Ecole Centrale Casablanca

Groupe N° :2

P2023 ECC

PROJET LEARNING BY DOING

Détection des maladies des plantes par le traitement d'images

Réalisé par :

- Abbadi Khaoula
- Ait Hajjoub Mohamed
- Bagagnan Abdoul Ganihou
- Dribine Hamza
- El Alem Mohamed Cheik

Encadré par :

Prof. Abdelouahed El Fatimy

Casablanca

Année Universitaire 2020/2021

Résumé

Le niveau développé des technologies de nos jours introduit des bouleversements dans le monde agricole, qui tend à une digitalisation de ce secteur à travers l'utilisation de nombreux gadgets, à savoir les capteurs, les réseaux intelligents, les applications, etc. Tout au long de cette année scolaire le projet Learning by Doing avait pour objectifs la détection et la résolution de problématique majeur en ce qui concerne l'agriculture tout en utilisant les nouvelles technologies. Notre équipe, après de recherches poussées sur le domaine, a décidé de s'intéresser à la thématique de détection de maladie des plantes tout en se fixant comme objectif « trouver une solution de détection de maladie des plantes en utilisant le traitement d'images ». En effet, bien vrai que de nombreuse solution existe dans différents pays à ce sujet, cependant ces solutions manque de précision et ne traite également que des plantes propres à chaque pays. La pomme et la pomme de terre étant des plantes ayant une place de choix dans l'économie marocaine et faisant parfois l'objet d'attaque de plusieurs pathologies, trouver une méthode de détection précise et rapide de ces maladies nous parait donc crucial. Les résultats envisagés étaient un algorithme de traitement d'image photographié et un traitement d'image hyperspectral. Cependant dans l'optique de réduire le coup et de rendre cela accessible pour les agriculteurs à moyen revenu nous nous sommes orientés vers l'algorithme de traitement d'images. Dans l'optique de traiter des problématiques pas encore résolut nous nous sommes uniquement intéressés aux maladies de la pomme de terre et des pommes. Ayant utilisé une base de données du site Kaggle, nous avions pu vérifier la validité et la précision de notre algorithme qui est de 0 ,98. Nous nous sommes orientés ensuite vers la mise en place d'un site-web en lieux d'une application. Nous avions été confrontés à la difficulté de la qualité des bases de données, problèmes que nous avions résolue en combinant les images de meilleure qualité de plusieurs bases de données.

Remerciements

Nos remerciements vont tout d'abord à l'endroit de la direction des programmes de l'Ecole Centrale Casablanca qui a su à travers le projet Learning by Doing nous mettre au-devant de la scène de notre formation. Par le biais de ce projet nous avions pu développer plusieurs compétences, à savoir le développement, l'innovation, la recherche et un très grand esprit d'équipe.

Nous tenons à remercier également notre tuteur, Monsieur Abdelouahed El Fatimy, qui a su nous inspirer et nous guider dans nos recherches à nous intéresser à une problématique pointue en ce qui concerne l'agriculture. A travers ses différentes remarques, il a su nous amener à effectuer des recherches plus développées sur notre problématique dans l'objectif de mieux l'affiner. Également, nous tenons à remercier monsieur Khalid Dahi qui a été là pour nous apporter des éléments de réponses à un certain nombre de nos préoccupations. Enfin nous tenons à remercier la corp professorale de l'école Centrale Casablanca qui, à travers leur remarque et propositions à nos divers soutenances, ont permis de mieux façonner notre solution.

A toute ces personnes nous souhaitons exprimer notre gratitude pour nous avoir accompagné dans la réalisation de notre solution.

Table des matières

Liste des figures

Liste des tableaux

1. Introduction

Augmenter la productivité agricole tout en conservant les ressources naturelles et la qualité de vie de l'agriculteur est le défi majeur de l'agriculture durable. Or, les maladies des plantes affectent la productivité des cultures et représentent une préoccupation constante pour les agriculteurs. Comment optimiser la gestion des cultures dans le contexte actuel, qui vise à limiter les apports d'engrais et de pesticides ?

Mieux détecter les maladies des plantes, et les détecter plus tôt, est un enjeu de recherche majeur pour améliorer l'efficacité des traitements des plantes. Dans le contexte d'une numérisation de l'agriculture ; il convient d'orienter cette recherche vers l'utilisation de nouvelle technologie afin de trouver une solution optimale qui puisse permettre de détecter avec une très grande précision et très rapidement les maladies des plantes. Dans la suite, ayant pour optique de décrire les étapes de mises en place de solution, nous allons d'abord présenter le contexte ainsi que la problématique, ensuite nous présenterons l'Etat de l'art ainsi que la méthodologie et la planification, enfin nous présenterons les résultats et les commentaires.

2. Développement du projet :

2.1 Planification

Dans l'optique de mieux définir les taches et de mieux nous organiser pour la réalisation du projet nous nous sommes reparti les taches tout au long de l'année. Également pour la rédaction du rapport final nous avions diviser les différentes sections entre les membres de l'équipe.

2.1.1 Gestion du projet

Membre de l'équipe	Rôles et responsabilités	Sections rédigées
Abdoul Ganihou Bagagnan	IA designer, chef d'equipe	Etat de l'art, Remerciement, contexte et problématique
Khaoula Abbadi	IA designer, Responsable qualité	Mise en place de la solution, analyse et commentaire
Hamza Dribine	IA designer, Responsable communication	Réalisation et test, Introduction, résumé
Mohamed Ait Hajjoub	Web designer, Responsable logistique	Mise en place de la solution et réalisation, Conclusion
Mohamed Cheick El Alem	Web designer, Responsable logistique	Réalisation, diagramme gant

Tableau 1 : Récapitulatif des rôles et des parties rédigées

2.1.2 Diagramme de Gant

	Nom	Durée	Début	Fin	Prédécesseurs	Noms des ressources
1	Lancement des projets: Prése	0 jours	14/10/20 08:00	14/10/20 08:00		
2	Etude Bibliographique	5 jours?	14/10/20 08:00	20/10/20 17:00		Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem
3	**⊟ace de la charte projet**	**24 jours?**	**18/11/20 08:00**	**21/12/20 17:00**		**Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem**
4	identification du besoin	24 jours?	18/11/20 08:00	21/12/20 17:00	2	
5	identifications des objectifs	24 jours?	18/11/20 08:00	21/12/20 17:00	2	
6	**⊟Recherche des solutions**	**30,875 jou...**	**22/12/20 09:00**	**02/02/21 17:00**		**Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem**
7	Recherche bibliographique	30,875 jours?	22/12/20 09:00	02/02/21 17:00		
8	premier choix	30,875 jours?	22/12/20 09:00	02/02/21 17:00		
9	**⊟Mise en place du cahier d**	**42,875 jou...**	**03/02/21 09:00**	**02/04/21 17:00**		**Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem**
10	analyse fonctionnelle	42,875 jours?	03/02/21 09:00	02/04/21 17:00		
11	choix et justification de la s	42,875 jours?	03/02/21 09:00	02/04/21 17:00		
12	**⊟Prototypage, test et vali**	**91 jours?**	**05/04/21 08:00**	**09/08/21 17:00**		**Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem**
13	Developpement du model a	17,333 jours?	05/04/21 08:00	28/04/21 10:40		Khaoula Abbadi;Bagagnan Abdoul Ganihou;Hamza Dirbine
14	Developpement du model a	12 jours?	10/06/21 08:00	25/06/21 17:00		Hamza Dirbine
15	**⊟Mise en place du site w**	**12 jours?**	**15/06/21 08:00**	**30/06/21 17:00**		**Mohamed Ait Ajjoub;Mohamed Cheik El Alem**
16	developpement de la part	12 jours?	15/06/21 08:00	30/06/21 17:00		
17	developpement de la part	12 jours?	15/06/21 08:00	30/06/21 17:00		
18	Etablissement de la liaison e	0,5 jours?	01/07/21 08:00	01/07/21 13:00		Mohamed Ait Ajjoub;Hamza Dirbine
19	Realisation des test	13 jours?	15/06/21 08:00	01/07/21 17:00		Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem
20	Finalisation du prototype	28 jours?	01/07/21 08:00	09/08/21 17:00		Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem
21	Visite sur le terrain pour val	1 jour	03/07/21 08:00	05/07/21 17:00		Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem
22	**⊟Soutenances finales**	**1 jour?**	**09/07/21 08:00**	**09/07/21 17:00**		**Khaoula Abbadi;Mohamed Ait Ajjoub;Bagagnan Abdoul Ganihou;Hamza Dirbine;Mohamed Cheik El Alem**
23	présentation de la solution	1 jour?	09/07/21 08:00	09/07/21 17:00		

Tableau 2 : Illustration des différentes étapes tout au long du PLBD

.

