



ÉCOLE
CENTRALELYON

ÉCOLE CENTRALE LYON

PAR 147 : RECONNAISSANCE DES PLANTES MARAÎCHÈRES
ET ANALYSE DE LEUR SANTÉ PAR RÉSEAUX DE NEURONES
PROFONDS

Rapport final de PAr

Elèves :
Yohan PELLERIN
Tristan CHEVREAU

Encadrant :
Charles-Edmond BICHOT
Conseillère :
Jacqueline VACHERAND-REVEL

26 mai 2023

1 Résumé

1.1 Français

Dans un contexte dans lequel de plus en plus de particuliers décident de faire leur propre potager, il est important que ceux-ci puissent avoir des outils pour mieux comprendre les besoins de leurs plantes. En particulier pour un néophyte, il est difficile de reconnaître les différentes carences que pourraient avoir leurs plantes maraîchères. Pour développer cet outil de détection de carences, nous avons choisi de comprendre quel était l'impact des carences courantes sur la forme, la taille et surtout la couleur de certaines parties de la plante. Il a fallu trouver des moyens pour séparer ces différentes parties. Nous avons ensuite quantifié ces impacts avec des scripts que nous avons appliqués à une base de données de photos de feuilles de plantes carencées. À partir des données récoltées, nous avons pu essayer de prédire, à travers différentes méthodes de classification, quelle était la carence des plantes présentes sur les images de test. Ces différentes méthodes ont donné des résultats variables, mais globalement concluants avec jusqu'à 90 % de réussite même pour les méthodes basiques (KPPV). Néanmoins, les méthodes plus avancées, même si plus consommatrices de puissance de calcul, donnent souvent des meilleurs résultats (SVM, descente du gradient). L'enjeu dans un cadre d'utilisation par des particuliers sera alors de choisir le meilleur compromis entre calculs et résultat qu'il reste encore à définir. Malgré tout, discriminer les carences en Potassium et en Phosphore reste une tâche difficile, qu'importe la méthode de classification, et il serait bénéfique d'exploiter de nouvelles manières de quantifier l'impact de ces carences, par exemple sur d'autres organes de la plante.

1.2 Anglais

In a context where more and more individuals are deciding to grow their own vegetable gardens, it is important that they have tools available to better understand the needs of their plants. For someone without knowledge, it is especially difficult to recognize the different deficiencies that their vegetable plants may have. To set up this deficiency detection tool, we chose to understand the impact of common deficiencies on the shape, size, and mainly the color of certain parts of the plant. We had to find ways to separate these different parts. We then quantified these impacts with scripts that we applied to a database of photos of deficient plant leaves. From the data collected, we were able to predict, through different classification methods, what the deficiency of the plant was for a certain number of test images. These different methods gave varying results, but generally conclusive with up to 90 % success even for basic methods (KPPV). Nevertheless, more advanced methods, even if more computationally intensive, often gave better results (SVM, gradient descent). The challenge in a context of use by individuals will then be to choose the best compromise between calculations and results (that remains to be defined). However, discriminating between Potassium and Phosphorus deficiencies remains a difficult task, regardless of the classification method. It would also be beneficial to explore new ways to quantify the impact of deficiencies for example on other parts of the plant.

Table des matières

1 Résumé	1
1.1 Français	1
1.2 Anglais	1
2 Introduction et contexte	5
3 Glossaire	6
4 Réduction du champ de l'étude	7
5 Démarche du PAr	8
6 Pilotage et gestion du projet	9
7 État de l'art sur l'apport de l'IA aux diagnostics de plantes	10
7.1 La recherche sur les diagnostics de maladies de plantes en général	10
7.2 L'impact des carences sur les plantes	10
7.3 La recherche sur les diagnostics de carences de plantes	11
8 Solutions choisies	13
8.1 L'approche descriptive et sa justification	13
8.2 Les différents datasets trouvés : choix d'une base de données	13
8.2.1 <i>PlantVillage</i>	14
8.2.2 La base de donnée 'faite maison' de l'okra	14
8.2.3 "Nitrogen deficiency for rice crop" sur <i>kaggle</i>	15
9 Travail effectué	16
9.1 Extraction et traitement des descripteurs	16
9.1.1 Séparation des différentes parties de la feuille	16
9.1.2 Analyse colorimétrique et sauvegarde des données	21
9.1.3 Normalisation des descripteurs	22
9.2 Classification	22
9.2.1 K plus proches voisins	22
9.2.2 Descente du gradient	25
9.2.3 Réseau de neurones	28
9.2.4 Machines à vecteurs de support (SVM)	34
9.2.5 Analyse et interprétation des résultats : comparaison des performances des algorithmes	38
10 Limites et prolongement de l'étude	40
10.1 Discriminer Potassium et Phosphore	40
10.2 Ajouter des classes et intégrer d'autres bases de données	40
11 Conclusion	41
12 Bibliographie	42

Table des figures

1	Organigramme de tâches	9
2	Diagramme de Gantt	9
3	Tableau récapitulatif	11
4	Carences en Azote du figuier (d'après [1])	11
5	Extrait du dataset <i>PlantVillage</i> montrant une feuille de pomme de terre atteinte de « brûlure alternarienne »	14
6	Extrait de la publication, montrant une des photos de leur dataset fait maison : okra en bonne santé	15
7	Extrait de la base de donnée : une feuille de riz carencée en phosphore . . .	15
8	Utilisation d'un masque en traitement d'images (d'après ResearchGate) . .	16
9	Les différentes étapes de la méthode	18
10	Un exemple d'image mal traitée par <code>remove.bg</code>	18
11	Un exemple d'image bien traitée par <code>remove.bg</code>	19
12	Un exemple de masque obtenu pour les contours de la feuille	19
13	Application du masque à l'image sans arrière-plan	20
14	Les différentes étapes de la méthode pour les nervures	20
15	Exemple de sortie de la fonction <code>analyse_color</code>	21
16	Représentation visuelle d'une classification KPPV (d'après <i>Wikipedia</i>) . .	23
17	Résultats de la classification KPPV non pondérée	23
18	Résultats de la classification KPPV pondérée	23
19	Résultats de la classification KPPV non pondérée (sans les nervures) . .	24
20	Résultats de la classification KPPV pondérée (sans les nervures)	24
21	Matrice de confusion pour k=1	24
22	Matrice de confusion pour k=1 (sans les nervures)	24
23	Représentation visuelle de la descente du gradient en 2D et 3D	25
24	Tableau de recherche des hyper-paramètres optimaux avec les nervures .	26
25	Tableau de recherche des hyper-paramètres optimaux sans les nervures .	26
26	Matrice de confusion avec les nervures	27
27	Matrice de confusion sans les nervures	27
28	Exemple de réseau de neurones	28
29	Fonction ReLU	29
30	taux de classification sur les tests avec alpha = 0.001	30
31	taux de classification sur les données d'apprentissage avec alpha = 0.001 .	30
32	taux de classification sur les tests avec alpha = 0.0001	30
33	taux de classification sur les données d'apprentissage avec alpha = 0.0001 .	30
34	taux de classification sur les tests avec alpha = 10^{-5}	30
35	taux de classification sur les données d'apprentissage avec alpha = 10^{-5} .	30
36	taux de classification sur les tests avec alpha = 10^{-6}	31
37	taux de classification sur les données d'apprentissage avec alpha = 10^{-6} .	31
38	taux de classification sur les tests avec alpha = 6.10^{-6}	31
39	taux de classification sur les données d'apprentissage avec alpha = 6.10^{-6} .	31
40	taux de classification sur les tests avec alpha = 8.10^{-5}	31
41	taux de classification sur les données d'apprentissage avec alpha = 8.10^{-5} .	31
42	taux de classification sur les tests avec alpha = 10.10^{-5}	32
43	taux de classification sur les données d'apprentissage avec alpha = 10.10^{-5} .	32

