



RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE
RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88

<http://www.ifri-uac.net> Courriel : contact@ifri-uac.net



Mémoire pour l'obtention du Master en Informatique

Estimation des rendements agricoles avec le Machine Learning : *cas du Sorgho*

M. Judiacel G. N. ZANNOU
madrojudi.gz@gmail.com

Sous la supervision de :
Dr Ing. Ratheil V. HOUNDJI

Année Académique : 2017-2018

Dédicace

À

- ma mère Rose HOUEMAGNON épouse ZANNOU;
- mon père Francois de Borgia ZANNOU;
- toute la famille ZANNOU.

Remerciements

Je tiens à remercier ici :

- l'Eternel Dieu, créateur du Ciel et de la Terre ;
- le Professeur EZIN Eugène, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- le Professeur EDAH Gaston, Directeur Adjoint de l'IFRI ;
- tous les enseignants de l'IFRI en général et mon directeur de mémoire Dr Ing. HOUNDJI V. Ratheil en particulier pour ses orientations ;
- M. Mouhamed BASSIROU pour son apport dans la réalisation du projet ;
- tout le personnel de l'administration de l'IFRI ;
- tous les étudiants de ma promotion ;
- tous ceux qui d'une manière ou d'une autre m'ont aidé tout au long de ma formation.

Table des figures

1.1 Carré de rendement [1]	8
1.2 Principales étapes pour pratiquer l'agriculture de précision [2]	9
1.3 Interface utilisateur dans la cabine [5]	10
1.4 Schéma illustrant un capteur de rendement et un capteur d'humidité [2]	11
1.5 Schéma d'utilisation du Machine Learning [13]	12
1.6 Apprentissage supervisé et l'apprentissage non supervisé	13
1.7 Exemple d'histogramme	16
1.8 Exemple de métadonnées sur un fichier image	19
1.9 Réseau de neurone du Deep Learning	20
1.10 Classification : Entropie croisée [8]	21
1.11 Architecture d'un réseau de neurones convolutifs	22
1.12 Architecture d'un réseau de neurones convolutifs	22
1.13 Allure de la fonction ReLU [9]	24
1.14 Schéma montrant tous les facteurs pertinents pour le calcul de la GSD [17]	26
2.1 DJI Phantom 4 Pro utilisé pour les prises de vues	29
2.2 Principe de fonctionnement de TensorFlow	30
2.3 Architecture logique de la solution	31
2.4 Découpage des images avec GIMP2	33
2.5 Etiquetage du Sorgho avec LabelImg	34
2.6 Forme d'épi de sorgho	38
3.1 Diagramme de cas d'utilisation	41
3.2 Diagramme des classes	42
3.3 Diagramme d'activités	43
4.1 Graphe des pertes lors de l'apprentissage	45
4.2 Test de détection de Sorgho non mature	45
4.3 Test de détection de Sorgho mature	46
4.4 Interface d'accueil d'AgriDroneFlow	48
4.5 Détection du sorgho non mure	49
4.6 Détection du sorgho mature et estimation du rendement agricole	49

4.7	Liste des épis détectées et leurs poids estimés	50
A.1	Détection du sorgho non mature sur une image originale du drone.	55
A.2	Détection du sorgho mure sur une image originale du drone.	56

Liste des tableaux

2.1	Quelques caractéristiques collectées	39
4.1	Evaluation de l'algorithme de détection des épis de sorgho	47
4.2	Evaluation de l'algorithme d'estimation du poids (gramme) des épis de sorgho .	47

Liste d'algorithmes

2.1	Conversion XML en CSV	33
2.2	Génération des TFRecords	35
A.1	Méthode de détection des cultures (Code Java)	56
A.2	Méthode d'estimation des poids des épis (Code Java)	57
A.3	Méthode de filtrage des doublon (Code Java)	59

Résumé. L'estimation de la production agricole future est un défi important pour les agriculteurs. Nous proposons un système basé sur des algorithmes d'apprentissage automatique pour estimer les rendements des champs à la ferme. Les expériences ont été menées sur un champ de sorgho. Nous utilisons TensorFlow avec les réseaux de neurones convolutionnels et la régression linéaire. Ces algorithmes nous permettent 1) de détecter les différentes épis de sorgho sur une image et 2) d'estimer leur poids. Sur notre ensemble de données, nous obtenons une justesse moyenne de 74,5% pour la détection du sorgho et une précision moyenne de 99% pour l'estimation du poids. Nous avons également développé une application desktop qui utilise ses algorithmes pour une utilisation pratique. Les tests montrent que le système est opérationnel.

Mots clés : *Estimation des rendements agricoles, TensorFlow, apprentissage profond, Sorgho, Afrique, Agriculture.*

Abstract. Estimation of a future agricultural production is an important challenge for farmers. We propose a system based on machine learning algorithms to estimate farm yields of fields. The experiments were conducted on a Sorghum field. We use TensorFlow with Convolutional Neural Networks and Linear Regression. These algorithms allow us 1) to detect the different ears of Sorghum on an image and 2) to estimate their weight. On our dataset, we obtain an average accuracy of 74,5% for the detection of sorghum and an average precision of 99% for the estimation of the weight. We have also developed a desktop application that uses its algorithms for practical use. The tests show that the system is operational.

Key-words: *Estimate farm yields, Machine Learning, Africa, Agriculture, TensorFlow, Sorghum.*

Table des matières

Dédicace	i
Remerciements	ii
liste des figures	iv
liste des tableaux	v
Liste d'algorithmes	vi
Liste des sigles et abréviations	vi
Résumé	1
Abstract	2
Introduction	5
1 État de l'art	7
Introduction	7
1.1 Technique d'estimation des rendements Agricoles	7
1.1.1 Méthode traditionnelle	7
1.1.2 Agriculture de précision	9
1.2 Détection d'objet	11
1.2.1 Machine Learning (ML)	11
1.2.2 Deep Learning	14
1.2.3 Généralité sur les images	14
1.2.4 Classification d'image par réseaux neurones convolutifs	19
1.3 Détermination de la taille réelle des objets au sol sur une image aérienne	25
Conclusion	27
2 Matériel et méthodes	28
Introduction	28
2.1 Matériel	28

2.1.1	DJI Phantom 4 Pro	29
2.1.2	Langage de programmation Python	29
2.1.3	OpenCV	30
2.1.4	TensorFlow	30
2.1.5	LabelImg	30
2.1.6	Java	31
2.1.7	Angular	31
2.2	Méthodes	31
2.2.1	Architecture logique de la solution	31
2.2.2	Mise en place du modèle de détection	32
2.2.3	Détermination de la superficie du champ agricole capturée sur une image	37
2.2.4	Détermination de la valeur estimée	38
2.2.5	Estimation des rendements agricoles	39
	Conclusion	39
3	Solution proposée	40
	Introduction	40
3.1	Présentation du système	40
3.2	Modélisation avec UML	41
3.3	Détection des cultures	43
	Conclusion	43
4	Résultats et discussions	44
	Introduction	44
4.1	Résultat	44
4.1.1	Tests, résultats et évaluation	44
4.2	Quelques interfaces de l'application	48
4.3	Discussions	48
4.3.1	Points positifs	48
4.3.2	Points négatifs	50
4.3.3	Améliorations	50
	Conclusion	51
	Conclusion et perspectives	52
	Bibliographie et Webographie	53
A	Annexe	55
	Bibliographie	55

Introduction

Contexte et justification

En Afrique l'agriculture est la plus grande richesse que nous avons, ce qui fait d'elle notre levier de la transformation économique. Plus de la moitié des Africains dépendent de l'agriculture pour tout ou partie de leurs moyens d'existence. Partant de là, promouvoir la croissance agricole durable revient à accroître les revenus et à améliorer globalement les conditions de vie. Promouvoir la croissance agricole stimule également le développement économique dans les secteurs en amont et en aval. Il faut aussi considérer le considérable gisement d'emplois que représente le secteur Agricole.

Problématique

En Afrique l'Agriculture que nous pratiquons est définie par opposition aux exploitations dont la production repose principalement sur des ouvriers salariés qui servent de variable d'ajustement pour maximiser les revenus des actifs, alors que les exploitations familiales maximisent l'utilisation de la main-d'œuvre de la famille. Les techniques utilisées étant aussi précaires, donne une énorme difficulté de la mise en pratique d'une agriculture étendue sur une grande surface. En matière de la détermination de la superficie cultivée et de la détermination des rendements de la production agricole, les techniques utilisées prennent un temps énorme et requièrent un grand effort physique.

Depuis quelques années, grâce à l'évolution de la technologie, l'agriculture connaît un sort d'une grande amélioration. Ce qui a donné naissance à l'Agriculture de précision, qui est un principe de gestion des parcelles agricoles qui vise l'optimisation des rendements et des investissements, en cherchant à mieux tenir compte des variabilités des milieux et des conditions entre parcelles différentes ainsi qu'à des échelles intra-parcelles[11]. En parlant de technologie, nous avons l'intelligence artificielle qui est une branche de l'informatique qui a pour but de comprendre des entités intelligentes et d'en construire, et aussi les drones qui permettent de

survoler de grande surface très facilement.

Le sorgho est un genre de plantes à fleurs de la famille des graminées Poaceae [19]. Le sorgho est la cinquième céréale du monde en volume de production, après le maïs, le riz, le blé et l'orge. Il s'agit de la principale céréale pour de nombreuses populations à faible revenu vivant dans les régions tropicales semi-arides d'Afrique et d'Asie. Nous proposons et expérimentons une approche avec ML pour estimer le rendement agricole d'un champ de sorgho avant la récolte.

Le reste de ce document est organisé comme ceci. Le chapitre 1 présente les techniques traditionnelles d'estimation des rendements des cultures actuelles, la ML dans la détection d'objets et des techniques pour déterminer la taille réelle des objets sur une image prise par un drone. Le chapitre 2 présente le matériel et les méthodes. Le chapitre 3 présente la solution proposée. Enfin, le chapitre 4 fournit les résultats obtenus et une analyse de ces résultats.

Objectifs

Nous proposons et expérimentons une approche qui, grâce au Machine Learning, permet de déterminer le rendement agricole d'un champ avant la récolte. Notre travail a pour finalité de :

- Apprendre à la machine à reconnaître les céréales sur une image de champ agricole ;
- Déetecter le nombre approximatif de pieds de spéculature sur une image de champ agricole ;
- Faire une estimation des rendements de la production agricoles des céréales ;
- Aider à la prise de décision.

Organisation du travail

La suite de ce document est organisée comme suit. Le chapitre 1 fait l'état de l'art des techniques de l'estimation des rendements agricoles utilisées, du Machine Learning dans la détermination des formes et les techniques pour déterminer la taille réelle des objets sur une image. Le chapitre 2 présente le matériel ainsi que les méthodes que nous utilisons pour la mise en œuvre de la solution proposée. Le chapitre 3 détaille ladite solution. Enfin, nous présentons dans le chapitre 4 les résultats obtenus après les tests de notre solution ainsi que l'analyse de ces résultats.

Etat de l'art

Introduction

Dans ce chapitre nous présenterons la revue de littérature des techniques de l'estimation des rendements agricoles, le Machine Learning dans la détection des objets et la détermination de la taille réelle des objets sur une image.

1.1 Technique d'estimation des rendements Agricoles

Le rendement agricole est la quantité de produit récoltée sur une surface cultivée donnée. Il est souvent exprimé en quintaux métriques (1 q = 100 kg) par hectare pour les grains, ou en tonnes par hectare pour les produits riches en eau (racines et tubercules, fruits...).

1.1.1 Méthode traditionnelle

De manière traditionnelle on procède par les carrés de rendement pour determiner le rendement de la production. Le carré de rendement est un carré posé au hasard dans une parcelle pour y effectuer un comptage de tous les plants [1]. Les carrés de rendement sont confectionnés avec des tiges en fer rond de 1 m à 6 m de longueur et de 6 mm de diamètre soudées entre elles (voir Fig. 1.1). Ils sont posés dans la parcelle la veille de la récolte. Pour poser les carrés de rendement il y a trois étape à suivre :

- Repérer la parcelle de la culture à évaluer ;
- Determiner la superficie de la parcelle : deux méthodes sont utilisées. La première en utilisant un decamètre et mésurer chaque coté de la parcelle et ensuite procéder à la determination de la superficie. La deuxième méthode se fait grâce au GPS. Le GPS permet de

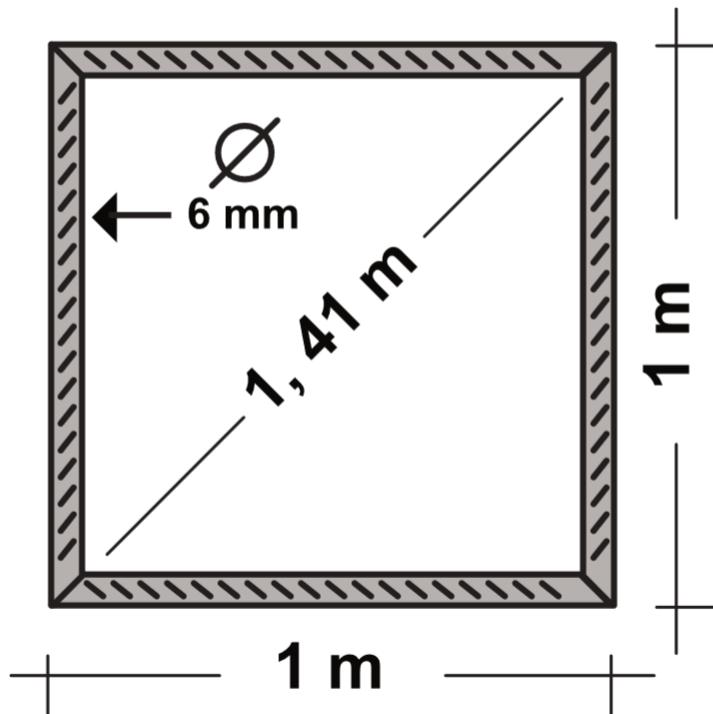


FIGURE 1.1 – Carré de rendement [1]

prendre les points limitrophe de la parcelle, et grâce à un logiciel, on détermine la superficie. Ensuite il faut élaborer le schéma de la parcelle.

- Identifier les diagonales de la parcelle et les points de pose.

Après ceci il faut poser 5 carrés par diagonal en évitant les bordures pour les parcelles d'un hectare. Pour les parcelles de moins d'un hectare il faut poser 2 à 3 carrés par diagonale avec un minimum de 4 carrés par parcelle. Pour chaque carré :

- Compter le nombre de plants ou de pieds par carré de rendement posé ;
- Récolter les produits (tubercules, bulbes ou fruits), les compter et les placer dans un contenant (sac ou sachet, seau, etc.) ;
- Tarer le peson ou la balance avec le contenant et peser les produits récoltés ;
- Reporter les données collectées sur la fiche de carré de rendement. Pour les récoltes échelonnées, additionner, sur le même carré, les données collectées de chaque récolte.

Pour connaître le rendement dans le carré de rendement, il faut diviser la quantité totale de récolte pesée en kg par la surface du carré de rendement.

$$Rendement_du_carre = \frac{Poids_total_de_la_recolte_dans_le_carre}{surface_du_carre} \quad (1.1)$$

Il est important, le plus souvent, que les rendements soient exprimés en kilogrammes par hectare (kg/ha). Cela permet de comparer les valeurs entre différents champs dans des endroits différents. Pour passer du rendement calculé dans le carré qui est exprimé en kg/m², il faut convertir en kg/ha.

1.1.2 Agriculture de précision

L'agriculture de précision permet d'adapter les pratiques agricoles en fonction de la variabilité spatiale et temporelle entre les champs, mais aussi à l'intérieur d'un même champ. La première étape pour pratiquer l'agriculture de précision consiste à mesurer la variabilité spatiale des rendements afin de produire des cartes permettant d'identifier les causes des variations (voir Fig. 1.2). Au préalable, l'acquisition des données existantes (historique, pédologie, photographies aériennes, etc.) fournit des indications sur les causes potentielles de variation. L'observation de cartes de rendement, combinée aux analyses de sols et aux autres renseignements disponibles, facilite l'interprétation des variations. Les causes de variation peuvent être associées aux caractéristiques naturelles ou aux pratiques culturales. Lorsque les facteurs responsables des variations sont connus, des correctifs peuvent être apportés ou des nouvelles pratiques culturales, incluant les applications à taux variables, peuvent être adoptées. Finalement, l'évaluation des modifications apportées se poursuit par la production des cartes de rendement[5].

