiellement un oiseau déterminées par la phase de prétraitement qui auront été découpées de l'image prise par la caméra selon une logique similaire. Cependant, il serait intéressant de mesurer les performances d'un réseau de neurones entrainé sur des images en taille réel pour voir à quel point le réseau de neurone arrive à reconnaitre les oiseaux sans prétraitement. Si le réseau de neurone fonctionne bien il serait peut être possible d'améliorer le réseau de neurone pour qu'il soit capable d'effectuer la reconnaissance sans prétraitement.

Pour ce faire, les zones annotées dans les images sources ont été redécoupées pour extraire des carrés de 96 par 96 sans redimensionnement. Cependant, les échantillons annotés comme sol ont été découpés trop près des oiseaux, des oiseaux se retrouvaient donc inclus dans les images annotées comme ne contenant pas d'oiseaux. La solution à ce problème a consisté à découpé de manière aléatoire des images ne contenant pas d'oiseaux dans l'image en considérant que si on découpe une zone de l'image qui n'a pas de recoupement avec les zones annotées alors on obtient une image ne contenant pas d'oiseau.

Pour ce faire, une classe python a été implémentée qui permet sur la base d'une image de tirer des zones aléatoires ou spécifique de l'image de manière à ce que tous les tirages aléatoires suivant ne contiennent pas de zone de l'image déjà tiré. L'implémentation est loin d'être évidente car on veut une structure de donnée qui permette à la fois de tirer rapidement et aléatoirement une partie de l'image et de supprimer facilement les zones qui ne peuvent plus être tirées. Une liste permet de facilement tirer aléatoirement un élément mais pas de tester l'existence et de supprimer un élément dont on ne connait pas forcément la position dans la liste. Un dictionnaire permet de facilement tester l'existence d'un élément et de le supprimer mais ne permet pas de tirage aléatoire rapide. Au final, l'optimisation ne posant pas trop de problème, l'implémentation a été faite en utilisant un dictionnaire qui stocke les coordonnées de tous les centres d'images valides dans les clés. Lorsqu'une image est tirée, tous les centres trop proches sont supprimés du dictionnaire. Pour le tirage aléatoire cela implique de créer une liste des clés ce qui n'est pas une solution très efficace d'un point de vue algorithmique. Pour avoir une solution efficace, il faudrait créer une structure de donnée qui combine les deux propriétés.

Une fois les nouvelles images générées un ResNet18 a été entrainé sur les nouvelles données, Le model atteint 5% de taux d'erreur ce qui est moins bon que dans le cas précédent mais reste quand même largement au dessus du hasard.

4.4.3 Intégration des modèles au code existant

Le but original de l'entrainement des réseaux de neurones était de pouvoir les faire tourner dans le script principale, boucle_resnet.py. Cependant, il n'y avait rien de spécifique prévu pour l'intégration de nouveaux modèles, les deux modèles précédents ayant été directement insérés dans le code principal. Pour pouvoir insérer nos modèles plus facilement et pour intégrer plus facilement de futurs modèles, il a été décidé de créer une bibliothèque python pour contenir les modèles. La bibliothèque contient chaque modèle dans un fichier python séparés. Le fichier __init__.py centralise tous les modèles implémentés et propose une fonction avec une syntaxe uniforme pour les modèles. Cela permet d'éviter au maximum de modifier le code principal lors de l'intégration de nouveau modèles et de bien séparer les codes nécessaires au chargement des différents modèles pour plus de lisibilités du code principal ainsi que du code des modèles.

4.4.4 Interprétabilité du modèle établi

Mais même avec un modèle performant, on en vient toujours à se demander si celui-ci est interprétable. On sait que les modèles d'IA, plus précisément les modèles de Deep Learning comme ResNet18, sont souvent assimilés à ce qu'on appelle des "boites noires" dû à leur fonctionnement opaque, cachés parmi les différentes couches de neurones du modèle, et qui n'est pas forcément compréhensible par les scientifiques. La complexité des modèles est une force, mais aussi une faiblesse puisque cela les rend peu interprétables.