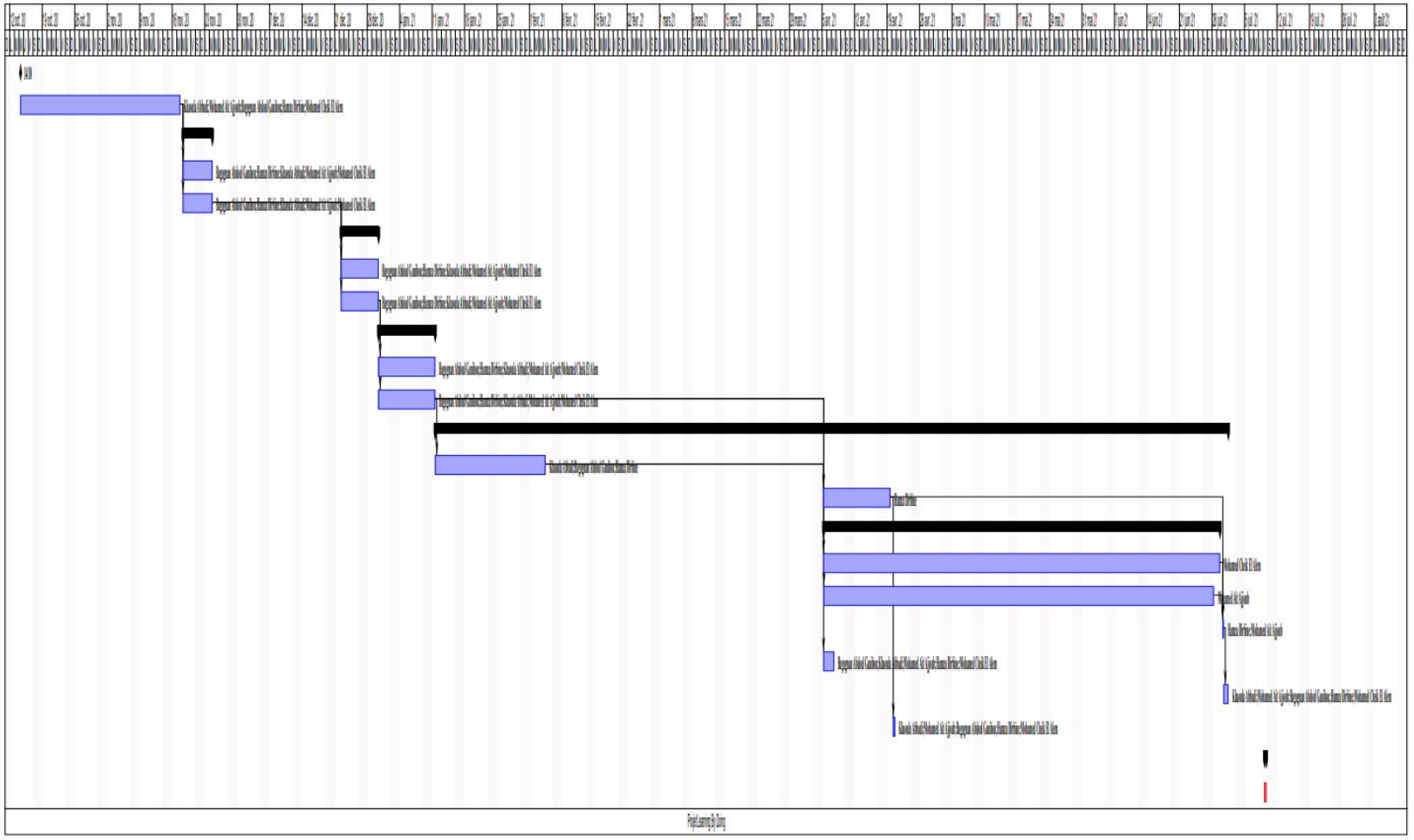

Tableau 3 : Diagramme Gant

2.2 Contexte et problématique

L'agriculture africaine en générale fait face à de nombreux problèmes qui gênent son développement. Le projet Learning By Doing a pour vocation de remédier à certains de ces problèmes en proposant des solutions technologiques innovantes à des problématiques majeurs de ce secteur. Le secteur agricole au Maroc, en particulier, est le secteur principal d'activité économique : le revenu d'environ 40% de la population en dépend. Or, le secteur agricole reste assez vulnérable face aux maladies des plantes, qui induisent des pertes non négligeables dans les cultures qui contribue activement à la sécurité alimentaire du pays. A titre illustratif le mildiou (qui est une maladie cryptogamique qui touche les pommes de terre) peut engendrer, selon les conditions météorologiques et le taux d'infection, jusqu'à 80% de pertes. En outre, les cultures de blé peuvent être attaquées par plusieurs agents pathogènes à différents stades de leurs développements. Parmi les maladies qui affectent le blé on peut citer : les maladies foliaires qui peuvent engendrer des pertes pouvant atteindre 70% pour le cas de la rouille jaune, les pourritures racinaires qui peuvent quant à elle induire des pertes de rendement pouvant atteindre 60% selon la période de l'infection.

La difficulté à détecter certaines maladies des cultures résident dans le fait que certaines maladies ne sont pas saisonnières, rendant donc les agriculteurs incapables de prédire une infection et donc d'avoir un plan d'action efficace qui pourrait minimiser les dégâts. Également les symptômes de certaines maladies ne sont pas très visibles à l'œil nu ou se confonde avec d'autre maladie, rendant donc très difficile l'identification. Nous nous sommes donc orientés après constatation de ces nombreuses conséquences occasionnées vers la thématique des maladies des plantes. La détection de manière précise et rapide, en utilisant les nouvelles technologies, permettrait de réduire les différentes pertes à travers un traitements localisé et très efficace. Après identification de ce problème dont la résolution est notre objectif nous avions procéder à une étude de l'existant suivit d'une étude critique de ces différentes solutions déjà mise en place, ensuite nous avions entamer la mise en place de la solution.

2.3 Etat de l'art

2.3.1 Solutions existantes

Des systèmes de détection des maladies des plantes construit sur la base de nouvelle technologie existe. Elle peut être classé essentiellement en deux catégories à savoirs les applications téléphoniques et les drones.

2.3.1.1 Les applications

Les applications sont mises en place de telle sorte qu'elle puisse à partir d'une capture d'image donner une analyse instantanée sur l'état de santé de la plante. Parmi les applications les plus rependue on peut citer :

a- Nuru :

Un assistant IA pour les agriculteurs. Nuru à trois objectifs : détection des maladies des cultures, prévision d'anomalies en utilisant certaines informations, et la compréhension du langage humain et les réponses automatiques aux questions posées par les agriculteurs.

Figure 1 : Interface de l'application Nuru

.

b- **Plantix** :

Une application mobile pour le diagnostic de Cassava Diseases. C'est une IA mise en place par PEAT GmbH, une Startup basé à Merlin.

Figure 2 : Interface de l'application Plantix

2.3.1.2 Technologies

c- **Drone DJI:**

On distingue essentiellement les Drones Dji dont la technologie permet une capture et une très bonne analyse des images. Les Drones de DJI, Equipés d'appareils photo 20 mégapixels et de caméras multispectrale /thermique, capturent des images claires et détaillées qui sont essentielles à l'analyse.

Figure 3 : Drone DJI

.

d- **Une combinaison de capteur et logiciel :**

AQiT-sensor de Carbon Bee est une solution unique de détection de stress et maladie sur végétal.

La solution est composée de :

- **AQit-sensor :** un capteur électronique, permettant de capturer des données depuis un drone, un enjambeur ou un robot. Il s'agit de capter comment le végétal (feuilles, baies, rameaux) reflète la lumière dans le spectre visible à l'œil humain, et le non visible.

- **AQiT-map :** un logiciel de traitement d'image, permettant l'analyse de « signatures » de maladies et de stress, reposant sur des algorithmes d'intelligence artificielle. Cela permet d'identifier avec une grande précision les zones détectées dans les parcelles.

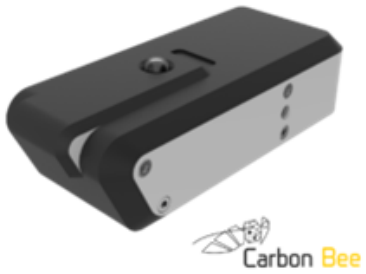

Figure 4 : Système Carbon Bee

2.3.2 Comparaison des solutions existantes

Les solutions bien que diverse et multiple présente un certain nombre de limites. En effet, les applications, en plus de ne pas traiter toutes les plantes, tels que la pomme et la pomme de terre, fournissent un résultat après une très longue période ce qui ne contribue pas à une minimisation des dégâts engendrer par les infections. Également, ces différentes applications ne savaient pas différencier les objets, tel une feuille blanche, des feuilles de plantes. Les deux autres types de solution, en plus de ne pas répondre au critère du traitement de toute les plantes, sont très couteuses ce qui rend leur accessibilité aux agriculteurs à moyen revenue très difficile.

.

Solutions	Avantages	Désavantages
Plant village Nuru	Traitent une grande majorité de maladie, Est accessible et gratuit, Système vocal pour favoriser l'interaction	Temps de traitement pouvant aller jusqu'à une journée, n'est pas adapté au contexte Marocain du faite de la non-disponibilité de langue arabe et du fait du non-traitement de plante ayant un fort impact sur l'économie Marocaine.
Plantix	Offre des possibilités multiple en matière de langue, est accessible et gratuit	Temps de traitement long, biaisé
Drone DJI	Traitement à grande échelle, prise en compte d'un nombre assez important de plante, utilisation de caméra hyperspectral qui offre d'autre possibilité.	Cout élevé
Système Carbone Bee	Traitement précis du fait de la combinaison de deux technologie, possibilité de traité à distance en utilisant le capteur	Cout élevé, nécessité de disposé d'appareil de haute gamme afin de faciliter le traitement.

Tableau 4 : Récapitulatif des solutions existantes

Etant donné les différentes solutions existantes et en vertu des contraintes de temps ainsi que des matériels dont nous disposons nous avions orienter notre solution vers le

développement informatique avec pour objectif d'améliorer les solution de type application disponible en utilisant par la même occasion le Deep Learning.

2.3.3 Etude théorique

2.3.3.1 Notion de Deep Learning

Le Deep Learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. Le deep Learning s'appuie sur un réseau de neurones (Un neurone est une cellule du système nerveux spécialisée dans la communication et le traitement d'informations) artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit.