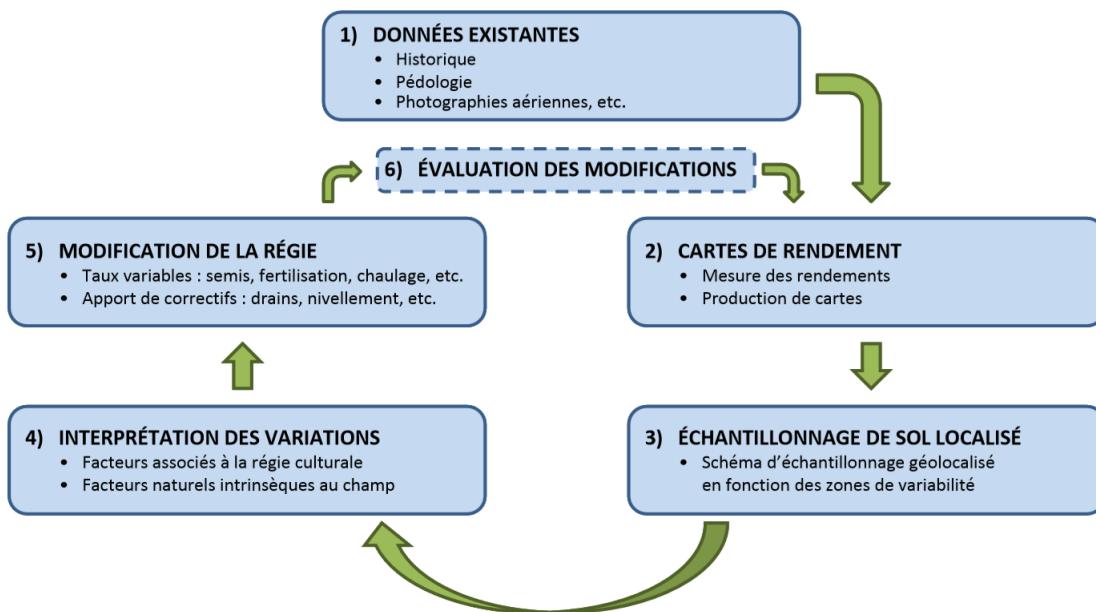


FIGURE 1.2 – Principales étapes pour pratiquer l'agriculture de précision [2]

Utilisation des capteurs de rendement

La précision des valeurs de rendement enregistrées par le capteur dépend de plusieurs facteurs. L'inspection et l'ajustement des composantes, ainsi que la calibration des capteurs, sont des étapes à ne pas négliger. Également, certaines consignes doivent être respectées pendant l'opération de la batteuse pour générer des cartes fiables.

Principales composantes

La mesure des rendements requiert des composantes qui sont situées à l'intérieur et à l'extérieur de la batteuse. Dans la cabine, une interface utilisateur (écran et clavier) est reliée à une console qui permet d'intégrer et de gérer les composantes (GPS et capteurs) du système (voir Fig. 1.3). Les données sont enregistrées sur une carte mémoire.



FIGURE 1.3 – Interface utilisateur dans la cabine [5]

Sur la batteuse, un capteur permet d'évaluer la masse de grains (à la sortie de l'élévateur) et un autre mesure l'humidité (voir Fig. 1.4). Le capteur d'humidité est localisé dans l'élévateur ou dans la vis à grain. Il permet de rapporter les rendements sur une base comparable (14 % d'humidité). Il y a également un capteur pour la hauteur de la table de coupe afin de démarrer ou d'interrompre l'enregistrement des données.

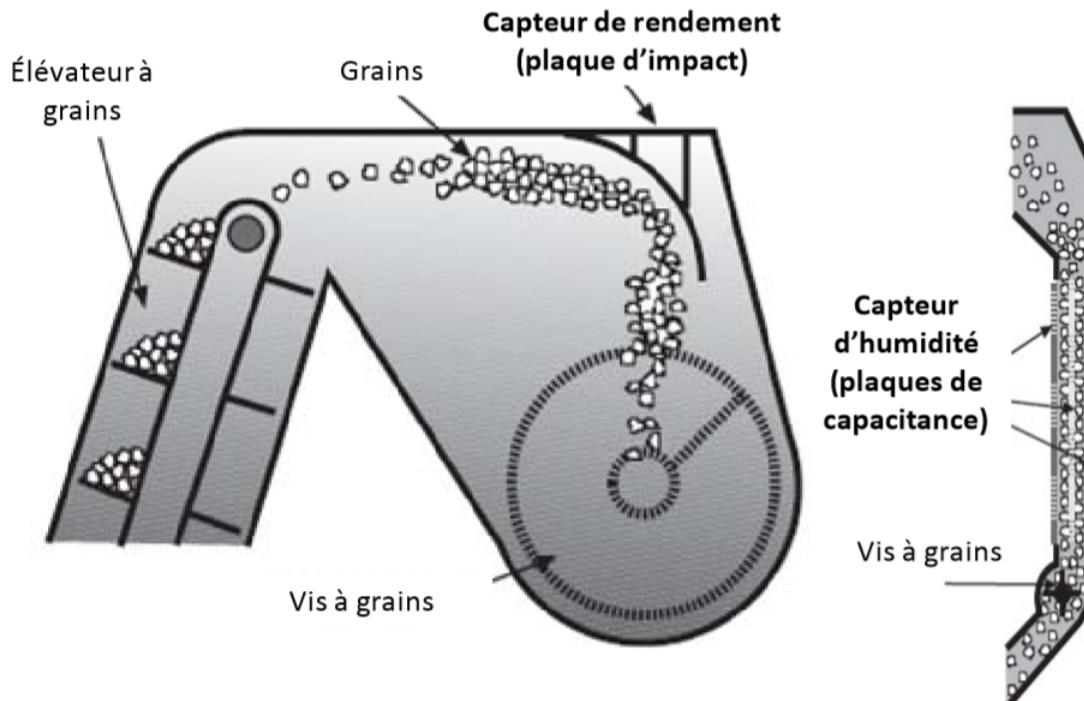


FIGURE 1.4 – Schéma illustrant un capteur de rendement et un capteur d'humidité [2]

Fonctionnement d'un capteur de rendement

La lecture des rendements par les capteurs se fait sur un intervalle de temps variable entre une et trois secondes. Par exemple, une lecture chaque deux secondes correspond à environ 500 lectures pour une superficie d'un hectare [2]. Cette variable peut être ajustée par l'opérateur de la machinerie. La largeur et la vitesse de la batteuse déterminent le délai entre la coupe et l'arrivée dans le chariot à grain (environ 10 à 15 secondes). L'ordinateur corrige l'effet de délai et amoindrit les changements brusques comme dans les bouts de champs. Plusieurs facteurs peuvent interrompre l'acquisition de données par le capteur de rendement. On peut penser à l'ajustement du niveau critique de la table de récolte. En effet, il est possible pour l'opérateur de la batteuse de déterminer à quelle hauteur la table de coupe envoie un signal d'arrêt d'enregistrement, ce qui empêche l'acquisition de nombreuses données sans fondement.

1.2 Détection d'objet

1.2.1 Machine Learning (ML)

Le Machine Learning est un champ d'étude de l'intelligence artificielle qui se base sur des approches statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune [12]. Pour apprendre et se développer, les ordinateurs ont toutefois

besoin de données à analyser et sur lesquelles s'entraîner.

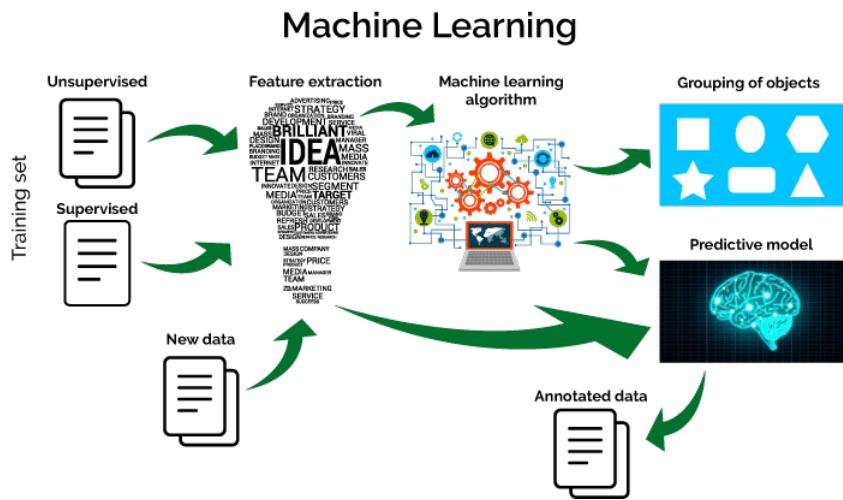


FIGURE 1.5 – Schéma d'utilisation du Machine Learning [13]

Les algorithmes d'apprentissage peuvent se catégoriser selon le mode d'apprentissage qu'ils emploient :

Apprentissage supervisé

Si les données sont étiquetées (c'est-à-dire que la réponse à la tâche est connue pour ces données) il s'agit de l'apprentissage supervisé. Les systèmes du ML apprennent comment combiner des entrées pour formuler des prédictions efficaces sur des données qui n'ont encore jamais été observées. On peut par exemple donner au système une liste de profils clients contenant des habitudes d'achat, et expliquer à l'algorithme lesquels sont des clients habituels et lesquels sont des clients occasionnels. Une fois l'apprentissage terminé, l'algorithme devra pouvoir déterminer tout seul à partir d'un profil client à quelle catégorie celui-ci appartient [12].

Apprentissage non supervisé

Il s'agit, pour un logiciel, de trouver des structures sous-jacentes à partir de données non étiquetées. Puisque les données ne sont pas étiquetées, l'apprentissage non supervisé est beaucoup plus complexe puisqu'ici le système va devoir détecter les similarités dans les données qu'il reçoit et les organiser en fonction de ces dernières. Cette façon de travailler présente un avantage indéniable en ce sens que la phase de catégorisation de l'apprentissage supervisé est un processus gourmand en ressources humaines. Son élimination, ou tout du moins sa réduction retire un frein à l'implémentation de la technologie [12].

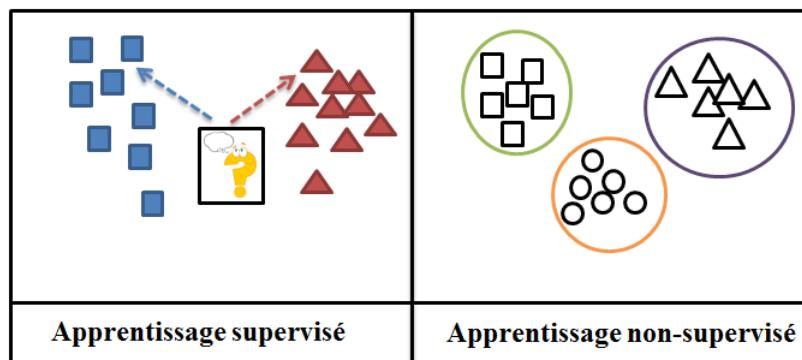


FIGURE 1.6 – Apprentissage supervisé et l'apprentissage non supervisé

Apprentissage semi-supervisé

Les systèmes à apprentissage semi-supervisé combinent à la fois le principe d'apprentissage supervisé et celui non supervisé. L'intérêt de cette combinaison provient du fait que l'étiquetage de données nécessite l'intervention d'un utilisateur humain. Lorsque les jeux de données deviennent très grands, cette opération peut s'avérer fastidieuse. Dans ce cas, l'apprentissage semi-supervisé, qui ne nécessite que quelques étiquettes, revêt un intérêt pratique évident [12].

Apprentissage partiellement supervisé

Entraînement d'un modèle avec des données où seulement certains des exemples d'apprentissage sont étiquetés. L'une des techniques d'apprentissage partiellement supervisé consiste à déduire les étiquettes des exemples sans étiquette, puis à entraîner le modèle avec les étiquettes déduites afin de créer un nouveau modèle. L'apprentissage partiellement supervisé peut être utile si les étiquettes sont coûteuses, mais que les exemples sans étiquette abondent [14].

Apprentissage par renforcement

L'apprentissage par renforcement consiste, pour un agent autonome, à apprendre les actions à prendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps. L'agent est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, un comportement décisionnel (appelé stratégie ou politique, et qui est une fonction associant à l'état courant l'action à exécuter) optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.

L'apprentissage par renforcement (RL pour Reinforcement Learning) fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre, à partir d'expériences successives, ce qu'il convient de faire de façon à trouver la meilleure solution. Dans un tel problème, on dit qu'un «agent» (l'algorithme, au sens du code et des variables qu'il utilise)

interagit avec «l'environnement» pour trouver la solution optimale. L'apprentissage par renforcement diffère fondamentalement des problèmes supervisés et non supervisés par ce côté interactif et itératif : l'agent essaie plusieurs solutions (on parle « d'exploration »), observe la réaction de l'environnement et adapte son comportement (les variables) pour trouver la meilleure stratégie (il « exploite » le résultat de ses explorations) [15].

1.2.2 Deep Learning

Le Deep Learning (l'apprentissage en profondeur) est un type particulier du Machine Learning qui permet d'obtenir une puissance et une souplesse considérables en apprenant à représenter le monde comme une hiérarchie imbriquée de concepts ou d'abstraction. Le Deep Learning s'inspire de la fonctionnalité de nos cellules cérébrales appelée réseau neuronal artificiel. Il prend simplement des connexions de données entre tous les neurones artificiels et les ajuste en fonction du modèle de données. Plus de neurones sont nécessaires si la taille des données est grande. Il intègre automatiquement l'apprentissage à plusieurs niveaux d'abstraction, permettant ainsi à un système d'apprendre le mappage de fonctions complexes sans dépendre d'aucun algorithme spécifique.

Les techniques du Deep Learning constituent une classe d'algorithmes d'apprentissage automatique qui :

- utilisent différentes couches d'unité de traitement non linéaire pour l'extraction et la transformation des caractéristiques ; chaque couche prend en entrée la sortie de la précédente ; les algorithmes peuvent être supervisés ou non supervisés, et leurs applications comprennent la reconnaissance de modèles et la classification statistique ;
- fonctionnent avec un apprentissage à plusieurs niveaux de détail ou de représentation des données ; à travers les différentes couches, on passe de paramètres de bas niveau à des paramètres de plus haut niveau, où les différents niveaux correspondent à différents niveaux d'abstraction des données.

Ces architectures permettent aujourd'hui de conférer du sens à des données en leur donnant la forme d'images, de sons ou de textes [16].

1.2.3 Généralité sur les images

Une image est plutôt difficile à décrire d'une façon générale. Une image est une représentation du monde. En traitement d'image, la majorité du temps, on considère qu'il s'agit d'une fonction mathématique de RxR dans R où le couplet d'entrée est considéré comme une position spatiale, le singleton de sortie comme l'intensité (couleur ou niveaux de gris) du phénomène physique. Il arrive cependant que l'image soit dite "3D" donc la fonction est de RxRxR dans R.

Les images couleurs peuvent être représentées soit par trois images représentant les trois couleurs fondamentales, soit par une image de RxR dans RxRxR. L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle, ou calculé à partir d'une description interne de la scène à représenter. L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants :

Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image [6].

Résolution

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels affichables horizontalement ou verticalement sur un moniteur ; plus grand est ce nombre, meilleure est la resolution.

Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur [6].

Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d'une image trop claire ou d'une image trop foncée. Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant [6].

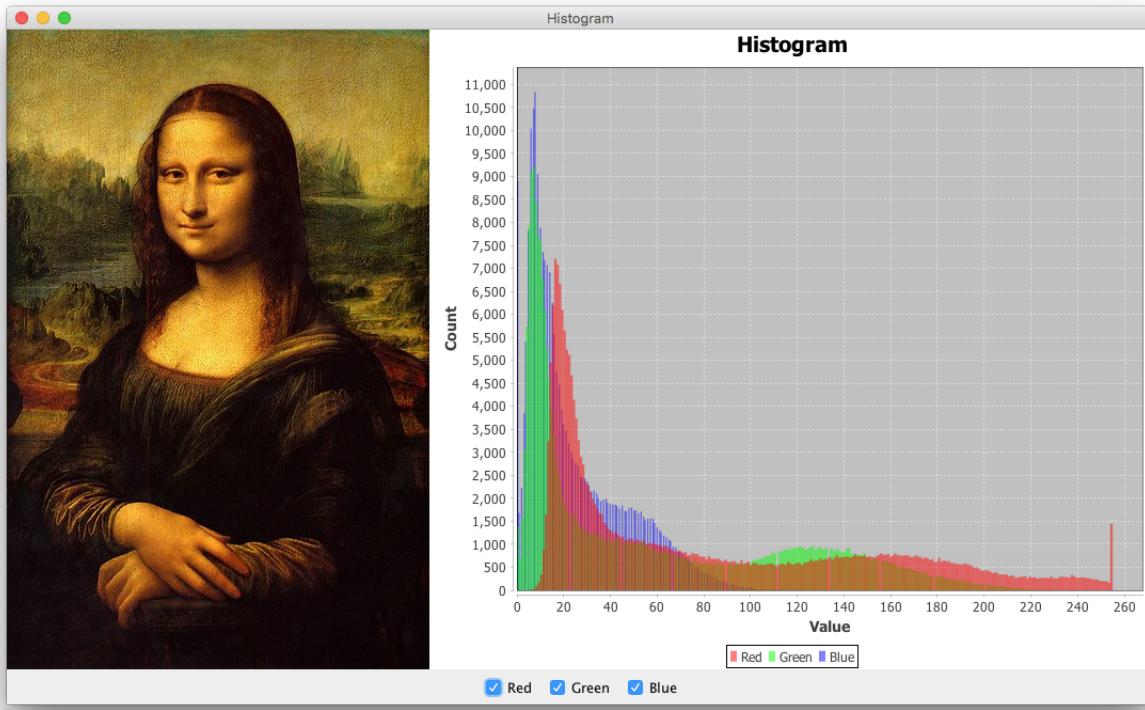


FIGURE 1.7 – Exemple d'histogramme

Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet [6]. Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes) ;
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir ; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si L_1 et L_2 sont les degrés de luminosité respectivement de

deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport [6]

$$C = \frac{L1 - L2}{L1 + L2} \quad (1.2)$$

Images à niveaux de gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Donc pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondant.

Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la "couleur" de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux [6].

Images en couleurs

Même s'il est parfois utile de pouvoir représenter des images en noir et blanc, les applications multimédias utilisent le plus souvent des images en couleurs. La représentation des couleurs s'effectue de la même manière que les images monochromes avec cependant quelques particularités. En effet, il faut tout d'abord choisir un modèle de représentation. On peut représenter les couleurs à l'aide de leurs composantes primaires. Les systèmes émettant de la lumière (écrans d'ordinateurs,...) sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (modèle R.V.B.) [6].

Les différents formats d'images

On peut classer les images en deux formats :

Format vectorielle

Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique.

Par exemple, un cercle est décrit par une information du type (cercle, position du centre, rayon). Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans [7].

Format matricielle

Une image matricielle est formée d'un tableau de points ou pixels. Plus la densité des points sont élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée. Corrélativement la place occupée en mémoire et la durée de traitement seront d'autant plus grandes.

Les images vues sur un écran de télévision ou une photographie sont des images matricielles.

On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanner [7]. Parmi ces formats on peut citer :

- **BMP (BitMap)** : Le format BMP est le format par défaut du logiciel Windows. C'est un format matriciel. Les images ne sont pas compressées. Son logiciel d'origine.
- Le format **EPS** : matriciel n'est pas très différent du EPS vectoriel. En fait seules les données contenues dans le fichier sont différentes. Ainsi un logiciel de retouche de photos tel que Photoshop permet l'importation, la modification et l'exportation de fichiers en format EPS.
- **GIF (GraphicalInterchange Format)** : Le format GIF est un format qui a ouvert la voie à l'image sur le World Wide Web. C'est un format de compression qui n'accepte que les images en couleurs indexés codé sur 8 bits, C'est un format qui perd beaucoup de son marché suite à une bataille juridique concernant les droits d'utilisation sur Internet.
- **JPEG (Joint Photographic Experts Group)** : Les images JPEG sont des images de 24 bits. C'est-à dire qu'elles peuvent afficher un spectre de 16 millions de couleurs. C'est la meilleure qualité d'images disponible.
- **PCX** : Le format PCX est utilisé par le logiciel Paintbrush sous Windows. C'est un format matriciel. **TIFF (Tagged Image File Format)** : Le format TIFF, conçu à l'origine par la compagnie Aldus est un format matriciel. Conçu au départ pour n'accepter que les images en RGB, ce format permet de coder des images CYMK.
- **PBM (Portable BitMap)** : Commençons par le plus simple : le format pbm. Ce format permet de stocker des images en noir et blanc
- **PGM (Portable GrayMap)** : Le format pgm permet de représenter des images en niveaux de gris dont les pixels ont des valeurs entières comprises entre 0 (noir) et 255 (blanc). La valeur de chacun des pixels est enregistrée dans le fichier au format ASCII.
- **PPM (Portable PixMap)** : Le format ppm concerne les images couleurs. Chaque pixel a pour valeur un triple (R, G, B) composé d'une composante rouge, verte et bleue. Chaque composante est représentée par un entier pouvant prendre ses valeurs entre 0

Métadonnées

Une métadonnée est littéralement une donnée sur une donnée. Plus précisément, c'est un ensemble structuré d'informations décrivant une ressource quelconque. Les ressources décrites par des métadonnées ne sont pas nécessairement sous forme digitale : un catalogue de bibliothèque ou de musée contient aussi des métadonnées décrivant les ressources que sont les ouvrages de la bibliothèque ou les objets du musée [7]. Une métadonnée peut être utilisée à des fins diverses :

- la description et la recherche de ressources ;
- la gestion de collections de ressources ;
- la préservation des ressources.

Observons quelques exemples de métadonnées qu'on peut récupérer sur un fichier image :

Property	Value
Description	
Title	DCIM\100MEDIA\DJ1_0072.JPG
Subject	DCIM\100MEDIA\DJ1_0072.JPG
Rating	★ ★ ★ ★ ★
Tags	v01.07.1641; 12.0; v1.0.0
Comments	Type=N; Mode=P; DE=None
Origin	
Authors	
Date taken	11/26/2018 11:34 AM
Program name	v01.07.1641
Date acquired	
Copyright	
Image	
Image ID	
Dimensions	5472 x 3648
Width	5472 pixels
Height	3648 pixels
Horizontal resolution	72 dpi
Vertical resolution	72 dpi

Property	Value
Bit depth	24
Compression	
Resolution unit	2
Color representation	sRGB
Compressed bits/pixel	3.3020853371550221
Camera	
Camera maker	DJI
Camera model	FC6310
F-stop	f/5
Exposure time	1/240 sec.
ISO speed	ISO-100
Exposure bias	+1 step
Focal length	9 mm
Max aperture	2.97
Metering mode	Center Weighted Average
Subject distance	0 mm
Flash mode	No flash function
Flash energy	
35mm focal length	24

Property	Value
Exposure program	Normal
Saturation	Normal
Sharpness	Normal
White balance	Auto
Photometric interpretation	
Digital zoom	
EXIF version	0230
GPS	
Latitude	10.18.28.283199999998132
Longitude	1.22.24.3592999999999999
Altitude	463.839
File	
Name	DJ1_0072.JPG
Item type	JPG File
Folder path	C:\Users\Judicael\Documents
Date created	12/6/2018 3:25 PM
Date modified	12/6/2018 10:57 AM
Size	8.12 MB
Attributes	A

FIGURE 1.8 – Exemple de métadonnées sur un fichier image

1.2.4 Classification d'image par réseaux neurones convolutifs

Pour bien comprendre les réseaux de neurones convolutifs, il est important de connaître les bases des réseaux de neurones. Les principaux éléments à retenir sont :

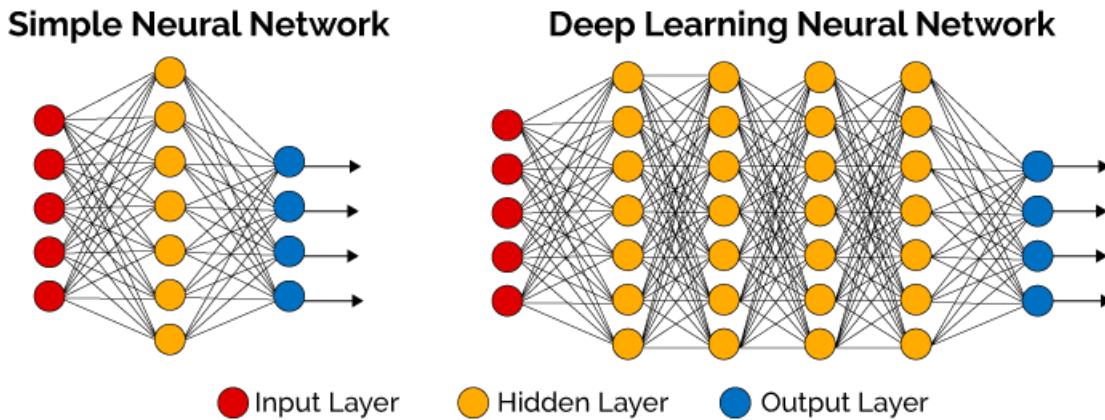


FIGURE 1.9 – Réseau de neurone du Deep Learning

- Un réseau de neurones est un système composé de neurones, généralement répartis en plusieurs couches connectées entre elles
- Un tel système s'utilise pour résoudre divers problèmes statistiques, mais nous nous intéressons ici qu'au problème de classification (très courant). Dans ce cas, le réseau calcule à partir de l'entrée un score (ou probabilité) pour chaque classe. La classe attribuée à l'objet en entrée correspond à celle de score le plus élevé [9] ;
- Chaque couche reçoit en entrée des données et les renvoie transformées. Pour cela, elle calcule une combinaison linéaire puis applique éventuellement une fonction non-linéaire, appelée fonction d'activation. Les coefficients de la combinaison linéaire définissent les paramètres (ou poids) de la couche [9] ;
- Un réseau de neurones est construit en empilant les couches : la sortie d'une couche correspond à l'entrée de la suivante [9] ;
- Cet empilement de couches définit la sortie finale du réseau comme le résultat d'une fonction différentiable de l'entrée [9] ;
- La dernière couche calcule les probabilités finales en utilisant, en général, pour fonction d'activation la fonction logistique (classification binaire) ou la fonction softmax (classification multi-classes) [9]

$$\sigma(w_0 + \sum_{j=1}^p w_j x_j) = \frac{1}{1 + \exp^{-(w_0 + \sum_{j=1}^p w_j x_j)}} \text{fonction sigmoïde pour la régression logistique} \quad (1.3)$$

$$\sigma(u) = \frac{e^u}{1 + e^u} \text{fonction softmax} \quad (1.4)$$

- Une fonction de perte (loss function) est associée à la couche finale pour calculer l'erreur de classification. Il s'agit en général de l'entropie croisée [9]. Dans le cas binaire l'entropie croisée est définie par

$$\text{Erreur}(f(x^{(i)}), y^{(i)}) = -y^{(i)} \log f(x^{(i)}) - (1 - y^{(i)}) \log(1 - f(x^{(i)})). \quad (1.5)$$

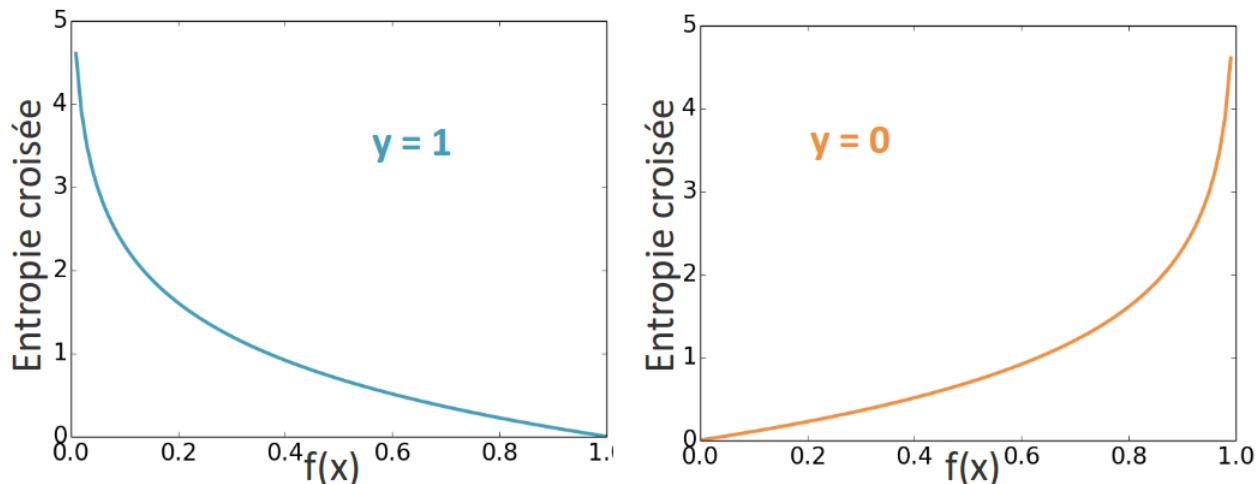


FIGURE 1.10 – Classification : Entropie croisée [8]

Quand $y = 0$, l'entropie croisée est d'autant plus élevée que $f(x)$ est proche de 1. Réciproquement, quand $y = 1$, l'entropie croisée est d'autant plus grande que la prédiction est proche de 0.

- Les valeurs des poids des couches sont appris par rétropropagation du gradient : on calcule progressivement (pour chaque couche, en partant de la fin du réseau) les paramètres qui minimisent la fonction de perte régularisée. L'optimisation se fait avec une descente du gradient stochastique [9].

Les réseaux de neurones convolutifs (CNN) désignent une sous-catégorie de réseaux de neurones : ils présentent donc toutes les caractéristiques listées ci-dessus. Cependant, les CNN sont spécialement conçus pour traiter des images en entrée. Leur architecture est alors plus spécifique : elle est composée de deux blocs principaux [10].

Le premier bloc fait la particularité de ce type de réseaux de neurones, puisqu'il fonctionne comme un extracteur de caractéristiques. Pour cela, il effectue du template matching en appliquant des opérations de filtrage par convolution. La première couche filtre l'image avec plusieurs noyaux de convolution, et renvoie des "feature maps", qui sont ensuite normalisées (avec une fonction d'activation) et/ou redimensionnées [10].

Ce procédé peut être réitéré plusieurs fois : on filtre les features maps obtenues avec de nouveaux noyaux, ce qui nous donne de nouvelles features maps à normaliser et redimensionner,

et qu'on peut filtrer à nouveau, et ainsi de suite. Finalement, les valeurs des dernières feature maps sont concaténées dans un vecteur. Ce vecteur définit la sortie du premier bloc, et l'entrée du second [10].

Le second bloc n'est pas caractéristique d'un CNN : il se retrouve en fait à la fin de tous les réseaux de neurones utilisés pour la classification. Les valeurs du vecteur en entrée sont transformées (avec plusieurs combinaisons linéaires et fonctions d'activation) pour renvoyer un nouveau vecteur en sortie. Ce dernier vecteur contient autant d'éléments qu'il y a de classes : l'élément i représente la probabilité que l'image appartienne à la classe i . Chaque élément est donc compris entre 0 et 1, et la somme de tous vaut 1. Ces probabilités sont calculées par la dernière couche de ce bloc (et donc du réseau), qui utilise une fonction logistique (classification binaire) ou une fonction softmax (classification multi-classe) comme fonction d'activation [10].

Comme pour les réseaux de neurones ordinaires, les paramètres des couches sont déterminés par rétropropagation du gradient : l'entropie croisée est minimisée lors de la phase d'entraînement. Mais dans le cas des CNN, ces paramètres désignent en particulier les features des images [10].