44	taux de classification sur les tests avec alpha = 12.10^{-5}	32
45	taux de classification sur les données d'apprentissage avec alpha = 12.10^{-5} .	32
46	taux de classification sur les tests avec alpha = 14.10^{-5}	32
47	taux de classification sur les données d'apprentissage avec alpha = 20.10^{-5} .	32
48	taux de classification sur les tests avec alpha = 20.10^{-5}	32
49	taux de classification sur les données d'apprentissage avec alpha = 20.10^{-5} .	32
50	Matrice de confusion	33
51	Représentation visuelle d'une application de fonction noyau (d'après <i>wikipedia</i>)	34
52	Comparaison des différents noyaux par défaut (avec et sans les nervures) .	35
53	Matrices de confusion pour chacun des noyaux, avec et sans les nervures .	36
54	Recherche par maillage : amélioration des résultats obtenus avec le noyau rbf	37
55	Comparaison des différentes méthodes	38

2 Introduction et contexte

"Les problématiques écologiques actuelles nous poussent à chercher de nouveaux modèles de consommation et à réfléchir à un plan de transition vers ceux-ci. Le maraîchage a été industrialisé de manière massive depuis le XXème siècle et nos modes de vie ont progressivement abandonné la culture potagère. Cependant, la réalisation de potagers dans les zones d'habitation qui le permettent peut contribuer à une meilleure alimentation, aux circuits courts, au recyclage des déchets, aux activités physiques ainsi qu'au développement personnel." Extrait de la fiche de présentation du PAr

Malheureusement, les connaissances agricoles disparaissent. Les personnes non-initierées n'ont plus les connaissances nécessaires pour diagnostiquer une maladie de plante, d'autant que ces maladies revêtent différentes formes selon le type de la plante. Ces maladies comprennent : les carences, les parasites, les virus et les bactéries. Cependant, les problématiques écologiques actuelles poussent les particuliers à faire leur propre culture potagère.

Pour ces particuliers, il y a alors un décalage entre ce besoin d'opérer des diagnostics et leurs propres capacités. Ainsi, ils pourraient avoir besoin d'un système d'aide au diagnostic de leurs plantes, ce qui leur permettrait de prendre des mesures et des précautions adaptées (engrais, phytosanitaires, lutte biologique, etc.).

Ce système d'aide au diagnostic doit ainsi être efficace (c'est-à-dire simple et rapide), pour pouvoir imaginer le rendre transportable (application mobile par exemple).

Le sujet nous invite à identifier les logiciels ou autres outils d'aide au diagnostic déjà existants, d'identifier une voie d'amélioration et de concevoir ce système en utilisant de l'intelligence artificielle.

3 Glossaire

Les mots prérequis pour la bonne compréhension de ce rapport et que nous avons identifiés comme non-courants ou ambigus sont définis ici. Ils sont pris dans le contexte de notre PAr. Certains termes sont redéfinis dans le rapport.

- **Dataset** : Dans notre rapport et comme c'est souvent le cas dans les articles traitant du machine learning que nous avons pu lire, *Dataset* désignera une base de données organisée par classes.
- **Classe** : La classe est le groupe auquel une donnée appartient. Par exemple, dans une base de données contenant des images de fruits organisées par espèces, la classe d'une image pourrait être "pomme" ou "ananas". Un *Dataset* contient un nombre fini de classes bien identifiées.
- **Nervures** : Les nervures sont les "veines" d'une feuille, qui forment un réseau.
- **Carence** : Une carence est un type de maladie de plante qui est dû à un manque de disponibilité d'un élément atomique (Phosphore, Azote et Potassium sont les plus connus.). La plante ne peut donc pas synthétiser les molécules nécessaires à sa bonne santé (croissance, floraison, hormones ...). Par exemple, l'azote sert à la synthèse de protéines et le phosphore à la synthèse de l'ADN.

4 Réduction du champ de l'étude

Après discussions avec nos tuteurs de PAr à l'issue du RVP1, il a été décidé de se concentrer sur une maladie des plantes en particulier : les carences.

Une carence est un type de maladie de plante qui est dû à un manque de disponibilité d'un élément atomique (Phosphore, Azote et Potassium sont les plus connus.). La plante ne peut donc pas synthétiser les molécules nécessaires à sa bonne santé (croissance, floraison, hormones ...) [2]. Par exemple, l'azote sert à la synthèse de protéines et le phosphore à la synthèse de l'ADN.

En effet, l'aide au diagnostic de carences de plantes paraît d'après nos recherches (voir la partie sur l'état de l'art) être un sujet nettement moins étudié que les maladies liées à des parasites ou autres.

Cela s'explique peut-être en partie par le prix faible des engrains chimiques et biologiques qui ne poussent pas forcément les acteurs principaux de l'agro-alimentaire à investir dans l'optimisation de leur utilisation, à l'inverse des produits phytosanitaires.

5 Démarche du PAr

L'objectif global est d'étudier les différentes techniques d'aide au diagnostic déjà existantes, d'identifier une voie d'amélioration et de la concevoir.

Pour cela, nous avons réalisé les tâches suivantes :

- Approfondir / Faire un état de l'art du sujet :
 - Sur les techniques actuelles de diagnostic et de reconnaissance des plantes.
 - Sur les impacts qu'ont les différentes maladies sur l'état des plantes, et en particulier les carences.
 - Sur les applications et les outils disponibles exécutant déjà une partie de nos objectifs de PAr pour comparaison¹.
- Repérer, dans cet ensemble de renseignements précédents et à l'aide de la bibliographie générée, un point peu ou pas exploité par les logiciels et méthodes déjà existants.
- De mettre en place une ou plusieurs techniques utilisant ce point peu exploité.
- De comparer différentes manières d'exploiter cette ou ces techniques
- De réaliser des livrables finaux faisant office de documentation de notre travail.

1. Après avoir lu plusieurs éléments bibliographiques, nous avons constaté que certains logiciels comme *PlantNet* ou *picturethisai* réalisent déjà de l'aide au diagnostic.