Pour palier à ça, différentes méthodes permettent de déterminer l'interprétabilité d'une IA, pour rendre le fonctionnement du modèle et ses résultats davantage transparent et compréhensible pour les différents acteurs. Celle que nous avons utilisé est la méthode Grad-CAM.

Comme le modèle utilisé est un modèle convolutif, le Grad-CAM est une des méthode spécifiques à ce type de modèle pour déterminer l'interprétabilité de notre modèle. Cela consiste à retrouver quelles parties de l'image ont conduit le réseau à sa décision finale. On reproduit donc les cartes thermiques, représentant les classes d'activations sur les images reçues en input. Ici, les deux classes d'activation sont "Oiseau" et "Non Oiseau". Les pixels vont être colorés en rouge ou bleu en fonction de l'importance de chaque pixel par rapport à la classe concernée.
Prenons les six images qui nous ont servi à tester notre modèle et appliquons la méthode Grad-CAM. (*Fig. 17*)

Premièrement on voit que pour la classe 1 : "Non oiseau", toute l'image est coloré en bleu. Cela veut dire que toute l'image n'est pas importante pour le modèle, donc aucun des pixels n'a été gardé.
Par contre, pour les images de la classe 0 : "Oiseau", les pixels colorés sont ceux sur lequel on peut observer un oiseau. Ainsi, les pixels importants pour le modèle sont les pixels sur lesquels se situent un oiseau. On peut donc en conclure que le modèle détecte la présence d'un oiseau sur les images.

Le problème est que certains des pixels importants se situe en dehors des pixels des oiseaux, tandis que d'autres ne sont pas considérés comme important : cela montre la limite du modèle à détecter correctement les oiseaux.
De plus, il s'agit d'un modèle où la descente de gradient n'est pas effectuée : on récupère uniquement la classe qui est prédite avec la plus grande probabilité. Or, la méthode Grad-CAM nécessite une descente de gradient pour pondérer les pixels, donc nous forçons le modèle à faire une descente de gradient, ce qui n'est pas recommandé. Une meilleure façon de procéder aurait été de calculer la fonction de perte par entropie croisée et de faire la descente de gradient à partir de là. L'autre inconvénient majeur de GradCAM était le suréchantillonnage à des résultats de carte thermique grossiers en artefacts et en perte de signal. Une autre méthode pourrait être envisagée en remplacement de celle-ci.

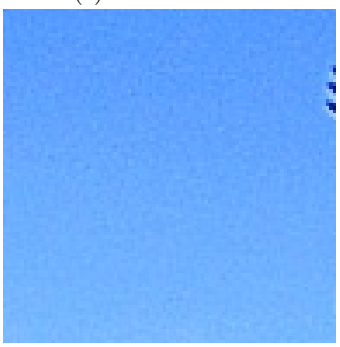
(a) 0 : Oiseau

(b) 0 : Oiseau

(c) 1 : Non oiseau

(d) 0 : Oiseau

(e) 0 : Oiseau



(f) 1 : Non oiseau

FIGURE 17 – Résultat de la méthode Grad-CAM sur les images de test

5 Bibliographie

5.1 Articles

— Classification tools in chemistry. Part 1 : linear models, PLS-DA, Davide Ballabio and Viviana Consonni, Anal. Methods, 2013, 5, 3790
— A System Using Artificial Intelligence to Detect and Scare Bird Flocks in the Protection of Ripening Fruit, Sensors 2021, 21, 4244. https ://doi.org/10.3390/s21124244
— DeepBeesAlert : vers un système de gestion et de protection durable des ressources, Alexis Vergne, CIRAD, IODAA 2021.
— Deep Co-Training for Semi-Supervised Image Recognition, Qiao S., Shen W., Zhang Z., Wang B., Yuille A., ECCV 2018.
— Attentive Generative Adversarial Network for Raindrop Removal from A Single Image, Rui Qian1, Robby T. Tan
— Ulyanin, S. “Implementing Grad-CAM in Pytorch”. Medium, Toward Data Science. Fevrier 2019.

5.2 GitHub

— https ://github.com/soja-soja/AIBirdWatching

5.3 Start up

— https ://www.avix.com/bird-deterrent-service-platform/