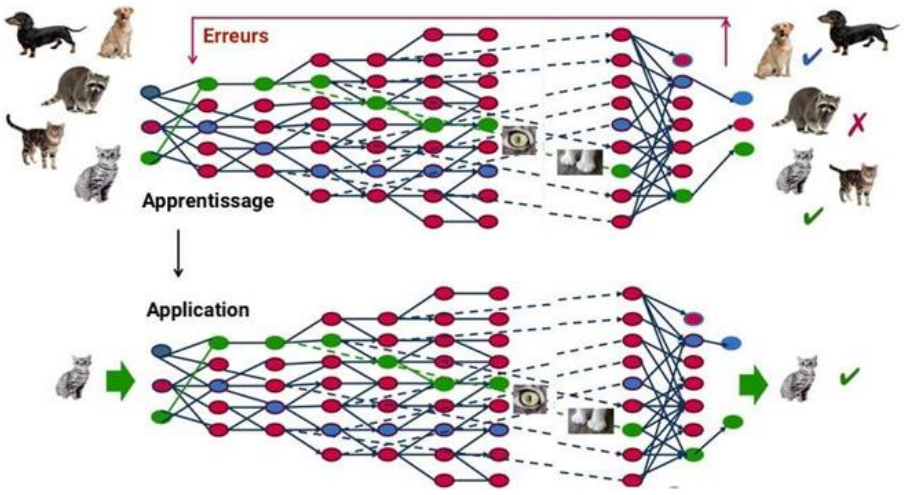

Figure 5 : Illustration d'un modèle de deep Learning

À travers un processus d'autoapprentissage, le deep Learning est capable d'identifier un chat sur une photo. À chaque couche du réseau neuronal correspond un aspect particulier de l'image.

Dans cette partie, nous allons nous focaliser sur un des algorithmes les plus performants du Deep Learning, les Convolutional Neural Network ou CNN : Réseaux de neurones convolutifs en français, ce sont des modèles de programmation puissants permettant notamment la reconnaissance d'images en attribuant automatiquement à chaque image fournie en entrée, une étiquette correspondant à sa classe d'appartenance.

2.3.3.2 Convulotional Neural Networks (CNN) :

Le réseau de neurones convolutifs (CNN) est un type de réseau de neurones de prédiction inspiré du mécanisme cognitif visuel biologique. C'est l'une des approches d'apprentissage en profondeur les plus populaires dans le domaine du traitement graphique car CNN est très performant dans le traitement d'images et traite directement des images brutes. CNN extrait les caractéristiques de l'image et compresse le volume de données par des opérations, telles que la convolution, la mise en commun, etc. Le modèle est entraîné par descente de gradient et rétropropagation algorithmique, afin qu'il puisse réaliser des fonctions comme la classification d'images. Généralement, cinq couches constituent l'architecture principale du CNN :

- **Couche d'entrée** : C'est l'entrée des données d'images brutes. Dans cette couche, les images peuvent être prétraitées en utilisant des opérations, y compris la normalisation, l'analyse des composants principaux et le blanchiment. Le prétraitement rend les images normatives, ce qui permet d'accélérer l'apprentissage du réseau modèles et élève ainsi les performances du modèle.

- **Couche de convolution :** Il s'agit de la couche principale d'un CNN, qui effectue la convolution sur les images entrées pour en extraire les caractéristiques. Généralement, une couche de convolution contient plusieurs convolutions noyaux comme filtres afin qu'il puisse obtenir plusieurs résultats de caractéristiques d'image.

- **Couche d'activation :** Cette couche est utilisée pour le mappage non linéaire des résultats de convolution de sorte que le réseau multicouche peut être non linéaire et a une meilleure capacité d'expression. Les fonctions d'activation couramment utilisé sont la fonction Relu et la fonction Sigmoïde.

- **Couche de mise en commun** : Ceci est également connu sous le nom de couche de sous-échantillonnage, et c'est la partie qui conduit à la réduction de la dimensionnalité pour la fonctionnalité extraite et la compression des données, de sorte que le surapprentissage puisse être réduit et la tolérance aux pannes du modèle peut être améliorée. Les méthodes de pooling(mise en commun) incluent MaxPooling et AveragePooling, et MaxPooling est couramment utilisé maintenant.

- **Couche entièrement connectée :** C'est la couche de sortie de résultat qui réalise la classification d'objet une fonction. Cette couche intègre les informations sur les caractéristiques de chaque neurone de la partie supérieure couche et classe les images en fonction de l'objectif. Il existe généralement deux sortes des fonctions de classification, la fonction Sigmoïde pour la classification binaire et la fonction Softmax pour le classement multiple.

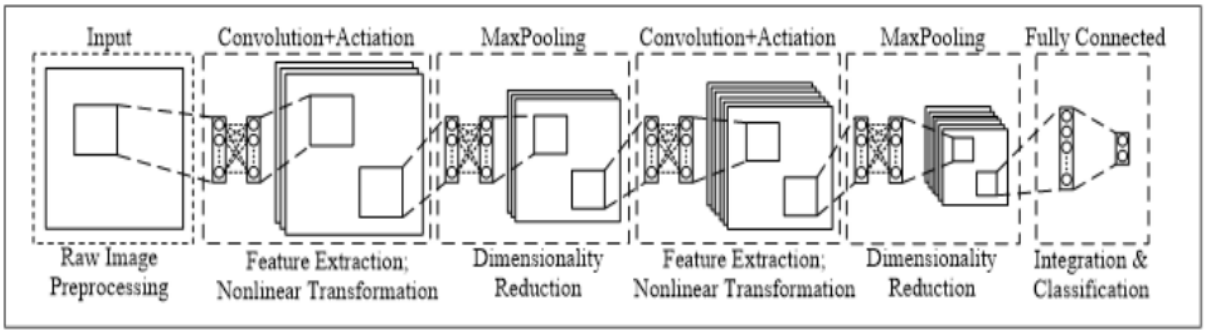

Figure 6 : Procédure de fonctionnement des réseaux de neurones convolutifs

2.3.3.3 Transfert learning

Le Transfer Learning, ou apprentissage par transfert en français, désigne l'ensemble des méthodes qui permettent de transférer les connaissances acquises à partir de la résolution de problèmes donnés pour traiter un autre problème.

Le Transfer Learning a connu un grand succès avec l'essor du Deep Learning. En effet, bien souvent, les modèles utilisés dans ce domaine nécessitent des temps de calcul élevés et des ressources importantes. Or, en utilisant des modèles pré-entraînés comme point de départ, le Transfer Learning permet de développer rapidement des modèles performants et résoudre efficacement des problèmes complexes en Computer Vision ou Natural Langage Processing, NLP.

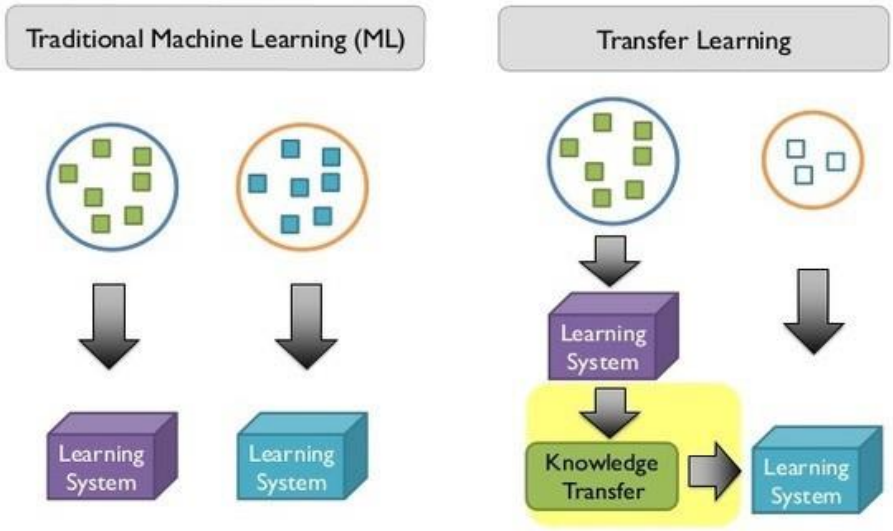

Figure 7 : Approche traditionnelle vs. Approche de Transfert Learning

2.3.3.4 Notion de bibliothèque

Une bibliothèque renvoie à un ensemble de module écrite en un langage donné et permet d'effectuer plus aisément un certain nombre de tache. Pour la mise en place de notre modèle de deep Learning nous avions utilisé une bibliothèque spécifique à la mise en place des réseaux de neurones, à savoir Keras. En effet Keras est une bibliothèque open source écrite en langage

Python agissant au niveau du modèle : elle **met à disposition des modules** permettant de développer des modèles de deep learning (apprentissage profond) complexes. Contrairement aux frameworks (Un **framework** (ou infrastructure logicielle en français) désigne en programmation informatique un ensemble d'outils et de composants logiciels à la base d'un logiciel ou d'une application.) indépendants, ce logiciel open source ne s'occupe pas des opérations « low level » et utilise à cet effet les bibliothèques de frameworks d'apprentissage automatique associés qui tiennent pratiquement lieu de moteurs back-end pour Keras. Conformément au **principe de modularité**, les couches désirées pour le réseau neuronal à mettre en place sont connectées. Pour autant, il n'est pas nécessaire de comprendre l'infrastructure réelle du framework choisi et l'utilisateur de Keras n'a pas à la démarrer directement.

Comme indiqué précédemment, Keras repose essentiellement sur les trois outils TensorFlow, Theano et Microsoft Cognitive Toolkit, qui disposent d'ores et déjà d'**interfaces prêtes à l'emploi** permettant un **accès rapide et intuitif à l'infrastructure concernée.**

2.3.4 Etude comparé des algorithmes possibles

La mise en place de notre solution qui est un algorithme de détection des maladies des plantes, tout en ayant pour cible majeur la pomme et la pomme de terre, c'est fait en suivant un certain nombre de principe et en adoptant un certain nombre de compromis. Dans l'optique de fournir une solution offrant une très grande précision tout en respectant le délai qui nous étais impartis nous étions contraints de nous orienter vers le transfert Learning, qui consiste à utiliser un algorithme déjà entrainer pour l'adapter à notre contexte à travers une modification et un ajout de couche supplémentaires (dit layers en anglais). Pour un choix d'algorithme ayant une meilleure performance nous avions effectué une étude comparer des avantages et inconvénients des différents algorithme que nous pouvions utiliser.

Les algorithmes auquel nous nous sommes intéressés sont des réseaux pré-entrainés à l'intérieur de Keras, une bibliothèque open source écrite en python, ils sont capables de reconnaitre avec une grande précision 1 000 catégories d'objets différentes.