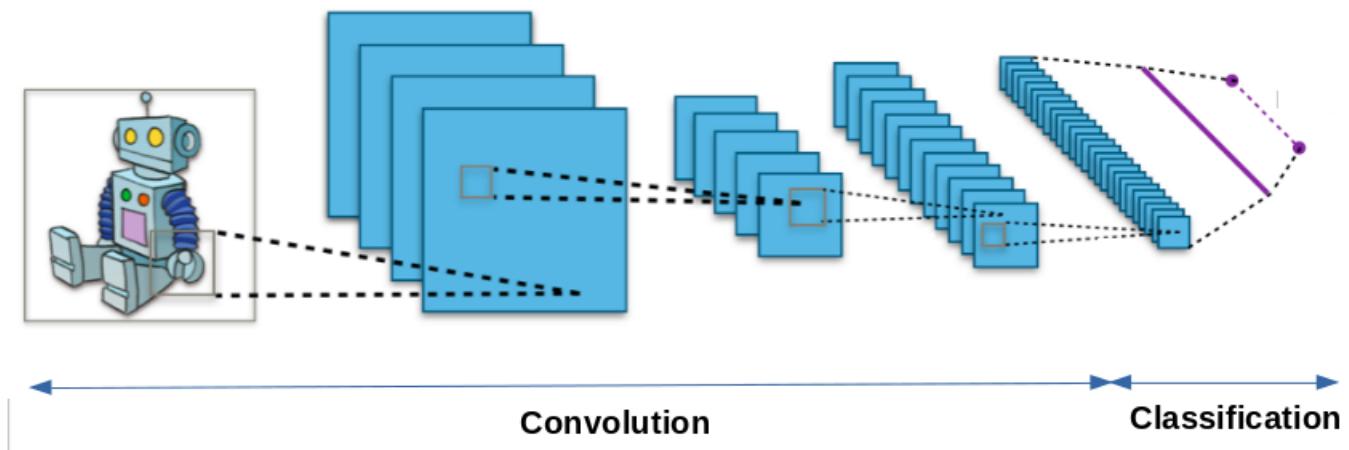


FIGURE 1.11 – Architecture d'un réseau de neurones convolutifs

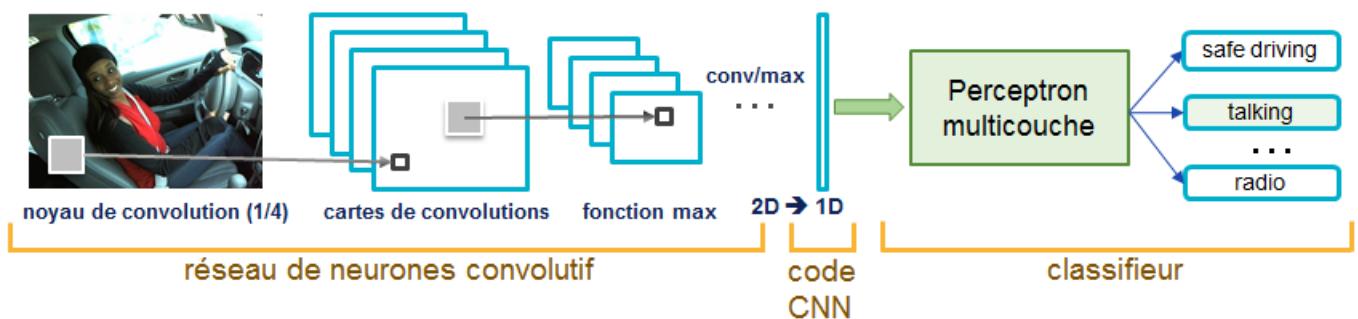


FIGURE 1.12 – Architecture d'un réseau de neurones convolutifs

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche fully-connected [9].

- **La couche de convolution** : est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Son but est de repérer la présence d'un ensemble de features dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une fenêtre représentant la feature sur l'image, et de calculer le produit de convolution entre la feature et chaque portion de l'image balayée. Une feature est alors vue comme un filtre : les deux termes sont équivalents dans ce contexte. La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux features que l'on souhaite retrouver dans les images. On obtient pour chaque paire (image, filtre) une carte d'activation, ou feature map, qui nous indique où se situent les features dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la feature [9].
- **La couche de pooling** : Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling. L'opération de pooling consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulières, puis on garde au sein de chaque cellule la valeur maximale. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2×2 pixels qui ne se chevauchent pas, ou des cellules de taille 3×3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent donc). On obtient en sortie le même nombre de feature maps qu'en entrée, mais celles-ci sont bien plus petites. La couche de pooling permet de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage. Les valeurs maximales sont repérées de manière moins exacte dans les feature maps obtenues après pooling que dans celles reçues en entrée – c'est en fait un grand avantage ! En effet, lorsqu'on veut reconnaître un chien par exemple, ses oreilles n'ont pas besoin d'être localisées le plus précisément possible : savoir qu'elles se situent à peu près à côté de la tête suffit ! Ainsi, la couche de pooling rend le réseau moins sensible à la position des features : le fait qu'une feature se situe un peu plus en haut ou en bas, ou même qu'elle ait une orientation légèrement différente ne devrait pas provoquer un changement radical dans la classification de l'image [9].
- **La couche de correction ReLU** : ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par

$$\text{ReLU}(x) = \max(0, x). \quad (1.6)$$

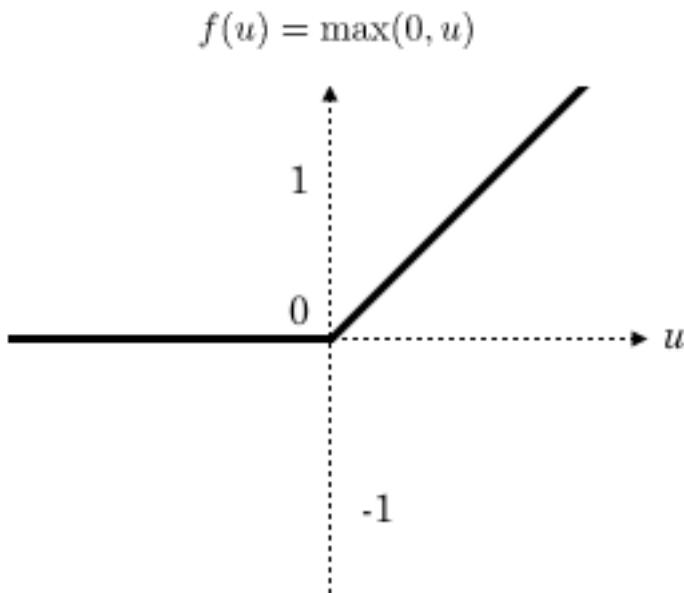


FIGURE 1.13 – Allure de la fonction ReLU [9]

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

- **La couche fully-connected** : La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non – elle n'est donc pas caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée. La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N , où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

Par exemple, si le problème consiste à distinguer les chats des chiens, le vecteur final sera de taille 2 : le premier élément (respectivement, le deuxième) donne la probabilité d'appartenir à la classe "chat" (respectivement "chien"). Ainsi, le vecteur $[0.9 \ 0.1]$ signifie que l'image a 90% de chances de représenter un chat. Chaque valeur du tableau en entrée "vote" en faveur d'une classe. Les votes n'ont pas tous la même importance : la couche leur accorde des poids qui dépendent de l'élément du tableau et de la classe. Pour calculer les probabilités, la couche fully-connected multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (logistique si $N=2$, softmax si $N>2$) : Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme fully-connected [9].

1.3 Détermination de la taille réelle des objets au sol sur une image aérienne

Pour déterminer la taille réelle d'un objet sur une image, il y a deux cas qui se présentent.

Cas 1

Le premier cas est lorsqu'on connaît la taille réelle d'un objet se trouvant sur l'image. A ce moment cet objet servira de référence en vue de déterminer la taille des autres objets sur l'image. Cet objet de référence doit avoir deux propriétés :

- Propriété n ° 1 : nous devrions connaître les dimensions de cet objet (en termes de largeur ou de hauteur) dans une unité mesurable (telle que les millimètres, les pouces, etc.).
- Propriété n ° 2 : nous devrions pouvoir trouver facilement cet objet de référence dans une image, en fonction de l' emplacement de l'objet (tel que l'objet de référence toujours placé dans le coin supérieur gauche d'une image) ou via les apparences (comme couleur ou forme distincte, unique et différente de tous les autres objets de l'image). Dans les deux cas, notre référence devrait être identifiable de manière unique .

Passons au calcul du ratio pixels par métrique (nombre de pixels pour une métrique donnée) :

$$\text{pixel_par_metrique} = \frac{\text{largeur_de_l'objet_dans_l'image}}{\text{largeur_relle_de_l'objet}} \quad (1.7)$$

Ensuite pour connaître la taille des autres objets on effectue ce calcul :

Soit l'objet X

$$\text{largeur_objet_X} = \frac{\text{largeur_de_l'objet_X_sur_l'image}}{\text{pixel_par_metrique}} \quad (1.8)$$

Cependant, tous nos résultats ne sont pas parfaits.

Cas 2

Ici prenons le cas d'une vue aérienne prise par un drone ou un photographe dans un avion volant avec un appareil fournissant les coordonnées GPS de la position de prise de vue. En prenant le cas du photographe, à tout ce que nous allons dire ici, il faut appliquer la position angulaire pour un vrai résultat. Dans notre cas c'est la vue prise par les drones qui nous interesse. Ses vue sont prise avec la camera perpendiculaire au sol.

Alors, pour connaître la taille d'un objet sur l'image prise par un drone, intervient la distance d'échantillonnage au sol (GSD) *Ground Sampling Distance en anglais*.

Les GSD dans les directions x et y, GSDx et GSDy, donnent les dimensions de l'empreinte au sol

d'un písel dans la matrice de plans focaux de la caméra. Le choix de la distance d'échantillonnage au sol est basé sur la taille du plus petit objet que l'utilisateur souhaite visualiser.

Soit H la hauteur (altitude) à laquelle le drone vole au dessus du sol pour sa mission.

Ainsi, il peut être utile d'utiliser l'équation suivante pour estimer la GSD en fonction de la résolution du capteur et du champ de vision (FOV). Dans cette équation, le champ de vue doit être réglé sur le champ de vision horizontal de la caméra et N doit être le nombre de pixel dans une rangée du réseau de plan focal (on peut également utiliser le champ de vision vertical et le nombre de pixels dans une seule image).

$$GSD = \frac{2 * H * \tan(\frac{FOV}{2})}{N} \quad (1.9)$$

Nous considérons ici le cas d'une vue prise par un drone avec un appareil fournissant les coordonnées GPS de la position de prise de vue.

La distance entre échantillons au sol (GSD) est la distance entre les points centraux de chaque échantillon prélevé sur le sol. Chaque échantillon est un pixel. En termes plus simples, le GSD est la taille de chaque pixel au sol [17]. Nous avons un diagramme montrant tous les facteurs pertinents pour le calcul de GSD. Les chiffres sont accessibles au public pour les drones avec lesquels vous allez travailler.

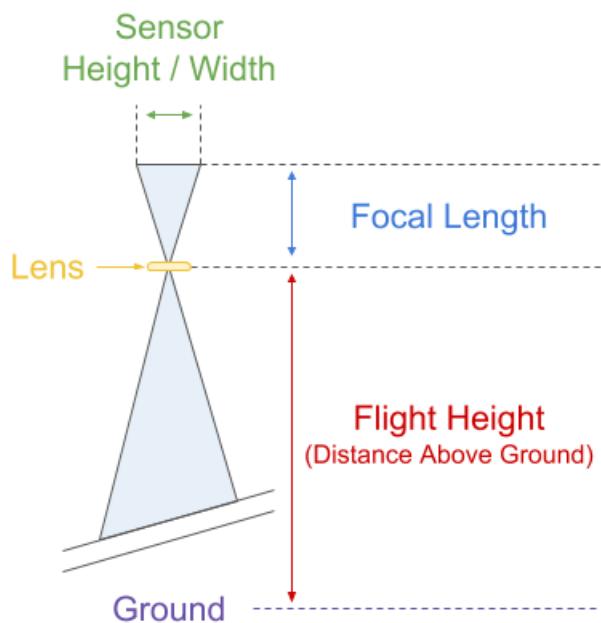


FIGURE 1.14 – Schéma montrant tous les facteurs pertinents pour le calcul de la GSD [17]

Lorsque nous projetons chaque pixel sur le sol, ils ne sont pas parfaitement carrés. Ainsi,

lorsque nous calculons le GSD, nous utilisons le plus grand des éléments suivants :

$$GSD_h = \frac{FlightHeight * SensorHeight}{FocalLength * ImageHeight} \quad (1.10)$$

$$GSD_w = \frac{FlightHeight * SensorWidth}{FocalLength * ImageWidth} \quad (1.11)$$

La GSD, une fois déterminée, permettra de déterminer la taille approximative des objets sur une images prise par un drone.

Alors pour calculer la taille d'un objet on effectue l'opération suivante :

$$largeur_de_l'objet = GSD * largeur_de_l'objet_sur_l'image \quad (1.12)$$

Conclusion

Beaucoup des recherches ont été menées pour améliorer la production agricole. Le Machine Learning et notamment le Deep Learning viennent apporter une grande aide dans l'avancement de l'agriculture de précision. Nous avons décidé d'utiliser pour les modèles d'apprentissage du Machine Learning pour améliorer les techniques d'estimation des rendement agricoles. Dans le chapitre suivant, nous verrons les matériels nécessaires pour mettre en place notre propre système dans l'estimation des rendements agricoles basée sur le Machine Learning. Nous détaillerons également des méthodes que nous allons utiliser dans chaque module du système.

Chapitre 2

Matériel et méthodes

Introduction

Le système d’information que nous proposons pour améliorer les techniques d’estimation des rendements agricoles sitées dans l’état de l’art, est constitué de deux systèmes :

- L’utilisation des drones pour capturer les images des champs agricoles, et
- Une application desktop qui se chargera d’effectuer l’estimation de ses rendements. Elle peut être supervisée ou non supervisées.

Dans ce chapitre, nous faisons la description du matériel utilisé tout au long de nos travaux ainsi que les outils de travail. Nous détaillons ensuite la manière dont nous avons préparé notre environnement de travail.

2.1 Matériel

Pour notre implémentation et nos tests, nous avons travaillé sur un ordinateur présentant les caractéristiques ci-après :

- Système d’exploitation : Windows 10 ;
- Type d’architecture du système d’exploitation : 64 bits ;
- Processeur : Intel® Core(TM) i5-5800U CPU @ 2.30GHz ;
- Mémoire vive : 16 GB ;
- Carte graphique : NVIDIA GeForce GTX 1050 4GB.

2.1.1 DJI Phantom 4 Pro

Le DJI Phantom 4 Pro¹ est un drone professionnel haut de gamme.

Le Phantom 4 Pro peut être utilisé pour des prises de vue professionnelles mais également pour créer des images de photogrammétrie 3D. Pour des images 3D parfaites, le GPS, le système de pilote automatique et la caméra du drone doivent fonctionner de manière transparente. Le Phantom 4 Pro fait tout à la perfection. Les images que nous avons utilisées sont des vues aériennes prise par le drone DJI Phantom 4 Pro.



FIGURE 2.1 – DJI Phantom 4 Pro utilisé pour les prises de vues

2.1.2 Langage de programmation Python

Python est un langage portable, dynamique, extensible, gratuit, qui permet (sans l'imposer) une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles [18]. En outre Python est un langage de choix lorsqu'il s'agit de Data Science. En outre, Le 31 juillet, l'Institute of Electrical and Electronics Engineers (IEEE) a publié son classement annuel des meilleurs langages de programmation. Il s'agit de son cinquième classement après ceux de 2014, 2015, 2016 et 2017. Dans celui de l'année dernière, c'est Python qui a pris la tête du classement général en dépassant Java et C. Et cette année, le langage de programmation de Guido Van Rossum conforte sa place de leader². L'explosion du Machine Learning et notamment du Deep Learning y est pour quelque chose. Son choix pour l'implémentation de notre système se justifie également du fait de l'utilisation de l'outil majeur dans le monde du Deep Learning que nous verrons dans la section suivante.

¹<https://www.dxomark.com/Cameras/DJI/Phantom4-Pro—Specifications>

²<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>

2.1.3 OpenCV

OpenCV(Open Source Computer Vision)est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel. Initialement développé par Intel, il a ensuite été soutenu par Willow Garage puis Itseez(qui a ensuite été acquis par Intel). La bibliothèque est multi-plateforme et libre d'utilisation sous la licence BSD open-source. Elle nous a permis de traité les images afin de tétourer les objets détectés sur les images. Nous avons utilisé la version 3.4.2

2.1.4 TensorFlow

La création de modèles d'apprentissage automatique précis, capables d'identifier et de localiser plusieurs objets dans une seule image, demeurait un défi fondamental en vision par ordinateur. Mais, avec les récents progrès de l'apprentissage en profondeur, les applications de détection d'objets sont plus faciles à développer que jamais. Tensorflow est le framework de formation en machine à code source libre de Google pour la programmation de flux de données dans diverses tâches. Les noeuds dans le graphique représentent des opérations mathématiques, tandis que les bords du graphique représentent les tableaux de données multidimensionnels (Tensor) communiqués entre eux.

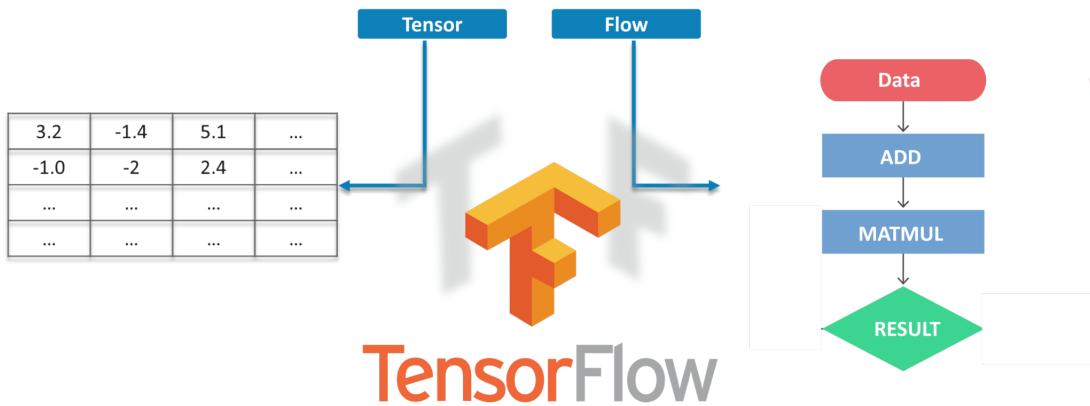


FIGURE 2.2 – Principe de fonctionnement de TensorFlow

TensorFlow propose une architecture facile, rapide et performante pour mettre en place un modèle d'apprentissage pour la detection d'objet sur les images. Ceci justifie notre choix de cette technologie. Nous avons utilisé la version 1.12.0.