6 Pilotage et gestion du projet

Nous avons d'abord divisé la liste précédente en sous-tâches, on obtient alors l'organigramme suivant :

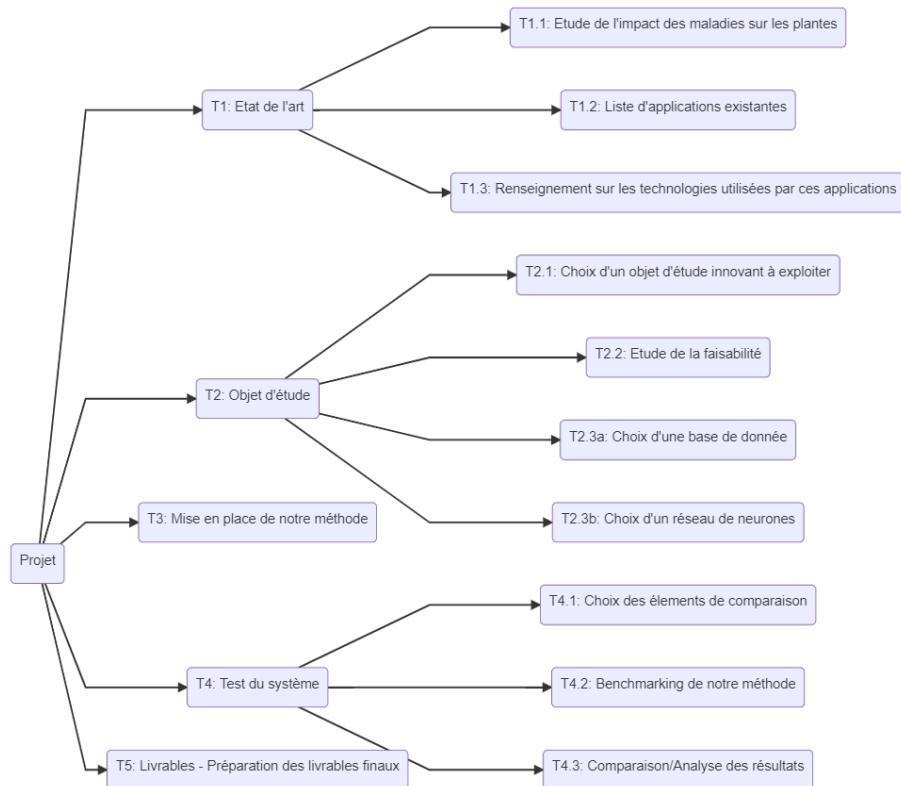


FIGURE 1 – Organigramme de tâches

On a aussi réalisé au début de l'année un diagramme de Gantt pour ces mêmes tâches :

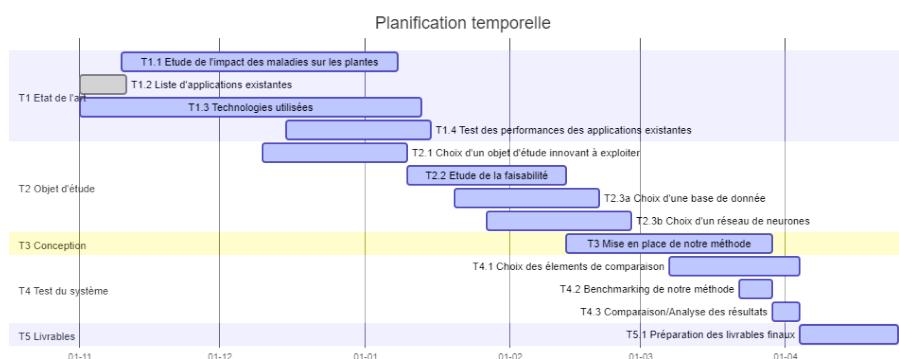


FIGURE 2 – Diagramme de Gantt

7 État de l'art sur l'apport de l'IA aux diagnostics de plantes

Pour comprendre les passages techniques des différents éléments bibliographiques, nous avons utilisé la thèse fournie par notre tuteur : [3].

7.1 La recherche sur les diagnostics de maladies de plantes en général

Les documents bibliographiques et outils de diagnostic des maladies de plantes sont nombreux. Certains de ces outils arrivent à des taux de classification supérieurs à 99% avec une approche *deep learning* (*EfficientNet*) et décrivent avec précision des méthodes d'évaluation de la performance [4]. D'autres obtiennent environ 90 % en utilisant des SVM [5].

Mais le problème de ces études, c'est qu'elles sont souvent basées sur la même approche globale, que les performances ne sont testées que superficiellement avec souvent la même base de données : *PlantVillage*. En effet, le travail nécessaire à la création d'une nouvelle base de données est le principal facteur limitant. De plus, les tests de cas réels sont rares : le plus souvent, les performances des classifications sont évaluées à partir d'une seule base de données : la même que celle utilisée pour l'entraînement. De plus, ces outils sont rarement maintenus correctement et très vite laissés à l'abandon [6].

7.2 L'impact des carences sur les plantes

Pour mieux comprendre comment construire nos descripteurs, nous avons recherché l'impact des carences sur les plantes. Chaque carence a des symptômes spécifiques qui lui sont associés et qui se retrouvent sur toutes les plantes [2]. Néanmoins, il peut être utile de voir les symptômes des carences pour une plante particulière comme le figuier (figure 4), pour comprendre que certains de ces symptômes sont aussi spécifiques à une plante [1]. Nous avons pu établir des listes de symptômes propres à chaque carence, et surtout celles qui sont les plus communes (Azote, Potassium, Phosphore, Fer, Calcium et Magnésium en particulier). On retrouve bien sur l'image 4 les effets observés de la première ligne du tableau 3. À partir des effets observés lors des carences, on a identifié des valeurs à quantifier. C'est-à-dire les paramètres à mesurer pour pouvoir déterminer l'appartenance d'une plante à une catégorie.

type de carence	Effet observé	Valeurs à quantifier
N	<ul style="list-style-type: none"> - Rend les plantes chlorotiques (jaunes), observation de chlorosis - Rend la maturité précoce - Les feuilles les plus vieilles jaunissent, en commençant par le bout de ses feuilles et par les parties les plus éloignées des nervures de la feuille 	<ul style="list-style-type: none"> - intensité moyenne de jaune - intensité de jaune sur l'extrémité - intensité de jaune sur les nervures
P	<ul style="list-style-type: none"> - Rend les feuilles vert foncé jusqu'à violet - Les feuilles sont tombantes - Les tâches peuvent être assez mobiles 	<ul style="list-style-type: none"> - intensité de vert/violet sur la feuille - angle d'inclinaison par rapport à la tige
K	<ul style="list-style-type: none"> - Les feuilles les plus vieilles jaunissent, en commençant par le bout de ses feuilles et en particulier par les parties les plus au bord - Ensuite les parties les plus jaunes nécrosent (brunissent) 	<ul style="list-style-type: none"> - intensité de brun et du jaune (marron) du bord des feuilles
Ca	<ul style="list-style-type: none"> - Les premiers symptômes apparaissent sur les feuilles les plus jeunes - Les feuilles se tordent et se replient sur elles-mêmes - Les bourgeons, fleurs et feuilles tombent prématurément 	<ul style="list-style-type: none"> - longueur de la feuille - courpore de la plante (à préciser) - forme de la plante
Mg	<ul style="list-style-type: none"> - Entre les nervures des feuilles, la couleur change (de manière chlorotique, mais cela peut aussi rougir ou bronzer). - Sur les nervures, la couleur est inchangée. - Touche particulièrement les feuilles vieilles - Peut se confondre avec une carence en K avancée 	<ul style="list-style-type: none"> - couleur (intensité de jaune, de brun, de rouge) entre les nervures - couleur des nervures
Fe	<ul style="list-style-type: none"> - Entre les nervures des feuilles, la feuille jaunit. - Sur les nervures, la couleur est inchangée. - Les feuilles deviennent presque blanches, surtout pour les jeunes feuilles 	<ul style="list-style-type: none"> - couleur (intensité de jaune, de brun, de rouge) entre les nervures - couleur des nervures - intensité moyenne de blanc