Algorithme pré-entrainé	Descriptifs	Avantages	Inconvénients
VGGNet(VGG 16 et VGG19)	Deux couches entièrement connectées, chacune avec 4 096 nœuds. Les « 16 » et « 19 » représentent le nombre de couches de poids dans le réseau. La taille du VGG16 est supérieur à 533 MO, celle de VGG19 supérieur) 574 MO	Simple, utilisation seulement de 3*3 couches convolution empilée les unes sur les autres dans l'augmentation de la profondeur,	Il est très lent à entrainer, les poids de l'architecture réseau eux-mêmes sont assez importants (en termes de disque/bande passante) ce qui rend le déploiement du VGG fastidieux.
ResNet	Il s'agit d'un modèle qui repose sur des modules de microarchitecture. Taille :102 MO, 50 couches de poids.	Beaucoup plus profond, taille réduit, Est plus facile à former Est plus tolérant envers les hyperparamètres, y compris la régularisation et le taux d'apprentissage initial Mieux généraliser	Dégradation des performances après une certaine profondeur, lent
Inception vN	L'objectif du module de Inception est d'agir comme un « extracteur de caractéristiques à plusieurs niveaux » en calculant *1 x 1*, *3 x 3* et *5 x 5* convolutions au sein du *même* module du réseau. N désigne le numéro de version publié par google. Les poids pour Inception V3 sont inférieurs à ceux de VGG et ResNet, atteignant 96 Mo. Le modèle lui-même est constitué de composants de base symétriques et asymétriques incluant	Beaucoup plus profond, taille réduit, plus précis, Temps de convergence réduit par rapport aux autres modèles, Dans le cas d'Inception v3, le taux d'apprentissage est initialement établi à environ 10 %, Adaptation du Taux d'apprentissage, consommation de moins de puissance de calcul.	

	convolutions, pooling moyen, pooling maximal, concaténations, abandons et couches entièrement connectées. La normalisation par lots (batchnorm) est amplement utilisée dans le modèle et appliqué aux entrées d'activation. La perte est calculée via Softmax.		
DenseNet	Un DenseNet est un type de réseau neuronal convolutif qui utilise des connexions denses entre les couches, via des blocs denses, où nous connectons toutes les couches (avec des tailles de carte de caractéristiques correspondantes) directement les unes avec les autres. Pour préserver la nature d'anticipation, chaque couche obtient des entrées supplémentaires de toutes les couches précédentes et transmet ses propres cartes de caractéristiques à toutes les couches suivantes.	Ils atténuent le problème du gradient de disparition, renforcent la propagation des caractéristiques, encouragent la réutilisation des caractéristiques et réduisent considérablement le nombre de paramètres.	Il est très lent à entrainer, sa performance es faible par rapport aux précédents algorithmes.
AlexNet	AlexNet est la première architecture de réseau de neurones convolutifs à grande échelle qui fonctionne bien sur la classification ImageNet. AlexNet a participé à la compétition et a été en mesure de surpasser tous les modèles précédents basés sur l'apprentissage non approfondi par une marge significative. L'architecture AlexNet est une couche de conv suivie d'une couche de pooling, d'une normalisation, de conv-pool-norm, puis de quelques couches	AlexNet permet une formation multi-GPU en plaçant la moitié des neurones du modèle sur un GPU et l'autre moitié sur un autre GPU. Non seulement cela signifie qu'un modèle plus grand peut être formé, mais cela réduit également le temps de formation. Chevauchement de la mise en commun.	Sa précision est faible par rapport aux autres modèle qi sont plus récent

	de conv supplémentaires, d'une couche de pooling, puis de plusieurs couches entièrement connectées par la suite. Ressemble en fait très au réseau LeNet.		

Tableau 5 : Tableau de comparaison des différents algorithmes

3 Mise en place de la solution

Notre choix de la solution finale était basé sur plusieurs considérations.

3.1 Choix des maladies à traiter :

Nous avons choisi de traiter le cas des maladies dont les symptômes ont un grand degré de similarité, vu que ce sont ces maladies-là qui présentent un défi pour les agriculteurs qui utilisent la détection à l'œil nue. De ce fait, les choix finaux étaient :

- Pour les pommes :

Rouille de cèdre du pommier
(Apple Cedar Rust)

Pourriture noire de la pomme
(Apple Black Rot)

Figure 8 : Maladies traitées pour la pomme

On remarque que les deux maladies ont pour symptômes au niveau des feuilles des taches de petite taille et de couleur orange-brune, ce qui les rend indiscernable à l'œil nue, de ce fait, l'usage d'un algorithme pour leurs détections est préférable.

- Pour les pommes de terre :

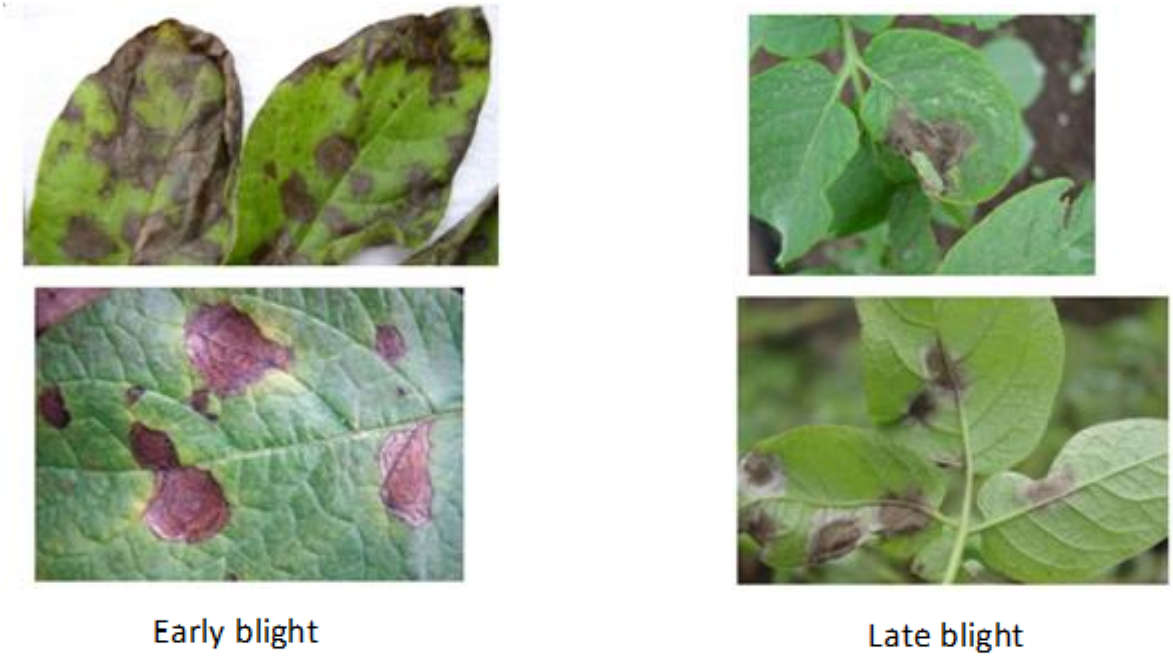

Figure 9 : Maladies traitées pour la pomme de terre

Ces deux maladies présentent quasiment les mêmes symptômes, rendant la différenciation entre eux presque impossible à l'œil nue. De même, automatiser ce processus serait préférable.

3.2 Choix des CNNs à utilisés :

Comme on a détaillé avant, notre algorithme sera à base des réseaux de neurones convolutionnels (CNNs) déjà existants (principe de Transfer Learning), et qui vont être personnalisés pour notre cas spécifique, à savoir la détection des 4 maladies mentionnés ci-dessus. Pour le faire, nous avons choisi d'implémenter VGG16, Inception V3 et ResNet, tous connus pour leurs performances exceptionnelles. Dans la prochaine partie, une étude

comparative sera menée des trois modèles pour choisir et déployer celui qui est le plus performant.

Il est utile de mentionner aussi que notre algorithme sera programmé en langage python, qui, en plus d'être le langage que tout le groupe projet maitrise, possède des librairies riches comme Keras et TenserFlow, et bien entendu les modèles eux-mêmes que nous allons utiliser, i.e. VGG16, Inception V3 et ResNet.

3.3 Choix des langages pour la partie frontend

L'algorithme ne pouvant pas être forcément compréhensible par l'agriculteur et pour des soucis d'ergonomie nous avions décidé de mettre en place un site web qui sera relier à l'algorithme et avec lequel l'agriculteur interagira. Pour la mise en place de ce site web nous avions utilisé une combinaison de plusieurs langages à savoir HTM (), CSS, JavaScript. En effet, Le langage de balisage hypertexte ou HTML est un langage de programmation utilisé pour décrire la structure des informations sur une page Web, le CSS contrôle l'apparence d'une page et le JavaScript permet de programmer ses fonctionnalités. On peut donc considérer que le code HTML constitue la base d'une page Web, tandis que les feuilles de style CSS en constituent la peau et que JavaScript en constitue le cerveau.

3.4 Choix des bases de données :

Les bases de données choisies sont toutes disponibles sur Kaggle, et leurs liens seront en références.

Il faut noter aussi que notre modèle doit être capable à reconnaitre les images dites «random», c'est-à-dire celles ne contenant pas des feuilles de plantes mais des objets complétement aléatoires, et ceci pour éviter que le modèle retourne un résultat de détection de maladie lorsque l'image ne contient même pas une feuille. Pour ce faire, une autre classe doit être ajoutée à notre modèle qui va être entrainé par des images aléatoires prises d'une base de données d'images arbitraires faite pour cette raison.

3.5 Concept clés pour la mise en place de la solution

La prochaine étape consistera à effectuer des tests sur les trois modèles pour comparer leur performance. Cette comparaison nécessitera la compréhension de plusieurs concepts clés dans le domaine de Machine Learning.