2.1.5 LabelImg

LabelImg³ est un outil d'annotation d'image graphique. Il est écrit en Python et utilise Qt pour son interface graphique. Les annotations sont enregistrées sous forme de fichiers XML au

³<https://github.com/tzutalin/labelImg>

format PASCAL VOC, format utilisé par ImageNet . Besdies, il supporte également le format YOLO. Dans notre projet il nous a permis d'entourer les sorghos sur les images afin de faire apprendre à la machine.

2.1.6 Java

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy, présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détiennent et maintient désormais Java. Java est un langage portable et à des outils facile à prendre en main. Nous l'avons utilisé pour faire le backend de notre logiciel car nous le maîtrisons mieux. Nous avons utilisé la version 1.8 de Java et la version 2.1.0 du framework Spring boot.

2.1.7 Angular

Angular est un cadriel (framework) coté client open source basé sur TypeScript dirigée par l'équipe du projet Angular à Google et par une communauté de particuliers et de sociétés. Il nous a permis d'implémenter le frontend de notre logiciel. Nous avons utilisé la version 7.0.1.

2.2 Méthodes

2.2.1 Architecture logique de la solution

La solution proposée se réalise en trois phases (voir Fig. 2.3).

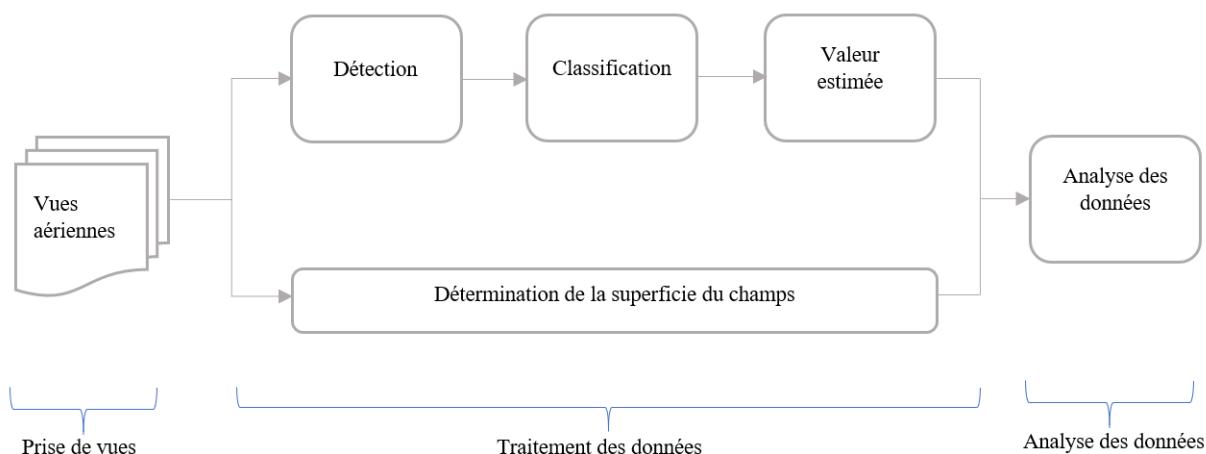


FIGURE 2.3 – Architecture logique de la solution

- **Prise de vues :** à ce niveau il faut prendre des vues aériennes du champ. Les images doivent être prises avec attention et une bonne précision. Ceci permettra à la phase de traitement des données d'être rapide.
- **Traitement des données :** à cette phase nous avons deux opérations qui s'effectuent à la fois.

La première opération s'effectue suivant trois blocs : le premier bloc consiste à faire la détection les objets³ sur les images, le deuxième est la classification qui permet de mettre en évidence la classe correspondant à nos objets (c'est-à-dire de classer les espèces selon leurs évolutions en terme de croissance de leurs cycles de vie), et le troisième, valeur estimée, qui consiste à déterminer la qualité que chaque objet représente en terme de production.

La deuxième opération consiste à déterminer l'étendue de la superficie du champ cultivé. Ceci permet de connaître l'air de la surface du champ.

- **Analyse des données :** l'analyse des données consiste à exploiter les résultats des étapes précédentes et en faire une interprétation afin de fournir des informations sur les différentes spéculations trouvées et de donner une estimation des rendements de chaque spéculation.

2.2.2 Mise en place du modèle de détection

Pour permettre à notre système de reconnaître différent type de spéulation nous avons pris un modèle⁴ pré-entraîné⁵ et l'avons entraîné de nouveau pour qu'il puisse reconnaître nos spéculations, car il peut s'écouler des jours avant que la entraînement ne commence à zéro.. Pour ce fait nous avons procédé comme suit :

Prise de vues aériennes

Nous nous sommes rendus à Natitingou⁶ pour prendre quelques vues aériennes rapprochées de la plante de Sorgho dans quelques champs. Nous avons utilisé le drone DJI Phantom 4 Pro pour prendre ses vues.

Les images fournies par la caméra du drone ont une résolution de 5472x3648 pixels, alors nous avons utilisé la logiciel GIMP2 pour découper et roger les images afin d'avoir des images de 512x512 pixels et 800x600 pixels avec lesquelles travaillés (voir Fig. 2.4).

³objet dans notre cas représente différent type de spéulation présente dans un champ et pour chaque spéulation le nombre de fois elle est présente.

⁴Les modèles d'apprentissage en profondeur sont construits à l'aide de réseaux de neurones

⁵Modèle ou composants de modèle (par exemple, représentations vectorielles continues) qui a déjà été entraîné.

⁶Natitingou est une ville située au nord du Bénin.

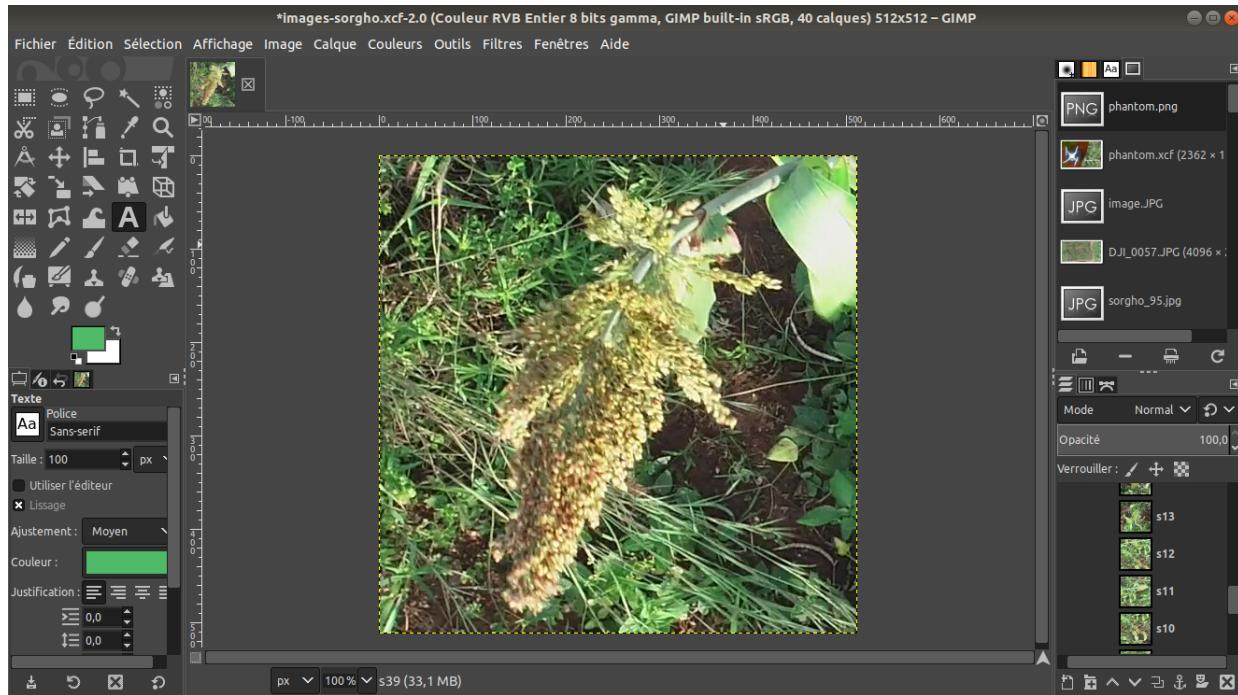


FIGURE 2.4 – Découpage des images avec GIMP2

Etiquetage avec LabelImg

Nous avons étiquetés manuellement nos images avec LabelImg. Nous avons enrégistré les annotations en tant que fichiers XML au format PASCAL VOC (voir Fig. 2.5). Notre travail étant porté sur les sorghos, l’élément le plus important, remarquable et déterminant la production du sorgho est son épis. Alors l’étiquète correspondant à un sorgho est juste la sélection des épis.

Création des TFRecords

L’API de détection d’objets TensorFlow utilise le format de fichier TFRecord . Par conséquent, nous devons enfin convertir notre ensemble de données en ce format de fichier. Pour préparer le fichier d’entrée pour l’API, nous avons créé un fichier qui contient la liste des cadres de sélection (effectué avec LabelImg) ($xmin, ymin, xmax, ymax$) pour l’image et la classe de l’objet dans le cadre de sélection. Nous avons utilisé ce script pour `xml_to_csv.py` le faire.

```

1 import os
2 import glob
3 import pandas as pd
4 import xml.etree.ElementTree as ET
5
6
7 def xml_to_csv(path):
8     xml_list = []
9     for xml_file in glob.glob(path + '/*.xml'):

```



FIGURE 2.5 – Etiquetage du Sorgho avec LabelImg

```

10     tree = ET.parse(xml_file)
11     root = tree.getroot()
12     for member in root.findall('object'):
13         value = (root.find('filename').text,
14                 int(root.find('size')[0].text),
15                 int(root.find('size')[1].text),
16                 member[0].text,
17                 int(member[4][0].text),
18                 int(member[4][1].text),
19                 int(member[4][2].text),
20                 int(member[4][3].text)
21             )
22         xml_list.append(value)
23     column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax',
24     'ymax']
25     xml_df = pd.DataFrame(xml_list, columns=column_name)
26     return xml_df
27
28 def main():
29     image_path = os.path.join(os.getcwd(), 'C:\\TensorFlow\\sorgho\\annotations')
30     xml_df = xml_to_csv(image_path)
31     xml_df.to_csv('C:\\TensorFlow\\sorgho\\data/test_labels.csv', index=None)
32     print('Successfully converted xml to csv.')
33
34

```

35 | main()

Algorithm 2.1 – Conversion XML en CSV

Ensuite nous avons utilisé le script generate_tfrecord.py pour créer les TFRecords, *train.records* et *test.records*.

```

1 """
2 Usage:
3     # From tensorflow/models/
4     # Create train data:
5     python generate_tfrecord.py --csv_input=data/train_labels.csv --output_path=
6         train.record
7     # Create test data:
8     python generate_tfrecord.py --csv_input=data/test_labels.csv --output_path=test.
9         record
10 """
11 from __future__ import division
12 from __future__ import print_function
13 from __future__ import absolute_import
14
15 import os
16 import io
17 import pandas as pd
18 import tensorflow as tf
19
20 from PIL import Image
21 from object_detection.utils import dataset_util
22 from collections import namedtuple, OrderedDict
23
24 flags = tf.app.flags
25 flags.DEFINE_string('csv_input', '', '/home/madrojudi/Sorgho training images/
26     train_labels.csv')
27 flags.DEFINE_string('output_path', '', '/home/madrojudi/Sorgho training images/
28     train.record')
29 FLAGS = flags.FLAGS
30
31
32
33
34 def split(df, group):
35     data = namedtuple('data', ['filename', 'object'])
36     gb = df.groupby(group)

```

```

37     return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(),
38                   gb.groups)]
39
40
41 def create_tf_example(group, path):
42     with tf.gfile.GFile(os.path.join(path, '{}.{}'.format(group.filename)), 'rb') as
43         fid:
44             encoded_jpg = fid.read()
45             encoded_jpg_io = io.BytesIO(encoded_jpg)
46             image = Image.open(encoded_jpg_io)
47             width, height = image.size
48
49             filename = group.filename.encode('utf8')
50             image_format = b'jpg'
51             xmins = []
52             xmaxs = []
53             ymins = []
54             ymaxs = []
55             classes_text = []
56             classes = []
57
58             for index, row in group.object.iterrows():
59                 xmins.append(row['xmin'] / width)
60                 xmaxs.append(row['xmax'] / width)
61                 ymins.append(row['ymin'] / height)
62                 ymaxs.append(row['ymax'] / height)
63                 classes_text.append(row['class'].encode('utf8'))
64                 classes.append(class_text_to_int(row['class']))
65
66             tf_example = tf.train.Example(features=tf.train.Features(feature={
67                 'image/height': dataset_util.int64_feature(height),
68                 'image/width': dataset_util.int64_feature(width),
69                 'image/filename': dataset_util.bytes_feature(filename),
70                 'image/source_id': dataset_util.bytes_feature(filename),
71                 'image/encoded': dataset_util.bytes_feature(encoded_jpg),
72                 'image/format': dataset_util.bytes_feature(image_format),
73                 'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
74                 'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
75                 'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
76                 'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
77                 'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
78                 'image/object/class/label': dataset_util.int64_list_feature(classes),
79             }))
79
80     return tf_example

```

```

80
81 def main(_):
82     writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
83     path = os.path.join(os.getcwd(), 'images')
84     examples = pd.read_csv(FLAGS.csv_input)
85     grouped = split(examples, 'filename')
86     for group in grouped:
87         tf_example = create_tf_example(group, path)
88         writer.write(tf_example.SerializeToString())
89
90     writer.close()
91     output_path = os.path.join(os.getcwd(), FLAGS.output_path)
92     print('Successfully created the TFRecords: {}'.format(output_path))
93
94
95 if __name__ == '__main__':
96     tf.app.run()

```

Algorithm 2.2 – Génération des TFRecords

Formation du modèle

Comme nous l'avons dit plus haut nous avons pris un modèle qui a été pré-entraîné, le modèle `ssd_mobilenet_v1_coco`⁶.

Pour notre part nous avons utilisé la configuration `ssd_mobilenet_v1_pets.config`⁷ où nous avons mis `num_classes` à 4 pour spécifier que nous voulons entraîner 4 classes et changé les chemins des fichiers de données de `train.records` et de `test.records` ainsi que la carte d'étiquettes. Maintenant nous pouvons lancer l'entraînement.