FIGURE 3 – Tableau récapitulatif

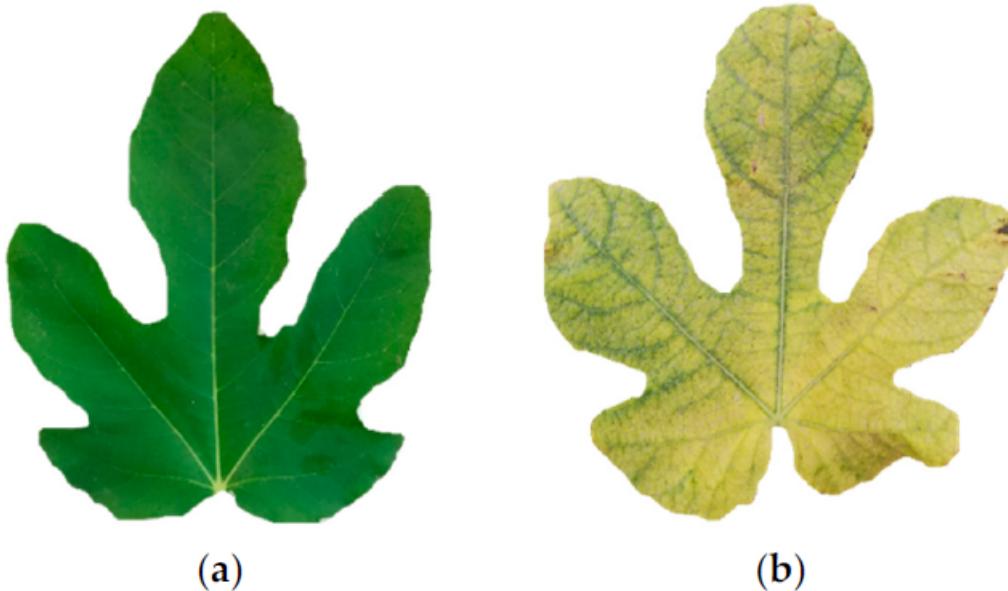


FIGURE 4 – Carences en Azote du figuier (d'après [1])

7.3 La recherche sur les diagnostics de carences de plantes

La plupart des études sur le sujet n'étudient qu'une seule plante à la fois, comme le piment [7], l'okra [8] ou le haricot urd [9]. Les bases de données sont fabriquées spécifiquement pour l'étude². L'approche est toujours celle de réseaux de neurones type *deep*

2. Nous n'avons reçu aucune réponse à nos nombreuses sollicitations pour recevoir ces bases de données.

learning. Les résultats sont légèrement variables, mais restent autour de 85 % de classification réussie (86 % pour l'okra, 83 %). Nous n'avons pas trouvé d'étude cherchant à mener une analyse descriptive de ces carences.

Le facteur limitant est encore une fois la base de données. C'est pourquoi certaines études se contentent simplement de préciser une méthode à utiliser (aussi bien pour la classification que la fabrication de la base de données) sans pouvoir la mettre en application [10].

Pour contourner ce problème de la base de données, certaines études utilisent des données physico-chimiques provenant de serres industrielles au lieu de photographies, mais cela demande une compréhension extensive des relations chimiques entre la plante et l'environnement [11] que nous n'avons pas.

Pour revenir au contexte de notre PAr, on remarquera que les plantes maraîchères européennes ne sont pas représentées dans les précédentes études.

8 Solutions choisies

8.1 L'approche descriptive et sa justification

Les recherches précédentes ont permis de mettre en avant les symptômes liés aux carences. Ainsi, sur les plantes, une carence peut modifier trois aspects de la plante : la couleur, la texture (réseau de nervures) et la forme (7.1). Cependant, la plupart des documents bibliographiques consultés ne prenaient pas en compte cet aspect-là, et se contentaient de donner une image complète au réseau de neurones et d'effectuer de nombreuses opérations mathématiques sur les pixels de l'image. Or, l'état de l'art précédent, a relevé que les carences avaient des conséquences directes et reconnaissables sur la couleur, la texture et la forme des différentes plantes. Notre idée est ainsi de se focaliser sur ces trois informations et de les extraire "à la main".

Pour identifier ces trois informations, nous allons utiliser des descripteurs. Les descripteurs sont des fonctions qui prennent en entrée l'image, et qui rendent une valeur, par exemple la couleur moyenne ou bien la luminosité. Ensuite, nous allons réaliser une classification à partir de ces descripteurs. Dans cette méthode, les entrées sont connues et imposées. La classification est donc uniquement faite sur des critères fixés par des humains.

À l'inverse, dans l'approche *deep learning* du corpus bibliographique, l'entrée est une image complète et ainsi les critères choisis sont nombreux et ne sont pas connus (c'est en général un ensemble de segmentations et de produits de convolution sur les pixels de l'image qui ne seraient pas vraiment interprétables par un humain, à l'inverse de notre approche). Au vu des résultats produits, cette approche *deep learning* fonctionne.

Cependant, on peut penser qu'ils ne sont pas tous pertinents et que la méthode est chronophage et énergivore. Ainsi, en minimisant les entrées, on choisit des critères que l'on sait pertinents.

Néanmoins, il est aussi possible que l'effet inverse soit observé. Il est possible que certains critères non identifiés lors de l'étape de bibliographie aient un impact. Ces critères peuvent cependant être pris en compte lorsque toute l'image est donnée au réseau de neurones. Ainsi, ne pas les prendre en compte conduirait à une classification moins performante.

8.2 Les différents datasets trouvés : choix d'une base de données

Pour pouvoir mettre en application la nouvelle méthode que nous proposons, il va nous falloir une base de données à exploiter. Ces bases de données où les images sont classées par type sont appelées datasets. Comme énoncé précédemment, c'est bien le dataset l'élément le plus limitant pour l'avancement dans les études actuelles. Nous avons besoin d'un dataset respectant au moins les éléments suivants :

- 1) Contenir une grande quantité d'images pour chaque classe (type) de carence
- 2) Être disponible et accessible (mail ou téléchargement direct sur internet)
- 3) Contenir une grande partie de la feuille pour exploiter les différences de couleur, forme et texture du début à la fin de la feuille

— 4) Contenir une ou plusieurs plantes maraîchères

Nous avons pourtant réalisé de nombreuses recherches et contacté différentes personnes pour obtenir les bases de données provenant des documents bibliographiques ; malheureusement, nous n'avons eu aucune réponse.