- **Loss function :** mesure la distance entre une valeur estimée et sa valeur réelle. Une fonction de perte associe les décisions aux coûts qui leur sont associés. Les fonctions de perte ne sont pas fixes, elles changent en fonction de la tâche à accomplir et de l'objectif à atteindre.

- **Freezing Layers :** c'est une technique permettant d'accélérer la formation des réseaux de neurones en « gelant » progressivement les couches cachées. Par exemple, lors de Transfer Learning, les premières couches du réseau sont « gelées » tout en laissant les dernières couches ouvertes à la modification.

- **Overfitting :** ou le surapprentissage, désigne le scénario dans lequel un modèle ne peut pas généraliser ou s'adapter correctement à un ensemble de données non vues. Un signe évident de ce phénomène est lorsque l'erreur du modèle sur l'ensemble de données de test ou de validation est beaucoup plus grande que l'erreur sur l'ensemble de données d'entrainement.

- **Epochs number :** Le nombre d'époques est le nombre de passages complets dans l'ensemble de données d'apprentissage.

- **Justesse (Accuracy) :** défini comme le pourcentage de prédictions correctes pour les données de test. Elle peut être calculée facilement en divisant le nombre de prédictions correctes par le nombre total de prédictions.

4 Réalisation et tests

4.1 Backend

4.1.1 Data augmentation :

Comme indiquée, Nous avons choisis une base de données qui contient 38 classes et l'avons réduit aux seules maladies qui nous intéressaient (Tableau 1). Et afin que notre algorithme soit capable de reconnaitre une image d'un objet autre qu'une plante comme tel, nous avons décidé d'ajouter une autre classe 'This is not a plant', que l'on a rempli d'images aléatoires. On se retrouve ainsi avec un total de 19 639 images divisés en un sous ensemble d'entrainement et un autre de validation.

Plante___Maladie	Training Data	Validation Data	Total
Apple___Apple_scab	2016	504	2520
Apple___Black_rot	1987	497	2484
Apple___Cedar_apple_rust	1760	440	2200
Apple___healthy	2008	502	2510
Potato___Early_blight	1939	485	2424
Potato___healthy	1824	456	2280
Potato___Late_blight	1939	4805	2424
This is not a plant	2047	750	2797
All Types	15520	4119	19639

Tableau 6 : Détails sur les caractéristiques de la base de données

Cependant, il était nécessaire d'augmenter d'avantage le nombre de données desquels on dispose. On a donc fait appel à la classe ImageDataGenerator dont dispose Keras, et qui nous permet d'augmenter le nombre de données qu'on a en prenant chaque image et en la dupliquant plusieurs fois tout en effectuant des transformations de translations, de zoom, de symétries et d'autres types.

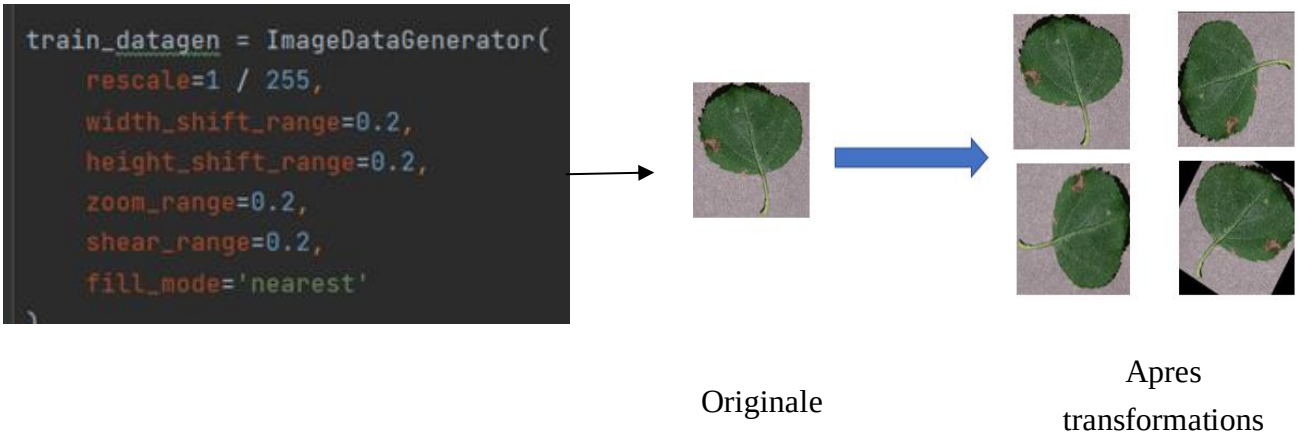

Figure 10 : : Augmentation de la base de données

4.1.2 Transfer Learning :

Dans cette partie, nous expliquerons brièvement les étapes réalisées afin d'importer un modèle CNN et de l'adapter à notre problématique. Pour se faire nous traiterons ici le cas du VGG16, sachant que pour les autres modèles la méthode et pratiquement similaire.

a. Importation du modèle :

On importe notre modèle pré-entrainé en choisissant la forme des images que le modèle prendra comme input et le type de paramètres weights qu'il utilisera.

```
#Creating Instence of pre-trained model from Keras Application
inception_model = InceptionV3(input_shape=(150, 150, 3),
                              include_top=False,
                              weights=None)

inception_model.load_weights(local_weights_file)
```

Figure 11 : Importation du modèle

b. « Freezing » toutes les couches du modèle

C'est là tout l'avantage du Transfer Learning, on importe un modèle déjà entrainée et on n'entraine que les dernières couches qu'on rajoutera plus tard, ce qui permet de gagner énormément de temps.

```
# Freezing all Layers of Inception V3 Model
for layer in inception_model.layers:
    layer.trainable = False
```

Figure 12 : Parcours des différentes couches dans le modèle

c. Aplatir la dernière couche

La dernière couche du IV3 est un tenseur de dimension 3 : (None, 3, 3, 2048). Or nous ce qu'on veut c'est avoir un output sous la forme d'un tenseur de dimension 1 qui contient les prédictions d'appartenances à chaque classe. Il est donc nécessaire d'aplatir ce tenseur de dimension 3 en un tenseur de dimension 1 de forme (None, 18432). Notons que 18432 = 3*3*2048.

```
# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
```

Figure 13 : Aplatissement de la dernière couche

Après application de cette fonction, on obtient :

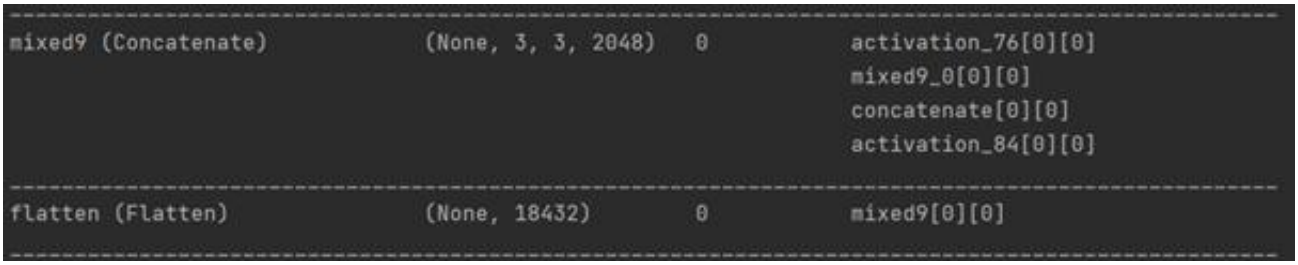

Figure 14 : Présentation des couches après aplatissement

d. Ajout des dernières couches

On ajoute graduellement des couches au modèles en diminuant à chaque fois la taille, la dernière couche doit nécessairement être de la forme (None, 8) puisque nous disposons de 8 classes.

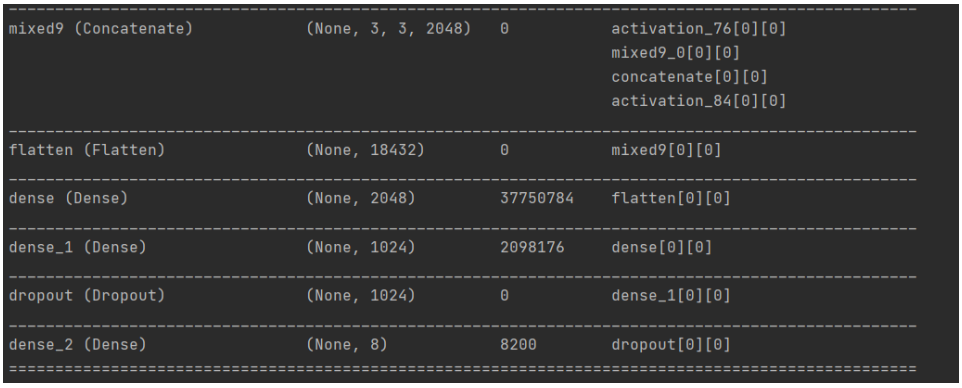

Figure 15 : Résultats après ajout des différentes couches

4.1.3 Etude comparatifs des trois algorithmes :

Maintenant que nous disposons de trois algorithmes opérationnels, il nous est nécessaire de déterminer le plus performant afin de l'instaurer dans notre solution finale. Nous avons ainsi mené une étude comparative qui reposent sur quatre aspects :

- **Le temps d'entrainement :** Combien de temps les modèles ont pris pour s'entrainer en 20 epochs.