2.2.3 Détermination de la superficie du champ agricole capturée sur une image

Lors de la prise de vue aérienne avec le drone, nous enregistrons certaines caractéristiques de vol du drone, telles que la hauteur de vol et la taille du capteur (hauteur et largeur du capteur). Les métadonnées de l'image fournissent le reste des paramètres dont nous avons besoin, tels que la longueur focale et la taille de l'image. Nous utilisons les équations (1.10) et (1.11), et mettons les valeurs du drone pour déterminer le GSD_h et le GSD_w . Après cela, nous utilisons l'équation `eqref equa2` pour déterminer la largeur et la longueur de la zone capturée par le drone. La largeur de l'objet sur une image est la largeur de l'image et la longueur de l'objet sur

⁶http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz

⁷`ssd_mobilenet_v1_pets.config` se trouve dans le répertoire `research/object_detection/samples/configs` du modèle TensorFlow accessible sur <https://github.com/tensorflow/models>

une image est la hauteur de l'image. Ensuite, nous calculons l'aire par la formule classique de l'aire d'un rectangle.

$$\text{Aire} = \text{Longueur} * \text{Largeur} \quad (2.1)$$

2.2.4 Détermination de la valeur estimée

Après avoir entraîné notre modèle au niveau de Formation du modèle, nous utilisons ce modèle pour base à la détection sur d'autre image. Ainsi sur une image nous détectons la liste des épis de sorgho. Etant donné que la machine nous renvoie une image de forme carrée ou rectangulaire contenant l'objet détecté, nous allons filtrer cette figure afin de retrouver la forme qu'a un épi sur chaque image découpée de l'objet détecté. Connaitre la forme exacte qu'a un épi de sorgho nous permet de donner un poids à ce sorgho en vue de donner une estimation de sa valeur propre en terme de production agricole. En effet nous avons relevé quelques caractéristiques sur ses épis. Pour un épi, nous avons enrégistré la forme obtenue, ensuite nous avons défait les épis et compté le nombre de grains par épi et mesuré leurs poids. Mais avant ça, nous déterminons d'abord la taille que peut avoir la forme détectée dans le monde réel. Nous utilisons la formule suivante :

$$\text{largeur_objet_rel} = \text{GSD} * \text{largeur_objet_en_pixel_sur_l'image} \quad (2.2)$$

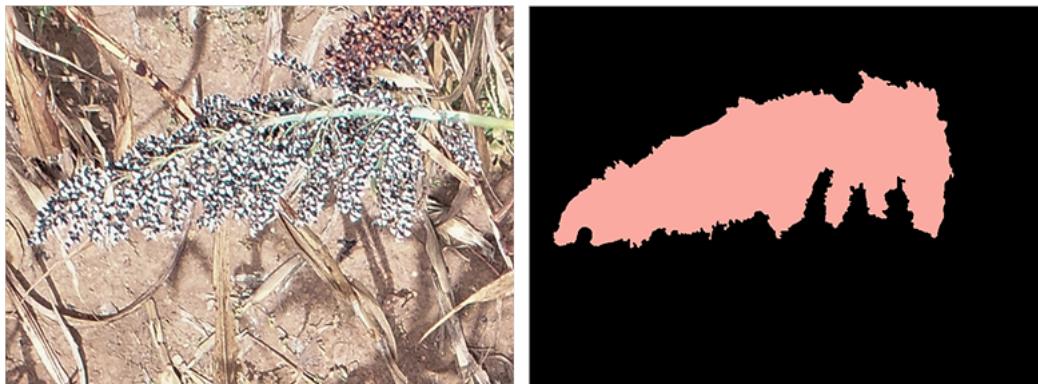


FIGURE 2.6 – Forme d'épi de sorgho

Ses données vont nous permettre de mettre en place un modèle afin d'estimer la qualité de chaque épi de sorgho détecté. Ici ce qui nous a intéressé est le *shape* et le *weight*. Nous voulons déterminer le poids que les grains de chaque épi peuvent donner et grâce aux images nous pouvons déterminer la forme de chaque épi. Alors pour transformer ce problème en un problème mathématique en IA nous dirons :

TABLEAU 2.1 – Quelques caractéristiques collectées

shape	number_grains	weight
140	1086	55.2
145	1305	56.5
143.6	1004	54.6
110	949	38.78
102.5	898	27.6
213	1254	60.6
216	1430	66.2

- *shape* représente les caractéristiques de notre problème (nommons le x_1)
- et *weight* représente l'étiquette (nommons le y).

Après analyse de ses données c'est à dire nos entrées et sorties, ce problème est un problème linéaire à résoudre. Alors nous utilisons la régression linéaire pour le résoudre. Alors nous aurons en ML l'équation :

$$y = b + w_1 x_1 \text{ où :} \quad (2.3)$$

- y est l'étiquette prédite (*weight* le poids estimé) ;
- b est le biais (l'ordonnée à l'origine) ;
- w_1 est la pondération de la caractéristique 1 ;
- x_1 est une caractéristique (*shape* connue).

2.2.5 Estimation des rendements agricoles

Après avoir déterminé la superficie du champ capturé sur une image et la valeur de chaque épis, nous pouvons déterminer le rendement agricole. Nous appliquons la formule suivante pour déterminer le rendement :

$$\text{rendement} = \frac{\sum_{i=1}^n \text{valeur_estime}}{\text{superficie_capture}} \quad (2.4)$$

Conclusion

Dans ce chapitre, nous avons justifié le choix du langage de programmation, les différents outils et bibliothèques associés à ce langage et décrit les caractéristiques de l'ordinateur utilisé pour l'implémentation ainsi que le drone utilisé. Nous avons expliqué également les méthodes et techniques mis en place. Le chapitre suivant portera sur la solution que nous proposons.

Solution proposée

Introduction

Le système d'information que nous avons conçu et développé permet d'une part de déterminer les cultures d'un champ agricole et d'autre part d'estimer le rendement agricole. Ce système nous l'avons dénommé AgriDroneFlow. Tout au long de ce chapitre nous présentons sa modélisation et parlons de leur implémentation.

3.1 Présentation du système

Notre système (dénommé AgriDroneFlow) est une application dite client lourd (communément appelée application de bureau) mais conçu avec les technologies des applications dites clients légers (communément appelées application web), à la différence que le client et le serveur sont embarqués dans la même application et tournent sur la même machine. Pour le bon fonctionnement de AgriDroneFlow, il y a deux phases à suivre :

- prise de bonne vues aériennes sur le champ agricole
- utilisation supervisée ou non supervisée de ladite application AgriDroneFlow.

L'application AgriDroneFlow en elle-même comporte deux grandes fonctionnalités pour un usage approprié à une agriculture de précision :

- détection des cultures
- estimer un rendement agricole.

3.2 Modélisation avec UML

Les diagrammes 3.1, 3.2 et 3.3 décrivent respectivement les diagrammes de cas d'utilisation et de classes et d'activités.

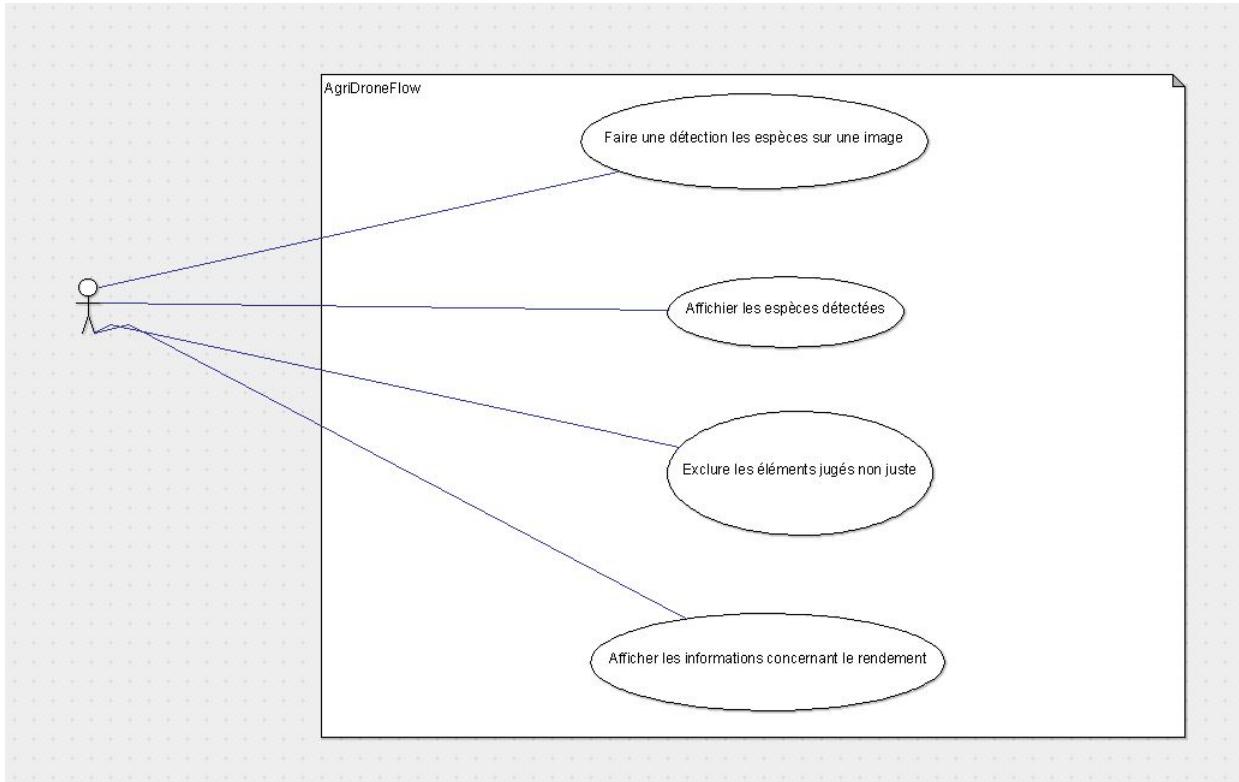


FIGURE 3.1 – Diagramme de cas d'utilisation

Pour effectuer une détection, un utilisateur suit les étapes suivantes :

- ajouter les images sur lesquelles le travail veut s'effectuer ;
- sélectionner les espèces recherchées ;
- lancer la détection ;
- après la détection, afficher le résultat en détail en vue d'exclure d'éventuelle erreur de détection ;
- afficher les informations concernant le rendement agricole.

Le système a été conçu sous une architecture RestFull. Le serveur en JavaEE avec le framework Spring Boot et le client en Angular. Ce qui offre une architecture facile d'implementation.

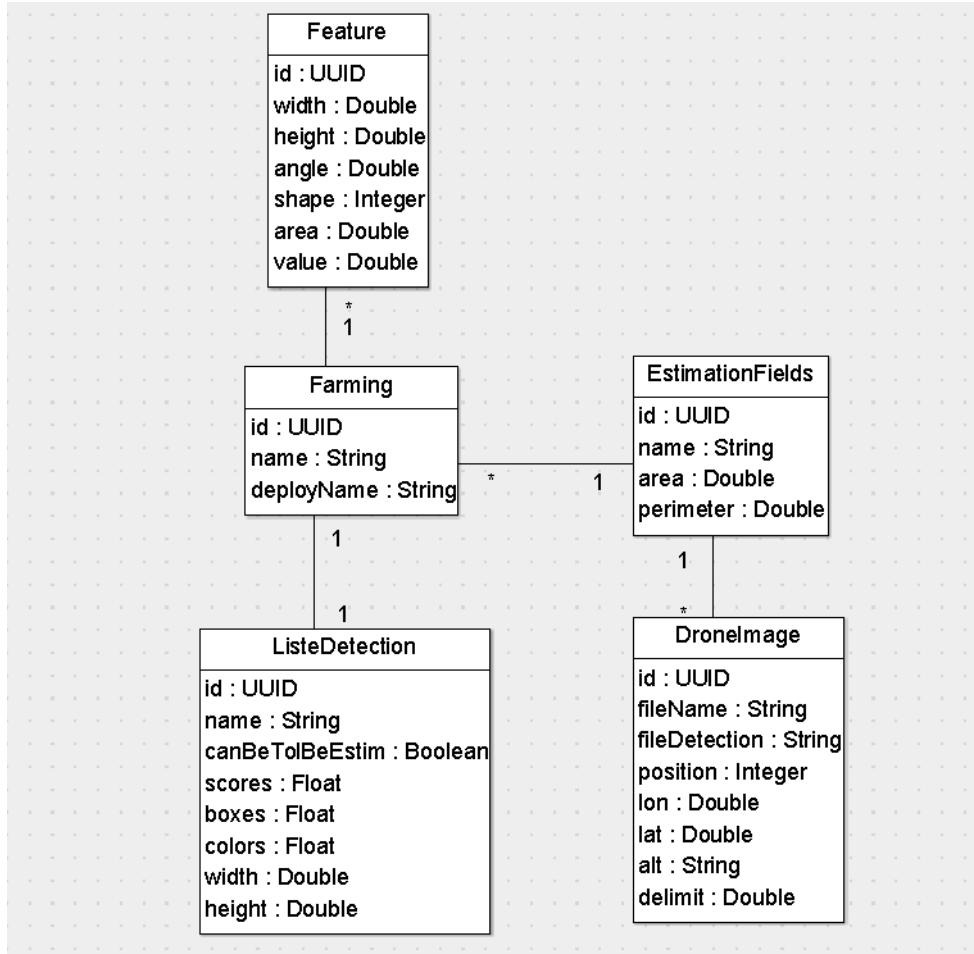


FIGURE 3.2 – Diagramme des classes

3.3 Détection des cultures

L'application permet de détecter la liste des pieds des cultures présentes (dans notre cas les pieds des Sorgho sont représentés par leurs épis) et d'estimer leurs rendements. Pour le faire, il faut sélectionner une image et lancer la détection. Après quelques secondes l'application affiche les résultats. Lors de la détection sur les images originales prises depuis le drone, les algorithmes de base de Tensorflow n'arrivaient pas à détecter directement les cultures. Pour palier à cela nous avons mis en place un algorithme pour découper les images en plusieurs autres images, après effectuer la détection sur ses images et enfin un autre algorithme qui filtre les détections afin de ne pas avoir des doublons. Le programme de détection et d'estimation est représenté sur le diagramme d'activité ci-dessous.

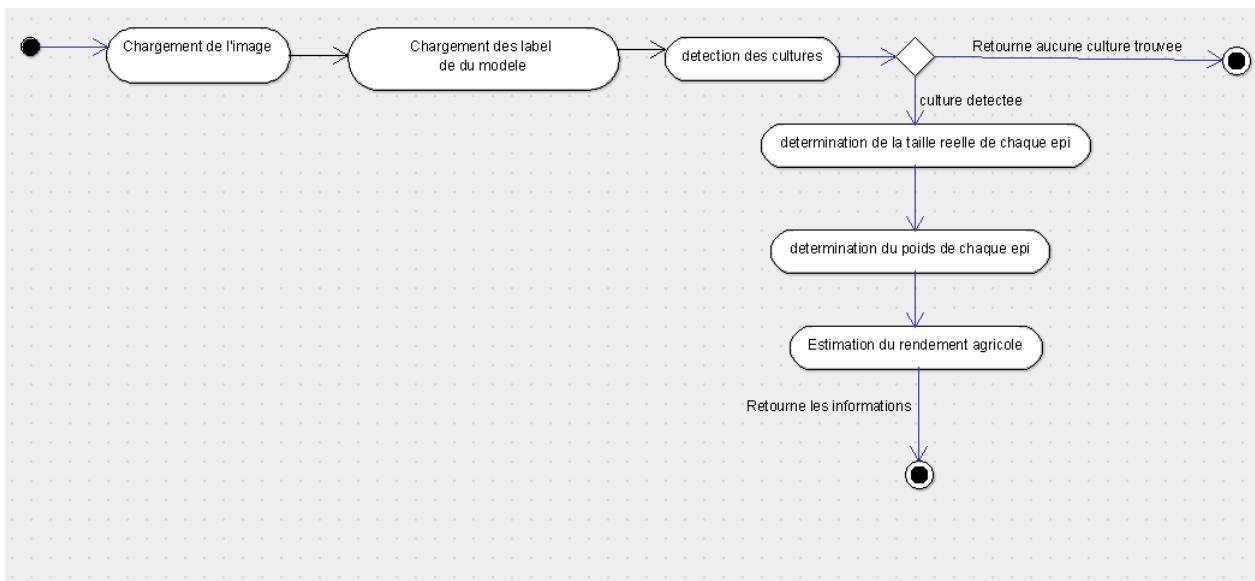


FIGURE 3.3 – Diagramme d'activités

Conclusion

Ce chapitre a traité de la solution proposée pour notre problème. Il a été question de l'estimation des rendements agricoles basée sur le Machine Learning. Le chapitre suivant portera sur les résultats et discussion.

Résultats et discussions

Introduction

Dans ce chapitre, nous fournissons les résultats obtenus dans les expériences. Certains de ces résultats seront présentés sous forme de tableaux, permettant de voir la relation entre la performance et certains des paramètres que nous mentionnons plus loin. Une discussion sur les méthodes et techniques utilisées est aussi présentée pour permettre d'évaluer le système et de cibler ce qui peut être amélioré.

4.1 Résultat

4.1.1 Tests, résultats et évaluation

Pour l'entraînement nous avons utilisé 237 formes et 33 formes pour les tests pour que le programme puisse vérifier si le modèle qu'il est entraîné à constituer est valable pour autre données que les données de l'apprentissage afin d'effectuer des ajustements. Nous pouvons observer au niveau des diagrammes 4.1 de perte que après 600 steps les pertes diminuent continuellement.

Après plus de 12000 steps nous observons de fort score de détection des épis, figures 4.2 et 4.3.