Parmi les datasets que nous avons trouvés, nous en présenterons trois : *PlantVillage*, La base de donnée 'faite maison' de l'okra [8], et "Nitrogen deficiency for rice crop" sur *kaggle*.

8.2.1 *PlantVillage*

La base de données *PlantVillage* est une référence. Elle est largement, et même trop largement, utilisée [6]. En effet, elle contient des photos carrées et cadrées de feuilles de plusieurs plantes maraîchères courantes en Europe (pomme, raisin, tomate, cerise, etc.) en résolution 256×256 pixels (figure 5). Pour chaque plante, il y a un ensemble d'images de plante saines et au moins un autre ensemble d'images de plante malade. Pour chaque ensemble, il y a environ 1000 photos, ce qui est largement suffisant.

Néanmoins, cette base de données ne convient pas pour l'implémentation de notre méthode, car ces maladies ne sont pas des carences, mais plutôt des infections comme le mildiou.



FIGURE 5 – Extrait du dataset *PlantVillage* montrant une feuille de pomme de terre atteinte de « brûlure alternarienne »

8.2.2 La base de donnée 'faite maison' de l'okra

Ce dataset concerne une plante médicinale appelée okra, utilisée dans l'étude [8]. Ce dataset contiendrait 231 images de 299×299 pixels d'okra non carencé ou carencé en Calcium, Potassium, Azote ou Phosphore (figure 6).

Ce dataset ne convient absolument pas, car c'est trop peu d'images pour 5 classes, et cela ne correspond pas à notre contexte (utilisateur européen dans son propre jardin) car c'est une plante qui n'est ni maraîchère ni européenne. De plus, nous n'avons pas réussi à l'obtenir en intégralité pour le moment.



FIGURE 6 – Extrait de la publication, montrant une des photos de leur dataset fait maison : okra en bonne santé

8.2.3 "Nitrogen deficiency for rice crop" sur *kaggle*

Ce dernier dataset contient des photos très recadrées de feuilles de riz. Il y a plus de 300 photos pour chacune des trois carences les plus connues : Phosphore, Potassium et Azote (figure 7). Malheureusement, il n'y a pas de classe "témoin" (riz non carencé). Ce n'est pas non plus la plante entière, ce qui limite l'étude possible sur la forme.

Même si le riz n'est pas caractéristique des cultures potagères européennes, il reste néanmoins cultivé en Camargue (30 % de la consommation française!³). De plus, il intéresse une partie des personnes qui font leur propre potager, comme le montrent les nombreux tutoriels disponibles sur *Youtube*.

Ensuite, ce dataset est facilement accessible, bien structuré et pré-traité et concerne une plante comestible. Il peut constituer une bonne base pour tester notre méthode, car de toute façon, nous étudions, dans le cadre des carences, surtout les feuilles. Nous avons donc décidé d'utiliser ce dataset, qui semble correspondre le mieux à nos attentes en comparaison avec les autres datasets.



FIGURE 7 – Extrait de la base de donnée : une feuille de riz carencée en phosphore

3. Communiqué de presse SEMAE du 4 mars 2022 : "LE RIZ : UNE EXCEPTION FRANÇAISE"

9 Travail effectué

9.1 Extraction et traitement des descripteurs

9.1.1 Séparation des différentes parties de la feuille

Pour rappel, la figure 7 montre un exemple d'image de feuille de riz que nous allons exploiter.

Pour exploiter les différences d'impact qu'ont les diverses carences sur chacune des parties de la plante, et la feuille en particulier, il a fallu séparer les images de feuilles en plusieurs images ne comprenant que les parties qui nous intéressent.

Pour cela, nous allons utiliser les masques. Un masque est une image de même dimension que l'image originale, sauf que chaque pixel d'un masque contient soit 0, soit 1. Par convention, on représente le 0 en noir et le 1 en blanc. Lorsqu'on applique un masque à une image, on ne garde que les pixels de l'image originale correspondant aux pixels blancs du masque. Un exemple d'application est présenté sur la figure 8. Nous ne parlons pas ici de noyau⁴, même si ceux-ci sont parfois appelés masques.

L'objectif, pour chaque partie de la feuille, est donc d'obtenir un masque pour chaque image de la base de données qui permettrait de ne conserver que la partie considérée.

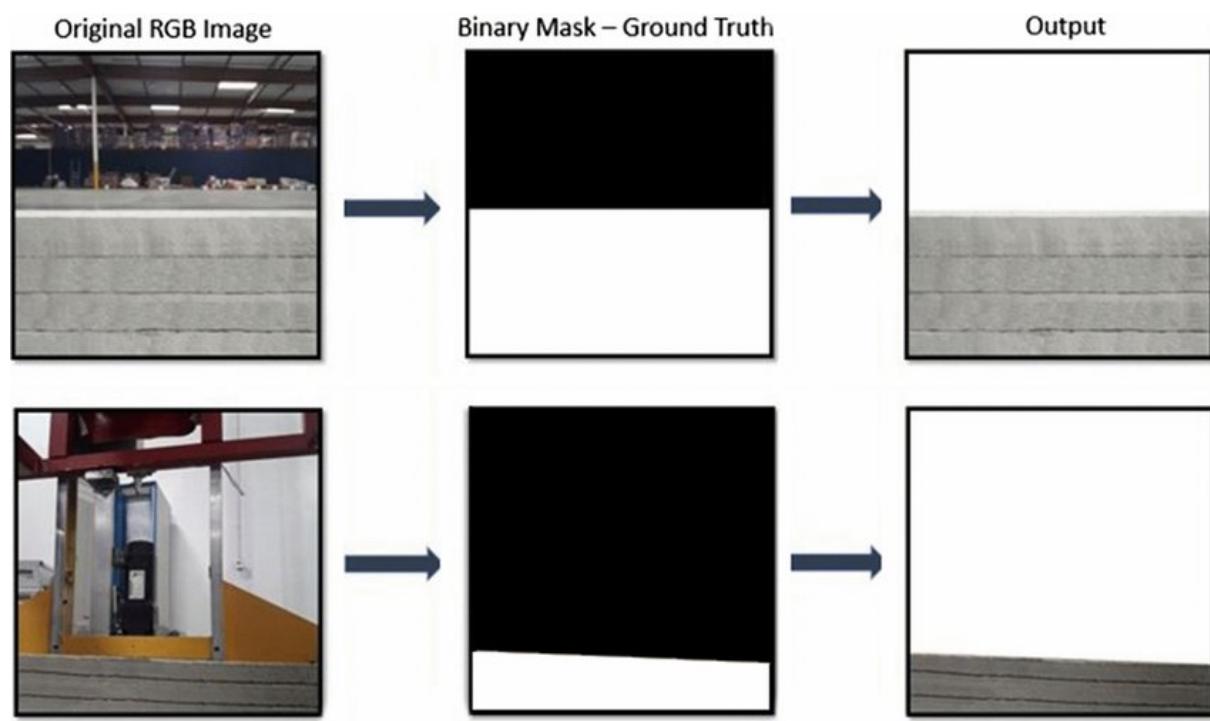


FIGURE 8 – Utilisation d'un masque en traitement d'images (d'après ResearchGate)

1.a) Enlever l'arrière-plan, méthode 1

4. [https://fr.wikipedia.org/wiki/Noyau_\(traitement_d%27image\)](https://fr.wikipedia.org/wiki/Noyau_(traitement_d%27image))

La première étape pour pouvoir traiter chaque partie de la feuille séparément est d'enlever l'arrière-plan des images.