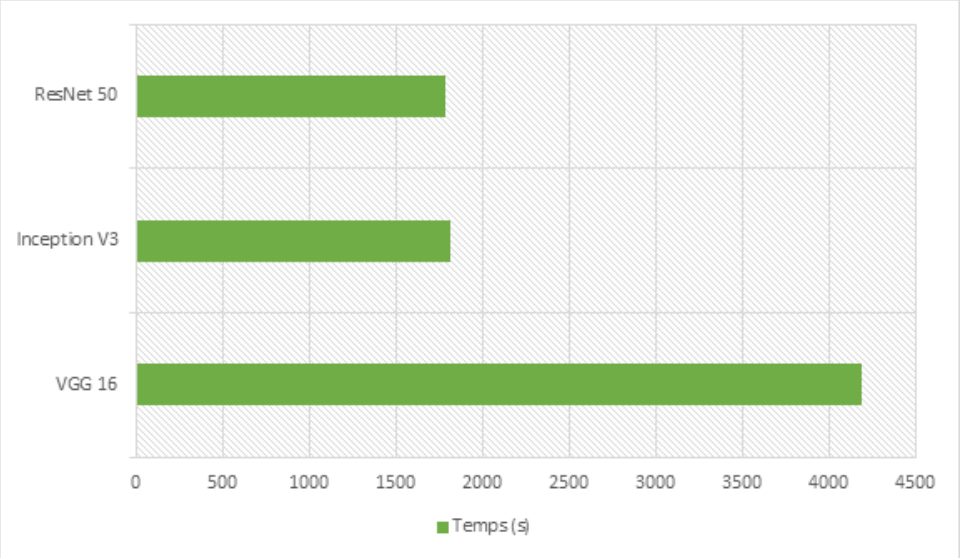

Figure 16 : Comparaison du temps d'exécution des trois modèles utilisés

Il est évident que parmi les 3 modèles le VGG16 est le plus long, tandis que les deux autres sont à peu près égaux.

- **Le changement en accuracy (performance) :** Généralement, dans le processus de formation des modèles d'apprentissage profond, avec la mise à jour constante des paramètres des neurones, l'accuracy du modèle dans la classification des maladies des plantes est élevée. Pour refléter les changements d'accuracy pendant la formation, deux courbes d'accuracy basées sur l'ensemble d'entrainement lui-même et l'ensemble de validation, respectivement, ont été adoptées pour chaque modèle, L'analyse des deux courbes de précision permet d'évaluer les performances de formation du modèle. Les modèles dont la fluctuation de la courbe est plus faible pendant l'augmentation de l'accuracy présentent une meilleure convergence de formation, et les modèles dont les deux courbes d'accuracy de l'ensemble de formation et de l'ensemble de validation sont plus proches et dont l'accuracy est plus élevée présentent de meilleures performances de formation. Les courbes d'accuracy de chaque modèle dans le processus sont illustrées à la figure 7.

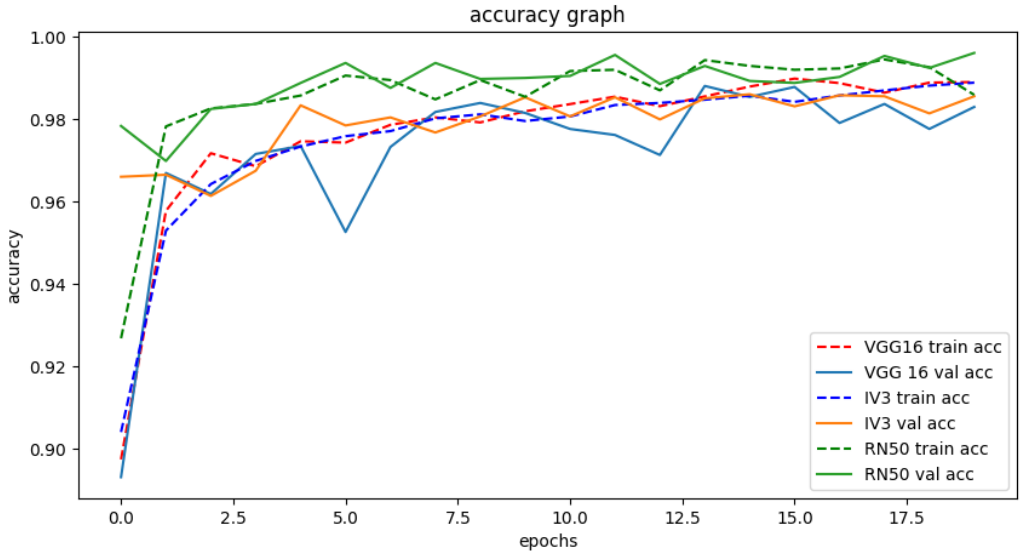

Figure 17 : Courbe de justesse combinée

- **Pertes et surapprentissage :** Lorsque les pertes sur l'ensemble de validation sont largement supérieures aux pertes sur l'ensemble d'apprentissage on dit que le modèle souffre de surapprentissage. La figure 8 illustre les différentes courbes de pertes des modèles.

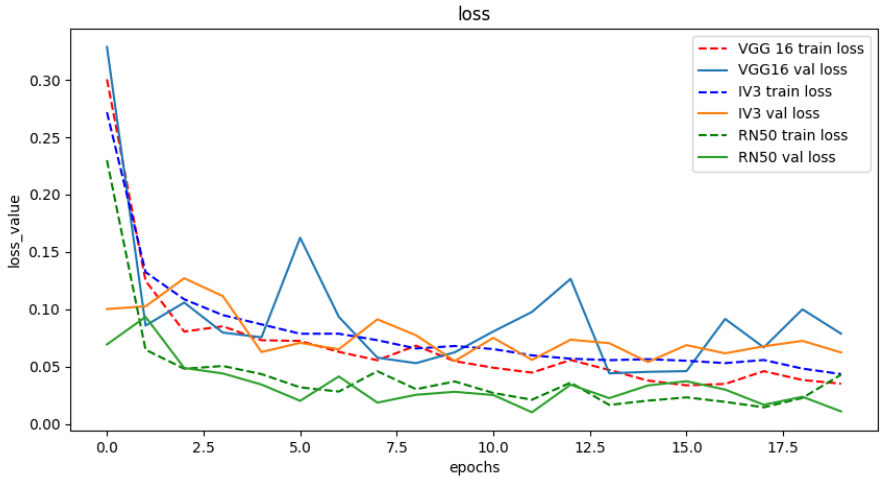

Figure 18 : Courbes de pertes des différents modèles

On remarque que les courbes de pertes des modèles IV3 et RN50 ont à peu près la même convergence décroissante respectivement pour l'ensemble de validation et celui d'apprentissage. Tandis que la courbe de perte sur l'ensemble de validation du modèle VGG16 commence à diverger de la courbe sur l'ensemble d'apprentissage à partir de cinq epochs.

Ces premières comparaisons nous ont permis d'éliminer le VGG16 comme candidat à la solution finale. Il était donc nécessaire de tester ces algorithmes en temps réels.

- **Tests des 3 modèles sur un ensemble de tests :** Finalement nous avons testé ces trois algorithmes sur un ensemble de 31 images qu'on a soit pris d'internet soit capturer par nous-même. Ci-dessous les résultats obtenus pour chaque modèle sous forme de matrices de confusions.

4.2 Frontend

4.2.1 Création du Front end de l'application web en utilisant HTML et CSS

Tout au long de de cette étape nous avions mis en place le serveur frontal de notre application web en utilisant trois langages de programmation à savoir : HTML, CSS et Javascript.

En effet, nous avons utilisé le langage de balisage hypertexte, ou HTML pour décrire la structure des informations sur notre page Web. Notre page Web contient des titres, des paragraphes, des images, des vidéos et de nombreux autres types de données. C'est pour cela nous avons utilisés des éléments HTML pour spécifier le type d'informations que contient chaque élément d'une page Web - par exemple, l'élément HTML "p" indique un paragraphe. L'élément HTML <header> représente le contenu d'introduction, généralement le groupe d'aides à l'introduction ou à la navigation. Il contient les éléments d'en-tête (Home, analyse, team, about.) mais aussi le logo. L'élément <footer> représente le bas de notre page. Et L'élément <script>…</script> permet d'inclure du code de script (dans notre cas le code JavaScript)

A cet effet dans l'image suivante vous verrez illustrer la première version d'une page de notre site en partant seulement du code HTML :