Une fois que notre modèle a suffisamment appris, nous parvenons à détecter correctement les épis de sorgho sur les images tests et images originales des drones, nous pouvons à présent appliquer notre algorithme de comptage des épis de Sorgho détectés. Les résultats obtenus pourraient être interprétés et évalués en termes de précision dans le comptage des épis de Sorgho. Pour ce fait, nous considérons les critères ci après :

- **Vrai Positif (VP)** : le nombre d'épis réels détectés par l'algorithme ;

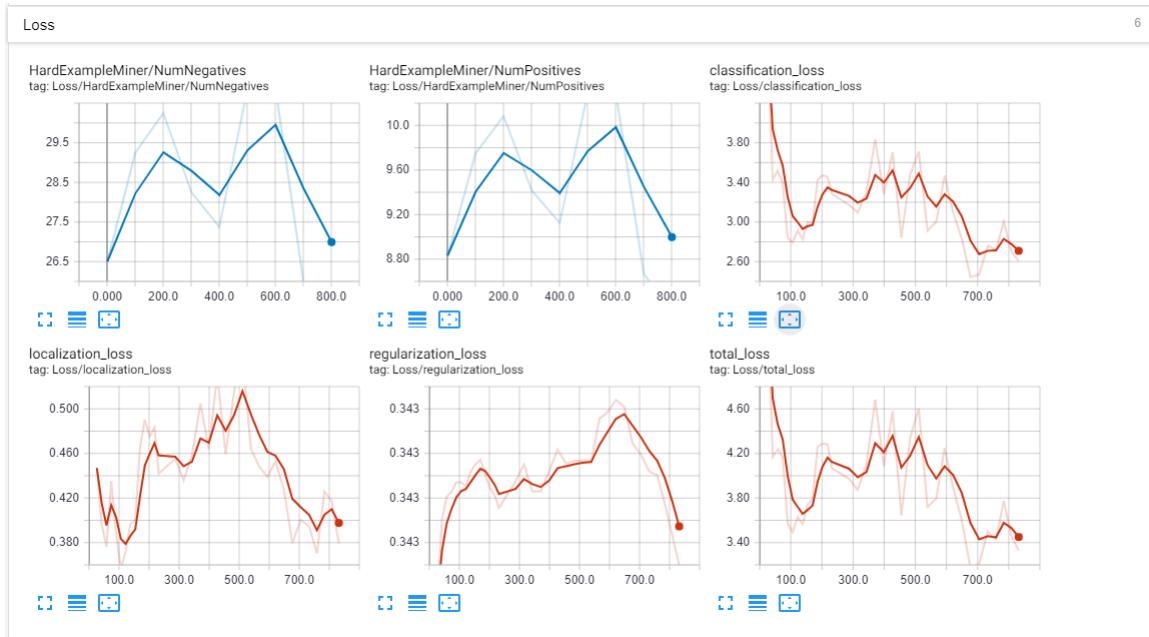


FIGURE 4.1 – Graphe des pertes lors de l'apprentissage



FIGURE 4.2 – Test de détection de Sorgho non mature



FIGURE 4.3 – Test de détection de Sorgho mature

- **Vrai Négatif (VN)** : le nombre de fois où l’algorithme ne détectait pas d’épis et est exacte ;
- **Faux Positif (FP)** : le nombre de fois où le système a détecté un épi mais il n’y en avait pas ;
- **Faux Négatif (FN)** : le nombre de épis qui n’ont pas été détectés par le système.

Ces métriques énumérés plus haut permettent de calculer la précision, le rappel et la justesse de notre algorithme en appliquant les formules ci-après :

$$Pecision = V/(VP + FP) \quad (4.1)$$

$$Rappel = VP/(VP + FN) \quad (4.2)$$

$$Justesse = (VP + VN) * 100/(VP + VN + FP + FN) \quad (4.3)$$

Le tableau ci-dessous montre les différents résultats obtenus pour une série de cinq (05) images de drone pour nos applications.

Avec ces résultats, nous pouvons conclure que notre algorithme de détection à une justesse moyenne de 74,5% sur nos données.

En ce qui concerne le poids, nous évaluons la précision par :

$$Precision_du_poids = 100 - \frac{|poids_reel - poids_estim|}{poids_reel} \times 100 \quad (4.4)$$

Avec ces résultats, nous pouvons conclure que notre algorithme d’estimation a une précision moyenne de 99%.

TABLEAU 4.1 – Evaluation de l'algorithme de détection des épis de sorgho

Image	<i>Nbre réel d'épi</i>	<i>Nbre détecté</i>	<i>VP</i>	<i>FP</i>	<i>FN</i>	<i>VN</i>
1	34	28	24	4	10	0
2	20	19	17	2	3	0
3	20	19	18	1	4	0
4	19	15	15	0	4	0
5	20	15	15	0	5	0

Image	<i>Précision</i>	<i>Rappel</i>	<i>Justesse</i>
1	0.83	0.70	63.16%
2	0.89	0.85	77.27%
3	0.95	0.82	78.26%
4	1	0.79	78.95%
5	1	0.75	75%

TABLEAU 4.2 – Evaluation de l'algorithme d'estimation du poids (gramme) des épis de sorgho

Image épis	<i>Poids réel</i>	<i>Poids estimé</i>	Précision du poids
1	53.8	53.3	99%
2	55.5	55.0	99%
3	56.9	56.4	99%
4	28.5	28.2	99%
5	27.9	27.6	99%

4.2 Quelques interfaces de l'application

Pour les tests de bon fonctionnement, nous présentons ici les captures d'écran essentielles. Les tests sont effectués sur notre ordinateur personnel dont les caractéristiques sont mentionnées au chapitre 2.

Au niveau de la figure 4.4 est présentée l'interface d'accueil de AgriDroneFlow. Elle montre à l'utilisateur les grands menus de l'application.

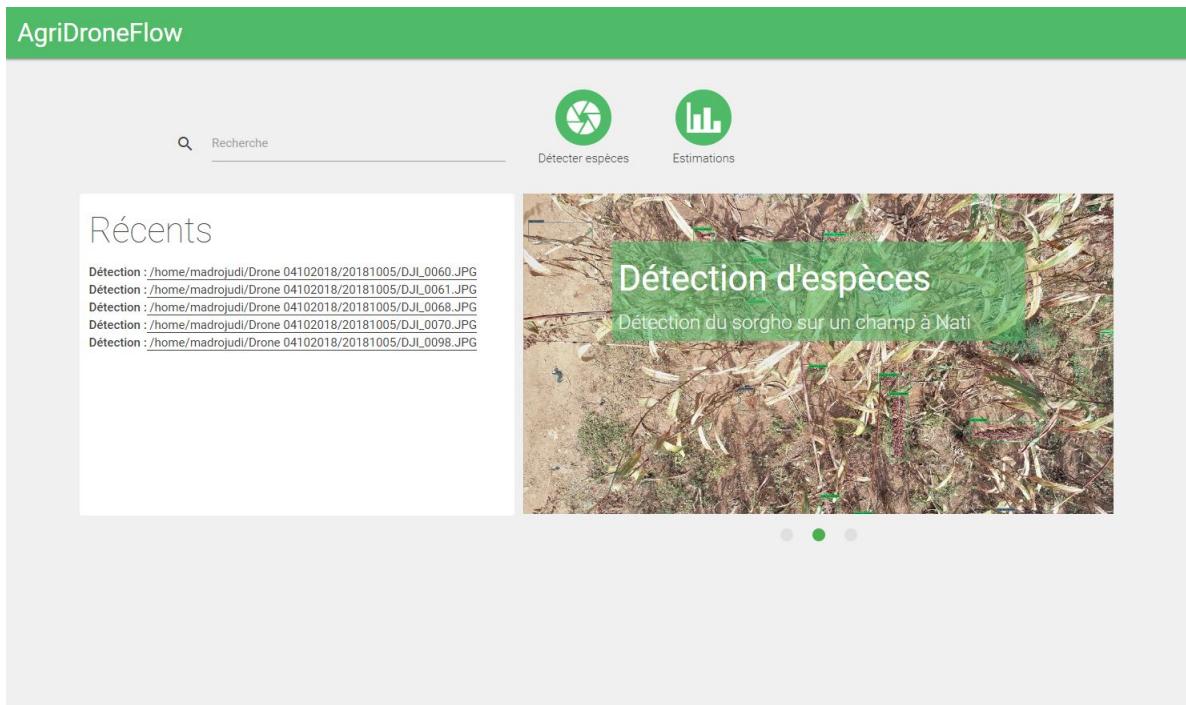


FIGURE 4.4 – Interface d'accueil d'AgriDroneFlow

Au niveau de la figure 4.5, une détection de sorgho non mure a été effectuée et les carateristiques de la zone capturée ont été données.

La figure 4.6 montre une détection de sorgho dans un champ prêt à être récolté. Les caractéristiques sur la zone capturée ont été données ici également et avec d'autres détails.

Pour finir la figure 4.7 présente la liste des épis de sorgho détectés sur une zone capturée de champ de sorgho.

4.3 Discussions

4.3.1 Points positifs

Dans l'estimation des rendements agricoles basée sur le Machine Learning dans un champs de Sorgho, à notre connaissance notre étude a été la première à être menée. De ce fait il n'y

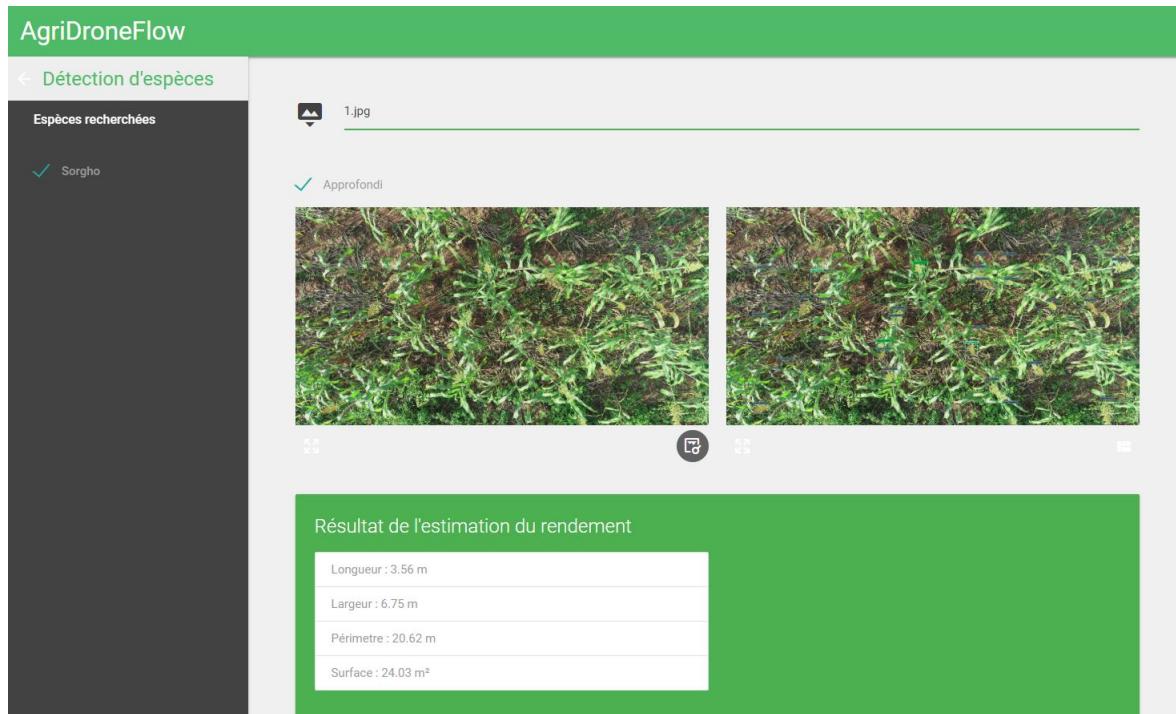


FIGURE 4.5 – Détection du sorgho non mure

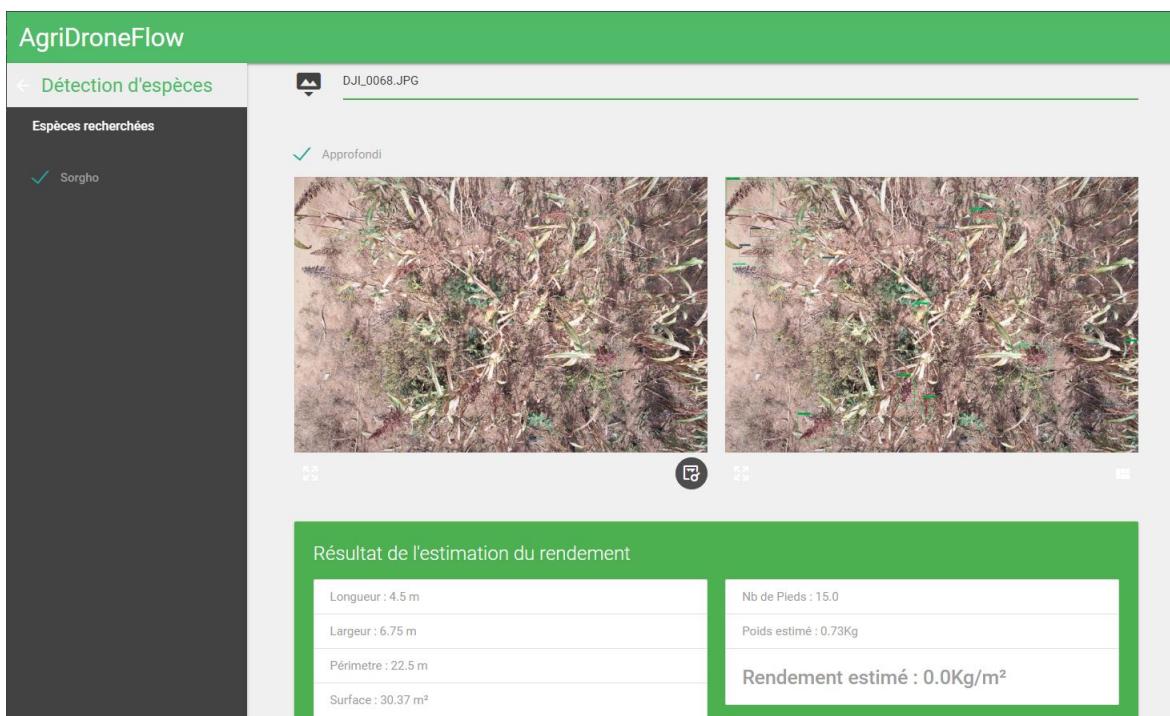


FIGURE 4.6 – Détection du sorgho mature et estimation du rendement agricole

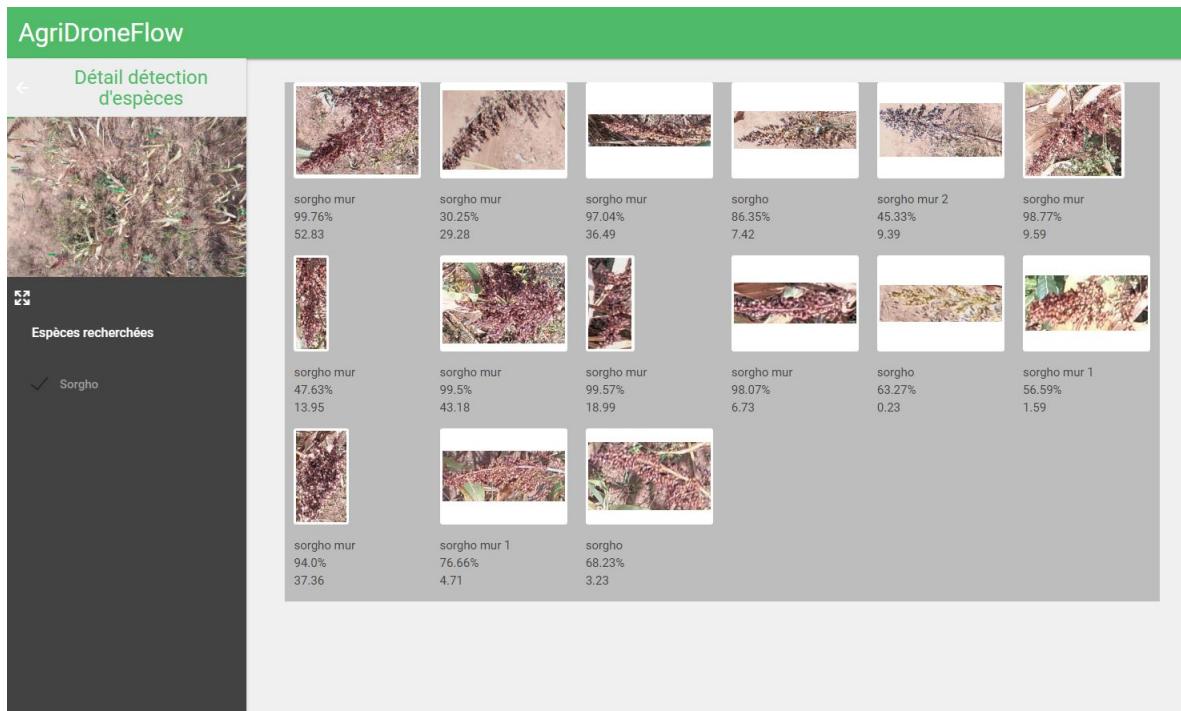


FIGURE 4.7 – Liste des épis détectées et leurs poids estimés

avait pas de données existantes à exploiter. Alors nous avons été sur le terrain afin de collecter les données et mis en place un modèle de reconnaissance des épis de sorgho et à estimer leur poids en vue d'estimer le rendement agricole. Pour une première étude une justesse moyenne supérieure à 70% est un grand point positif pour poursuivre les recherches afin d'améliorer nos algorithmes et notre modèle.