Dans un premier temps, nous avons implémenté notre propre script d'extraction d'images de feuilles d'après une amélioration du script accessible sur un tutoriel⁵ de la documentation officielle de *plantcv* [12].

Néanmoins, ce script se base notamment sur l'uniformité de la couleur de la feuille. Dans le cas notre base de données, certaines feuilles sont parfois lourdement carencées et donc pas du tout uniformes.

Celle-ci fonctionne ainsi (la figure 9 présente l'image à chaque étape de la transformation correspondante) :

1. Conversion de l'image en niveaux de gris.
2. On applique un seuil pixel-par-pixel : on ne conserve que les pixels d'une certaine luminance. Les valeurs ne sont que de 0 : "pas conservé" ou 1 : "conservé".
3. En parallèle, on fait une détection de contours de Canny sur l'image originale.
4. On soustrait l'image 2) à l'image 3) pour séparer la feuille des éléments de l'arrière-plan qui étaient de la même couleur.
5. La feuille est alors le plus gros groupe de pixels blancs de l'image : on supprime les groupes plus petits et on remplit les trous laissés par la soustraction.
6. On finit par un flou gaussien pour finir de boucher les pixels seuls et lisser le rendu du masque.

5. <https://plantcv.readthedocs.io/en/stable/tutorials/visTutorial/>

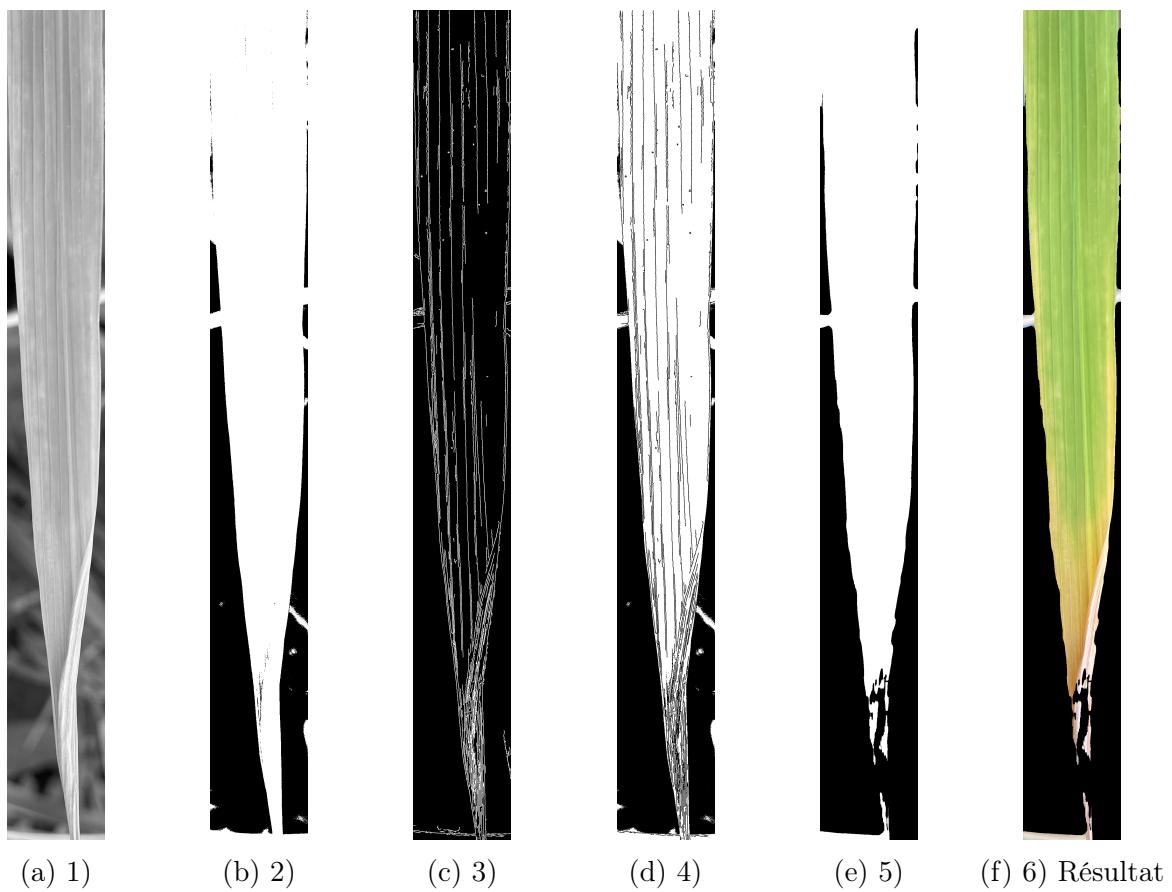


FIGURE 9 – Les différentes étapes de la méthode

Malgré quelques semaines d'essais, nous avons abandonné cette technique. En effet, on voit que les résultats ne sont pas très probants même dans un très bon cas. De plus, le traitement était très long.

1.B) Enlever l'arrière-plan, méthode 2

Après quelques recherches, nous avons trouvé une API (un moyen d'interfacer notre code et un logiciel / programme extérieur) pour python du site web `remove.bg` qui permet de faire cela de manière automatique. Le gros défaut de cette méthode est qu'il est impossible d'essayer des réglages différents. Lorsque cela ne marche pas, on ne peut rien faire de plus.

Pour la plupart des images, les résultats sont corrects. Malgré tout, il y a parfois des ratés (figure 10), en particulier lorsque la couleur de l'arrière-plan est très proche de la couleur de la feuille et lorsque la photo n'est pas très nette.



FIGURE 10 – Un exemple d'image mal traitée par `remove.bg`

Nous avons considéré que ces ratés étaient négligeables. En effet, nous n'avons pas voulu perdre de temps à vérifier chaque image une à une, d'autant qu'à part en le faisant à la main, il n'y avait pas la possibilité de corriger ces erreurs. Ce traitement étant opéré sur des serveurs tiers au travers de l'API, il est très long.