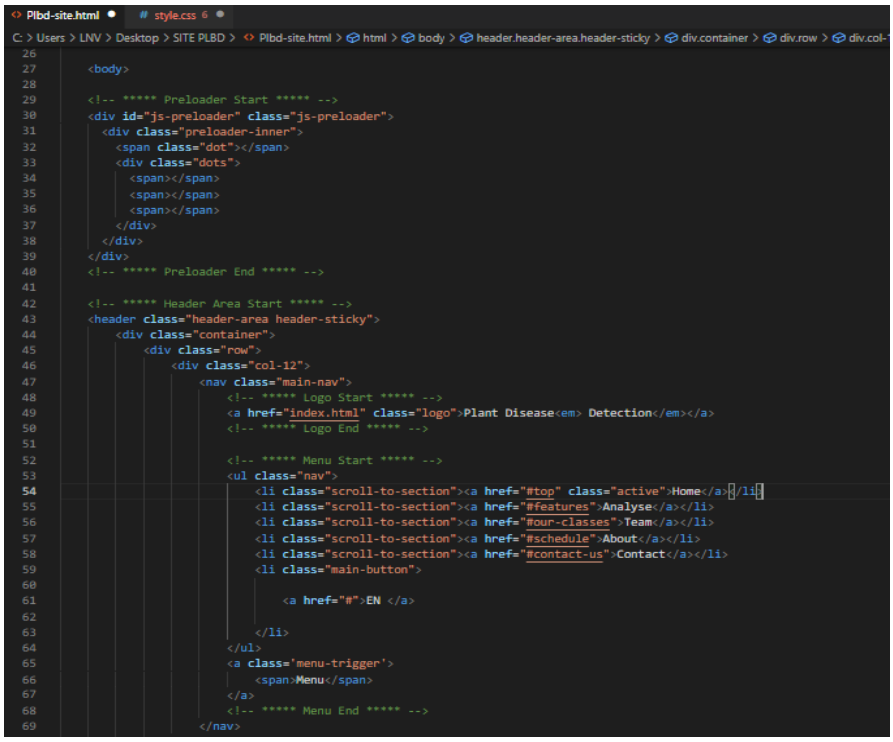

Figure 20 : Code de la première page

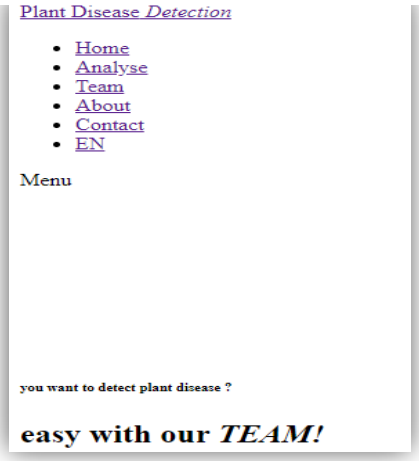

Figure 19 : Première page avec le code HTML

Ensuite nous avons utilisé le langage CSS pour décrire la présentation de nos pages Web, notamment les couleurs, la mise en page et les polices. Il nous a permis d'adapter la présentation à différents types de dispositifs, tels que les grands écrans, les petits écrans ou les imprimantes. Pour ce faire nous avons créé des fichiers CSS indépendants de HTML nommés "*style.css*", "*body.css*", "*bootstrap.min.css*", " *font-awesome.css*". Après cela, nous avons utilisé L'élément <link> pour créer les liens avec ces feuilles de style externes, en utilisant la formulation suivante : `<link rel="stylesheet" type="text/css" href="url">`

```
<!-- linnking CSS Files -->

<link rel="stylesheet" type="text/css" href="assets/css/bootstrap.min.css">

<link rel="stylesheet" type="text/css" href="assets/css/font-awesome.css">

<link rel="stylesheet" href="assets/css/style.css">

<link rel="stylesheet" href="assets/css/body.css">
```

Figure 21: Mise en relation des différents codes pour le site

Après avoir exécuter l'ensemble des codes on obtient le design final de notre site web :

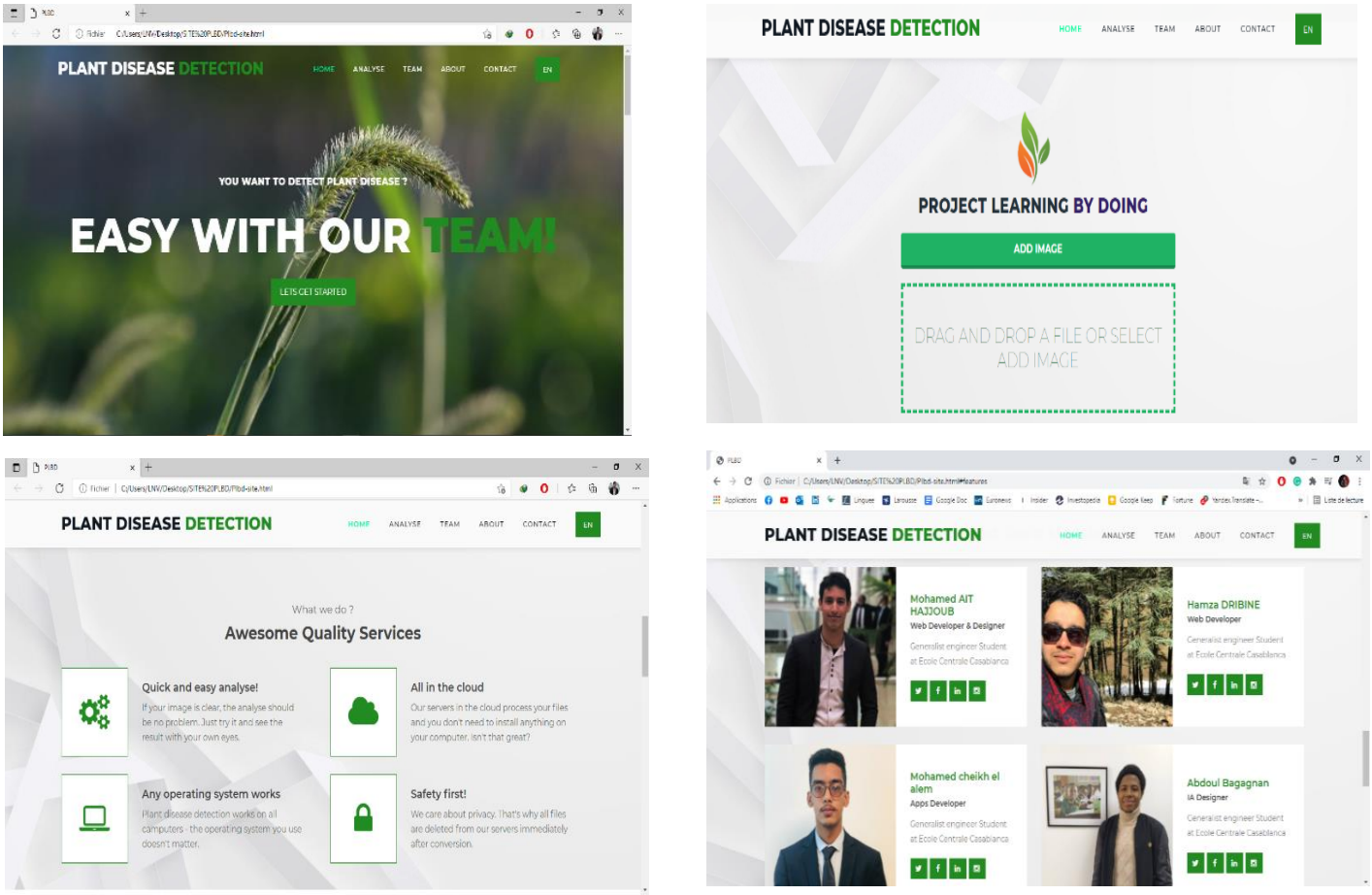

Figure 22 : Présentation des différents pages de notre site web

4.2.2 Écriture de la fonctionnalité en Javascript

Le code java script nous à permis de créer une expérience dynamique et interactive avec l'utilisateur. Les différentes fonctions utilisées nous ont permis d'assuré les différentes fonctionnalités de notre site web.

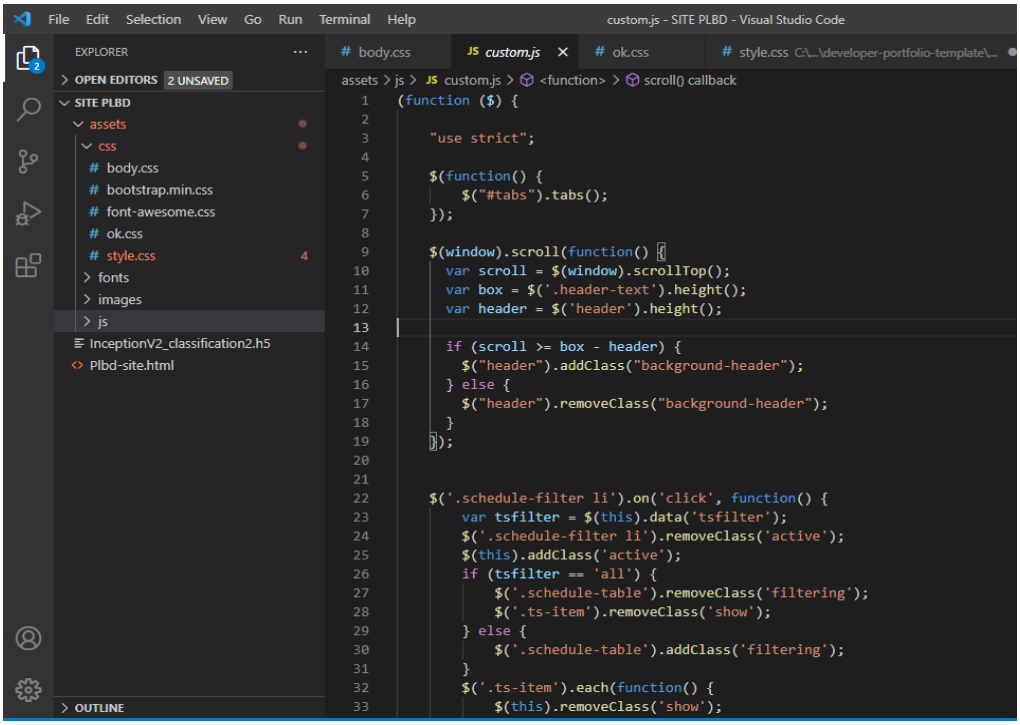

Figure 23 : Illustration d'un bout de code du programme écris en javascript

L'image ci-dessus illustre un bout de code pour la partie java-script et permet également de percevoir les différents codes dont nous avions fait allusion précédemment.

5 Résultats et Commentaires

5.1 Résultats obtenus

Les différents courbes tracés précédemment ainsi que la matrice de confusion nous ont permis d'orienter notre choix final vers Inception v3 qui présente une plus grande performance lors des tests ainsi qu'un temps d'entrainement assez réduit. En effet sur 30 images nous avions obtenu 27 bonne prédiction contrairement aux autres modèles.