4.3.2 Points négatifs

Nous avons eu que 270 formes d'épis de Sorgho pour effectuer le travail. Ce qui est vraiment petit pour obtenir une grande performance. Le taux d'erreur de détection n'étant pas négligeable nous avons du donner la mains à l'utilisateur de faire un travail supervisé depuis l'interface du système afin d'exclure les faux positifs.

4.3.3 Améliorations

Comme amélioration, nous pouvons citer la collecte de beaucoup plus de données, c'est-à-dire des vues aériennes sur lesquelles extraire plus de formes d'épis et être doté d'un machine avec un GPU (Graphcal Processing Unit) de mémoire d'au moins 11 Giga octets sera un atout non négligeable dans la mise en place du modèle. Il faut aussi bien régler le drone et effectuer les prises de vues à un bon moment de la journée (selon l'éclairage).

Conclusion

Les données sont les clés pour avoir un système performant et efficace, comme il est toujours le cas en machine learning, ce n'est pas celui qui a le meilleur algorithme qui gagne mais c'est celui qui a le plus de données. Néanmoins notre modèle de détection et d'évaluation des épis de sorgho a actuellement une justesse moyenne de 74,5% et une précision moyenne des poids de 99%.

Conclusion et perspectives

L'agriculture est le levier de transformation de l'Afrique. C'est dans cette perspective que notre travail veut améliorer les techniques utilisées dans le secteur agricole. Ce travail a porté sur l'estimation des rendements agricoles avec le Machine Learning. Le système mise en place est dénommé AgriDroneFlow. AgriDroneFlow est un ensemble de solutions matérielles et logicielles qui permet d'estimer le rendement d'un champs agricole. Il permet entre autres de détecter le nombre de pieds de sorgho présente sur une image et d'estimer leurs poids en vue d'estimer le rendement total. Il est totalement basé sur la détection des objets avec du Deep Learning. Notre modèle de détection des épis de Sorgho et d'estimation de leurs poids a actuellement une justesse moyenne de 74,5% et une précision moyenne des poids de 99%. Nous pensons que la première version de ce système peut être déjà utilisé est aidé beaucoup d'agriculteurs.

Nous aimeraisons poursuivre les recherches afin d'améliorer AgriDroneFlow pour mettre en place un système à plus de 99% de précision et fiable, et nous aimeraisons que l'amélioration fasse un thème de thèse. Nous pensons utiliser plusieurs capteurs dans le domaine de l'agriculture de précision afin de le rendre incontournable en terme d'Agriculture de Précision. En outre d'autre perspectives intéressantes sont envisagées pour rendre le système plus complet. Par exemple : étendre le modèle pour plusieurs cultures agricoles, intégrer l'analyseur des capteurs NDVI et infrarouge et intégrer le contrôle automatisé des drones.

Bibliographie

- [1] PADEN , *Guide pour l'évaluation du rendement et l'estimation de la production horticole*, Unité de Coordination du Programme (UCP).
- [2] Robert D Grisso, Marcus M. Alley, W.G. Wysor, D. Holshouser et Wade Thomason *Precision Farming Tools : Soil Electrical Conductivity*. Virginia Cooperative Extension ; Publication 442-508, Janvier 2009.
- [3] Dr. Gopi Kandaswamy , *Machine Learning for Agriculture*, Tata Consultancy Services.
- [4] Patrick Peccatte , *Métadonnées : une initiation*, Soft Experience, Décembre 2007.
- [5] AGRINOVA , *Guide pour la production et l'utilisation des cartes de rendement*, AGRINOVA.
- [6] D Boukhlouf, Chapitre Généralités sur le traitement d'images Introduction, Université Bis-kra, 2015.
- [7] Sabri akram & Benali Moustafa , *Développement d'une application de traitement d'images*, Université Abou Bakr Belkaïd-Tlemcen, Juin 2017.
- [8] Chloé-Agathe Azencott, *Utilisez des modèles supervisés non linéaires*, Open Classroom, Aout 2018.
- [9] Pascal Monasse & Kimia Nadjahi, *Classez et segmentez des données visuelles*, Open Classroom, Février 2019.
- [10] Google, *Cours d'initiation au Machine Learning avec les API TensorFlow*, Google Developers, Septembre 2018.

Webographie

- [11] Wikipedia, Agriculture de Précision, https://fr.wikipedia.org/wiki/Agriculture_de_pr%C3%A9cision, Février 2019.
- [12] Wikipedia, Apprentissage automatique, https://fr.wikipedia.org/wiki/Apprentissage_automatique, Mars 2019.
- [13] Rajesh Khadka, Introduction to Machine Learning #1; <https://towardsdatascience.com/machine-learning-65dbd95f1603>, Septembre 2017.
- [14] Datafranca, Apprentissage partiellement supervisé, https://datafranca.org/wiki/Apprentissage_partiellement_supervis%C3%A9, Janvier 2019.
- [15] DAP, Apprentissage par renforcement, <https://dataanalyticspost.com/Lexique/apprentissage-par-renforcement/>, 2018.
- [16] Wikipedia, Apprentissage profond, https://fr.wikipedia.org/wiki/Apprentissage_profound, Mars 2019.
- [17] Propeller Aero, <https://www.propelleraero.com/blog/ground-sample-distance-gsd-calculate-drone-data/>, Février 2018.
- [18] Developpez, <https://python.developpez.com/cours/TutoSwinnen/?page=Introduction>, 2013.
- [19] Wikipedia, Sorghum, <https://en.wikipedia.org/wiki/Sorghum>, Février 2019.

Annexe A

Annexe

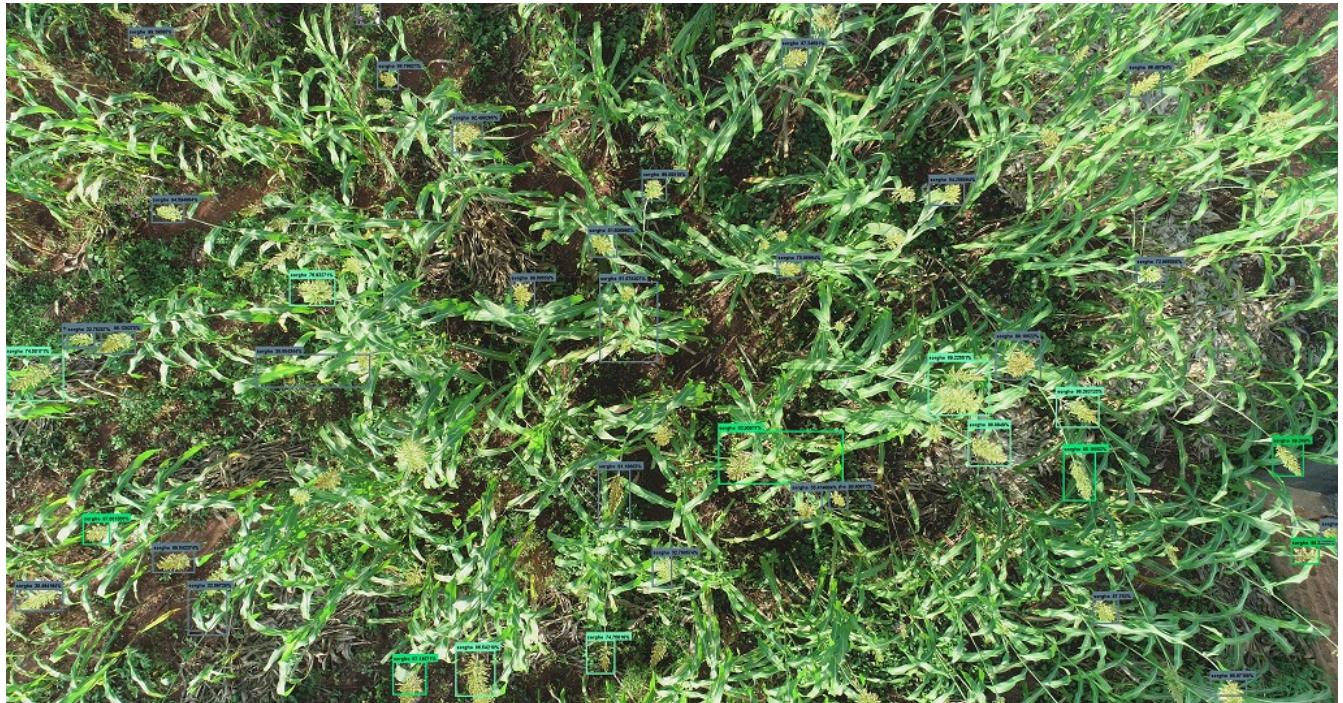


FIGURE A.1 – Détection du sorgho non mature sur une image originale du drone.



FIGURE A.2 – Détection du sorgho mure sur une image originale du drone.

```

1  @Override
2  public List<ListDetection> findObjects(String savedModel, String labelMap, String
3      images) throws Exception {
4      final String [] labels = loadLabels(labelMap);
5      List<ListDetection> listDetections = new ArrayList<>();
6      System.out.println(Arrays.toString(labels));
7      try (SavedModelBundle model = SavedModelBundle.load(savedModel, "serve")) {
8          printSignature(model);
9
10         final String filename = images;
11         List<Tensor<?>> outputs = null;
12         try (Tensor<UInt8> input = makeImageTensor(filename)) {
13             outputs = model.session().runner().feed("image_tensor", input).fetch("detection_scores")
14                 .fetch("detection_classes").fetch("detection_boxes").run();
15         }
16         try (Tensor<Float> scoresT = outputs.get(0).expect(Float.class);
17             Tensor<Float> classesT = outputs.get(1).expect(Float.class);
18             Tensor<Float> boxesT = outputs.get(2).expect(Float.class)) {
19             // All these tensors have:
20             // - 1 as the first dimension
21             // - maxObjects as the second dimension
22             // While boxesT will have 4 as the third dimension (2 sets of (x, y)
// coordinates).

```

```

23 // This can be verified by looking at scoresT.shape() etc.
24 int maxObjects = (int) scoresT.shape()[1];
25 float[] scores = scoresT.copyTo(new float[1][maxObjects])[0];
26 float[] classes = classesT.copyTo(new float[1][maxObjects])[0];
27 float[][] boxes = boxesT.copyTo(new float[1][maxObjects][4])[0];
28 // Print all objects whose score is at least 0.5.
29 System.out.printf("* %s\n", filename);
30 // boolean foundSomething = false;
31 for (int i = 0; i < scores.length; ++i) {
32     if (scores[i] > 0.3) {
33         ListDetection listDetection = new ListDetection();
34         listDetection.setObjectsFound(true);
35         listDetection.setBoxes(boxes[i]);
36         listDetection.setClasses(classes[i]);
37         listDetection.setScores(scores[i]);
38         listDetection.setName(labels[(int) classes[i]]);
39         listDetection.setColor(Color.CYAN.getRGB());
40         listDetections.add(listDetection);
41     }
42     // foundSomething = true;
43     // System.out.printf("\tFound %-20s (score: %.4f)\n", labels[(int)
44     classes[i]],
45     // scores[i]);
46     // System.out.println(classes[i]);
47 }
48 // if (!foundSomething) {
49 // System.out.println("No objects detected with a high enough score.");
50 // }
51 }
52 }
53 //return listDetections;
54 affectEstimateValue(images, listDetections, 127);
55 return listDetections;
56 }
```

Algorithm A.1 – Méthode de détection des cultures (Code Java)

```

1 private void affectEstimateValue(String filename, List<ListDetection>
2 listDetections, double thresh){
3
4     try {
5         BufferedImage img = ImageIO.read(new File(filename));
6
7         for (int i = 0; i < listDetections.size(); i++) {
8             float[] boxes = listDetections.get(i).getBoxes();
```

```

8     float xStart = img.getWidth() * boxes[1];
9     float yStart = img.getHeight() * boxes[0];
10    float xEnd = img.getWidth() * boxes[3];
11    float yEnd = img.getHeight() * boxes[2];
12    BufferedImage thumnable = img.getSubimage((int)xStart, (int)yStart, (int)(xEnd-xStart), (int)(yEnd-yStart));
13
14    String filenametmp = "shape_" + UUID.randomUUID().toString() + ".jpg";
15    storageService.store(thumnable, filenametmp);
16    Path path = storageService.loadImage(filenametmp);
17
18
19    //org.opencv.imgproc.Imgproc.
20    //System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
21    Mat mat = Imgcodecs.imread(path.toString());
22    Mat gray = new Mat();
23    Imgproc.cvtColor(mat, gray, Imgproc.COLOR_BGR2GRAY);
24    Size ksize = new Size(5, 5);
25    Mat blur = new Mat();
26    Imgproc.GaussianBlur(gray, blur, ksize, 0);
27    Mat binary = new Mat();
28    Imgproc.threshold(blur, binary, thresh, 255, Imgproc.THRESH_BINARY_INV);
29    List<MatOfPoint> contours = new ArrayList<>();
30    Mat hierarchy = new Mat();
31    Imgproc.findContours(binary, contours, hierarchy, Imgproc.RETR_EXTERNAL,
32    Imgproc.CHAIN_APPROX_SIMPLE);
33    double area = 0;
34    MatOfPoint points = new MatOfPoint();
35    for (MatOfPoint point : contours) {
36        Mat clone = point.clone();
37        double tmp = Imgproc.contourArea(clone);
38        if (area < tmp) {
39            area = tmp;
40            points = point;
41        }
42    }
43
44    listDetections.get(i).setQuantity(area);
45
46    Imgproc.fillConvexPoly(mat, points, new Scalar(0, 255, 0));//(mat, contours
47    , -1, new Scalar(0, 255, 0), 5); //drawContours(mat, contours, 10, Scalar.all
48    (102));
49    Imgcodecs.imwrite(filenametmp + "_g.jpg", mat);
50    //storageService.deleteImage(filenametmp);

```

```

49     }
50
51
52 } catch (IOException e) {
53     // TODO Auto-generated catch block
54     e.printStackTrace();
55 }
56
57 //return listDetections;
58 }
```

Algorithm A.2 – Méthode d'estimation des poids des épis (Code Java)

```

1 private void affectEstimateValue(String filename, List<ListDetection>
2 listDetections, double thresh){
3
4     try {
5         BufferedImage img = ImageIO.read(new File(filename));
6
7         for (int i = 0; i < listDetections.size(); i++) {
8             float[] boxes = listDetections.get(i).getBoxes();
9             float xStart = img.getWidth() * boxes[1];
10            float yStart = img.getHeight() * boxes[0];
11            float xEnd = img.getWidth() * boxes[3];
12            float yEnd = img.getHeight() * boxes[2];
13            BufferedImage thumnable = img.getSubimage((int)xStart, (int)yStart, (int)
14 xEnd-xStart), (int)(yEnd-yStart));
15
16            String filenametmp = "shape_" + UUID.randomUUID().toString() + ".jpg";
17            storageService.store(thumnable, filenametmp);
18            Path path = storageService.loadImage(filenametmp);
19
20
21            //org.opencv.imgproc.Imgproc.
22            //System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
23            Mat mat = Imgcodecs.imread(path.toString());
24            Mat gray = new Mat();
25            Imgproc.cvtColor(mat, gray, Imgproc.COLOR_BGR2GRAY);
26            Size ksize = new Size(5, 5);
27            Mat blur = new Mat();
28            Imgproc.GaussianBlur(gray, blur, ksize, 0);
29            Mat binary = new Mat();
30            Imgproc.threshold(blur, binary, thresh, 255, Imgproc.THRESH_BINARY_INV);
31            List<MatOfPoint> contours = new ArrayList<>();
32            Mat hierarchy = new Mat();
33            Imgproc.findContours(binary, contours, hierarchy, Imgproc.RETR_EXTERNAL,
```

```

    Imgproc.CHAIN_APPROX_SIMPLE);
32     double area = 0;
33     MatOfPoint points = new MatOfPoint();
34     for (MatOfPoint point : contours) {
35         Mat clone = point.clone();
36         double tmp = Imgproc.contourArea(clone);
37         if (area<tmp) {
38             area = tmp;
39             points = point;
40         }
41     }
42
43     listDetections.get(i).setQuantity(area);
44
45     Imgproc.fillConvexPoly(mat, points, new Scalar(0, 255, 0));//(mat, contours
46     , -1, new Scalar(0, 255, 0), 5); //drawContours(mat, contours, 10, Scalar.all
47     (102));
48     Imgcodecs.imwrite(filenameTmp + "_g.jpg", mat);
49     //storageService.deleteImage(filenameTmp);
50
51 }
52 } catch (IOException e) {
53     // TODO Auto-generated catch block
54     e.printStackTrace();
55 }
56
57 //return listDetections;
58 }
```

Algorithm A.3 – Méthode de filtrage des doublon (Code Java)