Les images rendues par l'API ne peuvent pas être utilisées telles quelles dans notre cas. En effet, l'outil étant destiné à l'utilisation pour des montages photo en particulier, il ajoute une information de transparence à l'image, appelé "canal alpha" qui permet un rendu plus lisse lorsqu'on superpose plusieurs images. L'image rendue par l'API est donc de la forme (R, V, B, A) et non pas (R, V, B). Nous n'avons pas besoin de cette information, nous avons donc opéré un seuillage comme dans le paragraphe précédent. Lorsque la valeur de A pour un pixel était supérieure à un certain seuil (choisi manuellement), on conserve ce pixel. Sinon, on remplace par du noir.

Un exemple de bon résultat est présenté sur la figure 11.

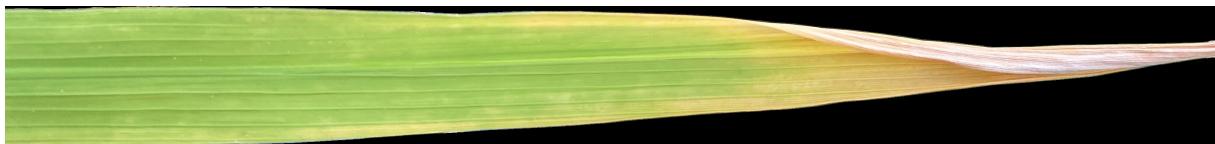


FIGURE 11 – Un exemple d'image bien traitée par `remove.bg`

2) Extraire les contours de la feuille

Pour extraire les contours de la feuille, nous avons utilisé la détection de contours de Canny disponible dans la bibliothèque *PlantCV*, appliquée aux images dont le fond avait été enlevé. Comme la feuille est rarement noire, et en tout cas jamais parfaitement noire ($R=0, G=0, B=0$), la sensibilité de la détection de contours n'avait pas à être très élevée. Nous nous sommes fixés une certaine largeur en pixels pour le contour. Un exemple de masque obtenu est présenté figure 12.

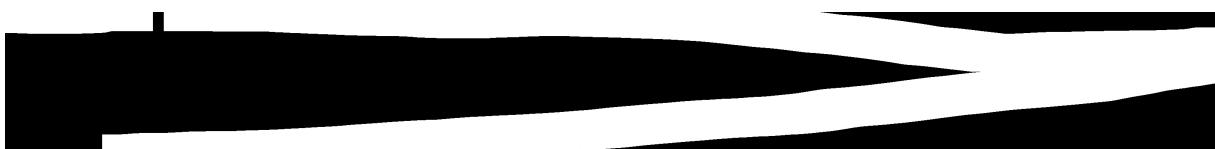


FIGURE 12 – Un exemple de masque obtenu pour les contours de la feuille

L'application de ce masque à l'image sans arrière-plan est présenté figure 13. On voit sur cet exemple que lorsque le bord de la feuille dépasse le bord de l'image, cela crée des "trous" dans le contour. Nous aurions pu, dans ce cas, modifier l'algorithme pour qu'il prenne tout de même le bord de l'image ; mais nous avons remarqué que dans certains cas, cela donnait un résultat moins représentatif du bord de la feuille, car celui-ci dépassait trop de l'image.

En effet, même s'il manquait une partie du contour, cela avait peu d'impact, car cela restait représentatif. Rajouter dans ce masque quelque chose qui n'était pas le contour de la feuille était d'après nous plus dommageable.



FIGURE 13 – Application du masque à l'image sans arrière-plan

3) Extraire les nervures de la feuille

Pour extraire les nervures de la feuille, nous avons imaginé le traitement suivant :

1. On détecte les contours comme pour le paragraphe précédent, avec une épaisseur moins importante.
2. On détecte les variations de couleur de l'image encore une fois avec un filtre de Canny d'une sensibilité plus élevée.
3. On soustrait l'étape 2) à l'étape 1) pour se débarrasser des contours de la feuille et ne garder que les variations dans la feuille.



FIGURE 14 – Les différentes étapes de la méthode pour les nervures

On voit (figure 14) que les résultats de cette méthode sont assez corrects. Malgré tout, l'extrémité de la feuille est surreprésentée par ce masque.

9.1.2 Analyse colorimétrique et sauvegarde des données

Une fois tous les masques réalisés, il faut réaliser une analyse colorimétrique des différents morceaux de la feuille. Pour cela, on utilise le code couleur RGB. Il aurait été possible d'utiliser d'autres codes couleur (HSV, YCbCr), cependant celui-là est le plus connu, de plus ils contiennent tous la même information.

Pour réaliser cette analyse colorimétrique, nous avons utilisé la fonction "analyze_color" de la bibliothèque *PlantCV*. Celle-ci permet de donner pour chacune des trois couleurs primaires la proportion de pixels avec une intensité de 0 à 255.