5.2 Apports de notre solution par rapport à l'existant

Limites des solutions existante	Apport de notre solution
Temps de traitement très élevé	L'utilisation de l'Inception V3 nous assure un diagnostic dans de bref délai
Ne prend pas en compte des plantes ayant un fort impact dans l'économie Marocaine	Notre solution a pour but de traiter spécifiquement les plantes ayant une place de choix dans l'économie Marocaine
Précision réduit du fait de l'utilisation dans la plupart des cas du VGG 16	L'Inception au niveau des Test nous assure une très bonne prédiction et son taux de performance au niveau de la validation ne peut qu'illustrer la précision élever.
Solution biaisée du fait de la non-différenciation entre une plante et un autre objet	L'utilisation d'une classe supplémentaire formé d'image aléatoire nous permet de résoudre ce problème

	biais et d'identifier par conséquent s'il s'agit ou pas d'une plante.
Pas adapter à la culture Marocaine en termes de Langue	La proposition de langue varié à savoir, l'anglais, le français et le darija nous permet de pallier cette limite initialement constatée.
Manque de tutoriel expliquant le fonctionnement du dispositif	Nous proposons un tutoriel disponible sur le site qui permet pour une première connexion de guider l'utilisateur sur l'utilisation de notre application web
Absence de proposition de définition ainsi que de remède	Notre solution après diagnostic, propose une définition précise de la maladie en évoquant les éventuelles pathogène et propose par la même occasion des remèdes pour le traitement.
Cout élevé de certaines solutions tels que Aqit-sensor de carbon bee et les drones Dji	Notre solution est gratuite et disponible en ligne à la porter de tous les agriculteurs
Disfonctionnement de certaines solutions. En effet des utilisateurs ont effectuer un certain nombre de commentaire sur la page YouTube de l'application plant village soulignant un disfonctionnement de l'application sans avoir obtenu de retour	Nous avions mis à disposition dans notre site web nos différents contact permettant aux agriculteurs de pouvoir nous contacter directement en cas de disfonctionnement dans leur localité.
Défaut de qualité de la base de données utiliser par certaines solutions. En effet au cours de nos recherche nous avons noté que certaines bases de donner contenait des images ne donnant pas assez une très bonne illustration de la maladie en question (exemple : images renfermé).	Pour la mise en place de notre solution, nous avions utiliser plusieurs bases de données en prenant le soin de bien vérifier la qualité des différentes images afin d'assurer une très bonne performance pour notre application.

Tableau 7 : Résolution des différentes limites soulevé par notre solution

6 Conclusion Générale

Notre projet avait pour objectif final de construire une solution permettant d'identifier et diagnostiquer 4 maladies des pommes et pommes de terre. Notre apport personnel réside d'abord dans notre choix des maladies à cibler, à savoir des pathologies qui ont des symptômes très similaires rendant la différenciation entre eux à l'œil nue difficile pour les agriculteurs. De ce fait, nous sommes arrivés à construire une solution capable de diagnostiquer ces 4 maladies avec une précision de 98.89%. En outre, cette solution est déployée dans un site web accessible à l'agriculteur et facile à utiliser, et qui permet en outre d'afficher les résultats du diagnostic, ainsi qu'une brève définition de la maladie et des remèdes possibles pour celle-ci.

Or, nous avons rencontré plusieurs difficultés lors de notre travail sur ce projet, la difficulté majeure étant le manque de connaissances approfondies dans le domaine d'informatique en général, et dans le domaine de Deep Learning en particulier. De ce fait, le progrès effectué le long de la durée de notre travail fut lent. Cependant, les remarques et conseils des différents enseignants lors des soutenances, ainsi que ceux de notre tuteur ont fait en sorte de nous motiver à cerner notre but final pour ne pas s'égarer et de mieux départager les tâches et les responsabilités.

Finalement, ce projet nous a permis de s'initier au domaine fascinant de l'intelligence artificielle. En effet, nous avons pu apprendre les bases d'une technologie récente et prometteuse, ainsi qu'appliquer ces bases pour résoudre une problématique réelle avec des enjeux concrets. En outre, nous avons appris à travailler en équipe à travers les nombreuses réunions et soutenances pour lesquelles nous avons préparé, chose qui n'était pas facile vu que cette première année à l'Ecole Centrale Casablanca marque une période de transition vers un enseignement par projet, et le PLbD fut notre premier grand projet.

7 Perspectives

La mise en place du site web est une alternative que nous avions développée afin de pouvoir rendre fonctionnel et accessible notre solution aux agriculteurs dans le temps qui nous étais impartis. Cependant nous envisageons plusieurs perspectives d'amélioration qui contribuerons à rendre notre solution encore plus simple et accessible. En effet nous prévoyons mettre en place une application mobil qui pourra être directement téléchargeable sur Play store qui en plus des rubriques déjà disponibles dans notre site web contiendra d'autre rubrique à savoir la faite de pouvoir effectuer certaines analyses sans connexion. Également nous envisageons avec l'application mobile prendre en compte toutes les plantes cultivées au Maroc peu importe leur impact sur l'économie.

8 Webograhie

1. https://blog.engineering.publicissapient.fr/2017/03/01/tensorflow-deep-learning-episode-1-introduction/ article sur le Framework Tensorflow.
2. https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-que-keras/ Article précisant la définition de la bibliothèque ainsi que les différents Framework avec lequel il est compatible.
3. https://keras.io/api/applications/, site web de keras mettant à disposition les modèles disponibles sur keras ainsi qu'un guide pour l'utilisation de keras.
4. https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/ Explication des différents modèles que nous avions testée.

5. https://www.kaggle.com/bachchan1232313/plant-vision Base de données sur kaggle que nous avions eues à utilisé

6. Tutorial 28- Create CNN Model Using Transfer Learning using Vgg 16, Resnet Vidéo illustrant la mise en place d'un CNN pour le cas d'un modèle donné.
7. https://www.coursera.org/specializations/deep-learning?ranMID=40328&ranEAID=OyHlmBp2G0c&ranSiteID=OyHlmBp2G0c-jOGvREh3sHnogrd0_Q3_Bw&siteID=OyHlmBp2G0c-jOGvREh3sHnogrd0_Q3_Bw&utm_content=10&utm_medium=partners&utm_source=linkshare&utm_campaign=OyHlmBp2G0c#about Cours sur coursera sur le deep learning, que nous avions suivit conformément à la planification de la précédente soutenance.

8. https://www.kaggle.com/vipoooool/new-plant-diseases-dataset Base de données sur kaggle que nous avions eues à utiliser.
9. New Plant Diseases Dataset | Kaggle
10. Plant Pathology 2020 - FGVC7
11. Article vers l'ensemble des solutions existantes listées :
12. https://apps.apple.com/us/app/plantvillage-nuru/id1441395371

9 Annexes

Algorithme Inception V3 :

```
for root, dirs, files in os.walk(valid_path):
    for file in files:
        validation_images.append(os.path.join(root, file))


print("Training:")
print("Training Path:" + train_path)
print("Training Classes:" + str(len(os.listdir(train_path))))
print("Training Images:" + str(len(training_images)))

print("\n")

print("Validation:")
print("Validation Path:" + valid_path)
print("Validation Classes:" + str(len(os.listdir(valid_path))))
print("Validation Images:" + str(len(validation_images)))

print("\n")

print("Testing:")
print("Testing Path:" + test_path)
print("Testing Images:" + str(len(os.listdir(os.path.join(test_path, 'test')))))

train_datagen = ImageDataGenerator(
    rescale=1 / 255,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    shear_range=0.2,
    fill_mode='nearest'


training_images = []

for root, dirs, files in os.walk(train_path):
    for file in files:
        training_images.append(os.path.join(root, file))

validation_images = []
```

```
validation_datagen = ImageDataGenerator(rescale=1 / 255)

test_datagen = ImageDataGenerator(rescale=1 / 255)

batch_size = 32

train_generator = train_datagen.flow_from_directory(
    train_path,
    batch_size=batch_size,
    class_mode='categorical',
    target_size=(150, 150),
    color_mode="rgb",
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    valid_path,
    batch_size=batch_size,
    class_mode='categorical',
    target_size=(150, 150),
    color_mode="rgb",
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    test_path,
    batch_size=1,
    class_mode=None,
    target_size=(150, 150),
    color_mode="rgb",
```

```
    shuffle=False
)

class_dict = train_generator.class_indices
pprint(class_dict)

class_list = list(class_dict.keys())
pprint(class_list)

train_num = train_generator.samples
valid_num = validation_generator.samples

# Creating Instence of pre-trained model from Keras Application
inception_model = InceptionV3(input_shape=(150, 150, 3),
                              include_top=False,
                              weights=None)

inception_model.load_weights(local_weights_file)

# Freezing all Layers of Inception V3 Model
for layer in inception_model.layers:
    layer.trainable = False

inception_model.summary()

# Taking output from 'mixed8' layer
last_layer = inception_model.get_layer('mixed9')
print('Last Layer Output Shape:', last_layer.output_shape)
last_output = last_layer.output
```

```
123
124 # Adding our own layers at the end of Inception Network
125
126 # Flatten the output layer to 1 dimension
127 x = layers.Flatten()(last_output)
128
129 # Add a fully connected layer with 1024 hidden units and ReLU activation
130 x = layers.Dense(2048, activation='relu')(x)
131
132 # Add a fully connected layer with 1024 hidden units and ReLU activation
133 x = layers.Dense(1024, activation='relu')(x)
134
135 # Add a dropout rate of 0.2
136 x = layers.Dropout(0.2)(x)
137
138 # Add a final sigmoid layer for classification
139 x = layers.Dense(8, activation='softmax')(x)
140
141 model = Model(inception_model.input, x)
142
143 model.summary()
144
145 model.compile(optimizer=optimizers.RMSprop(lr=0.0001),
146               loss='categorical_crossentropy',
147               metrics=['accuracy'])
148
149 # # checkpoint
150 # weightpath = "best_weights.hdf5"
151 # checkpoint = ModelCheckpoint(weightpath, monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=True, mode='max')
```

```
154 history = model.fit(
155     train_generator,
156     steps_per_epoch=train_num // batch_size,
157     validation_data=validation_generator,
158     validation_steps=valid_num // batch_size,
159     epochs=30,
160     #     callbacks=callbacks_list,
161     verbose=1
162 )
163 # loss
164 print(history.history)
165 plt.plot(history.history['loss'], label='train loss')
166 plt.plot(history.history['val_loss'], label='val loss')
167 plt.legend()
168 plt.savefig('LossVal_loss_IV3')
169 plt.show()
170
171 # accuracies
172 plt.plot(history.history['accuracy'], label='train acc')
173 plt.plot(history.history['val_accuracy'], label='val acc')
174 plt.legend()
175 plt.savefig('AccVal_acc_IV3')
176 plt.show()
177
178 model.evaluate_generator(
179     generator=validation_generator,
180     steps=valid_num // batch_size
181 )
182
183 model.save('InceptionV2_classification2.h5')
```