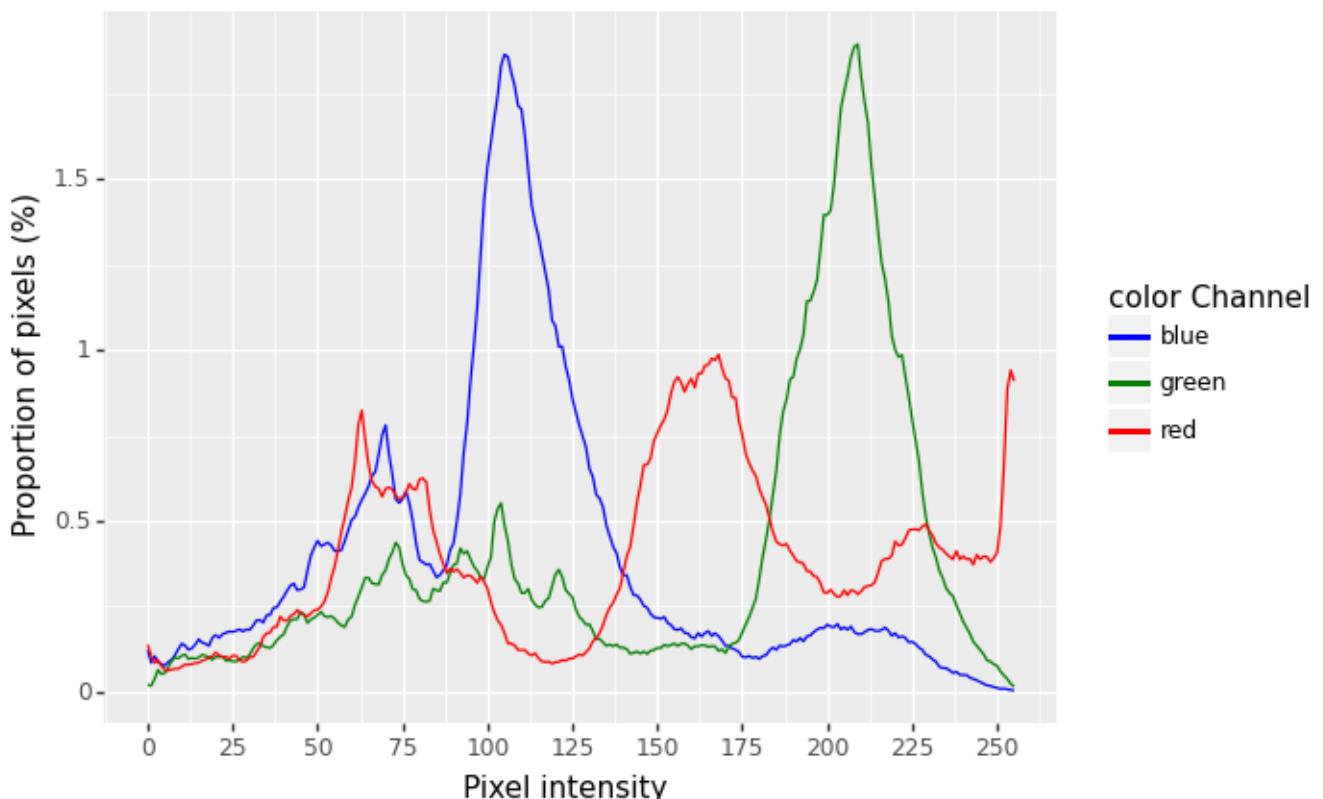


FIGURE 15 – Exemple de sortie de la fonction analyse_color

Cependant, cette fonction permet uniquement d'afficher le graphe 15. Ainsi, pour avoir accès aux valeurs contenues dans le graphe, nous avons dû modifier le code initial de *PlantCV*. Comme celle-ci était disponible entièrement, nous avons simplement modifié l'affichage pour obtenir un tableau de valeurs pour chaque couleur contenant la proportion de pixel à intensité 0, 1 etc jusqu'à 255. D'autre part, comme on applique cela à des images avec des masques, il ne faut pas compter toutes les cases complètement noires ($R=0$, $V=0$, $B=0$) qui faussent les proportions. Pour cela, il suffit de compter les cases noires et d'ajuster les proportions pour ne plus les prendre en compte.

On a ainsi pu appliquer cela à chaque image et pour chaque morceau de feuille. On a regroupé les tableaux par classes et par type de morceau. On a ainsi obtenu 9 tableaux différents. En effet, il y a 3 parties de la feuille différentes (contours, feuille entière et nervures) et 3 classes (Azote, Phosphore, Potassium). Chaque tableau compte autant de

lignes de données qu'il y a d'images, et contient trois fois 256 colonnes (256 colonnes pour chaque couleur : Rouge, Vert et Bleu). Ce sont ces tableaux qui joueront le rôle de descripteurs pour la suite du rapport.

9.1.3 Normalisation des descripteurs

Une fois les tableaux réalisés, il faut passer à la classification ; Cependant, avant cela, il faut réaliser une normalisation des descripteurs. Ceci consiste à supprimer la valeur moyenne du descripteur et diviser par l'écart-type ($x_i = \frac{x_i - x_{moy}}{\sigma(x)}$). Ainsi tous les descripteurs ont le même écart type et la même variance. Chaque descripteur a donc initialement le même poids pour la classification.

9.2 Classification

Nous avons des vecteurs de descripteurs exploitables pour chaque image de la base de données. Il faut ensuite les utiliser pour classifier, c'est-à-dire prédire une classe pour un vecteur de descripteurs donné. Nous allons expliquer les techniques de classification que nous avons testé.

Pour faire nos tests pour chacun des algorithmes de classification, nous avons retiré 40 vecteurs de descripteurs pris au hasard dans notre base de données en tant que base de test et nous avons gardé le reste (environ un millier de vecteurs) comme base d'entraînement. Nous avons ensuite calculé un taux global de réussite en comptant le nombre de fois où la classe prédictive par l'algorithme pour chaque vecteur de test était effectivement sa vraie classe. Nous avons ensuite fait la moyenne sur 100 itérations de ce taux global de réussite pour obtenir un résultat stable.

9.2.1 K plus proches voisins

La méthode des K plus proches voisins (KPPV) [13] est une des méthodes les plus simples de classification. Elle consiste simplement à calculer chaque distance entre le vecteur de descripteurs à tester et chaque vecteur de descripteurs de la base de données d'entraînement. On trie alors toutes ces distances par ordre croissant, et on compte les occurrences de chaque classe parmi les "k" premières valeurs de la liste triée. La classe ayant le plus d'occurrences dans cette liste est la classe "gagnante".

Un exemple visuel est présenté figure 16. Pour k=3 (frontière en trait plein) sur cette figure, le point vert serait considéré de la classe "triangle rouge". Pour k=5 (frontière en trait pointillé), il serait considéré de la classe "carré bleu".

Nous avons testé deux méthodes KPPV différentes [14] une méthode sans pondération (chaque image a la même valeur pour le décompte final des k premiers, qu'importe sa proximité avec le vecteur à tester - 17), et une méthode pondérée à la distance avec le vecteur testé (figure 18). On voit sur les deux courbes que la méthode est la plus efficace dans les deux cas k=1. Pondérer ne sert donc à rien lorsqu'il n'y a qu'un seul voisin⁶. Le meilleur taux de classification atteint avec la méthode KPPV est donc 86 %.

6. les résultats ne sont pas exactement identiques non plus en raison de la part de hasard dans le choix des vecteurs d'entraînement

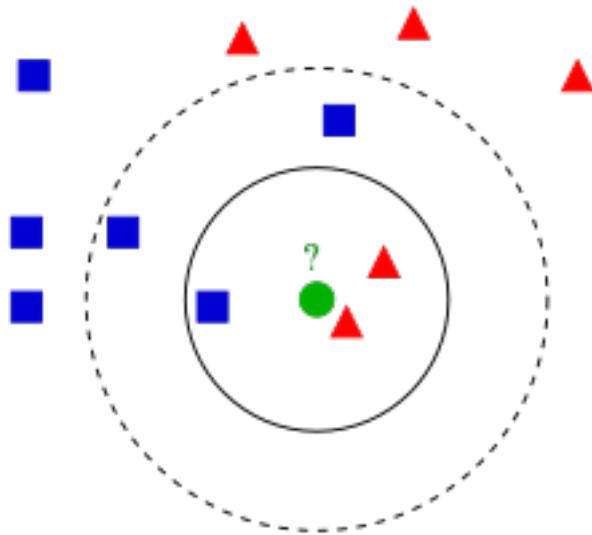


FIGURE 16 – Représentation visuelle d'une classification KPPV (d'après Wikipedia)

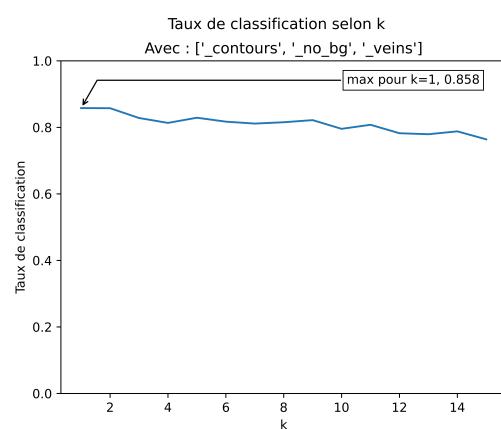
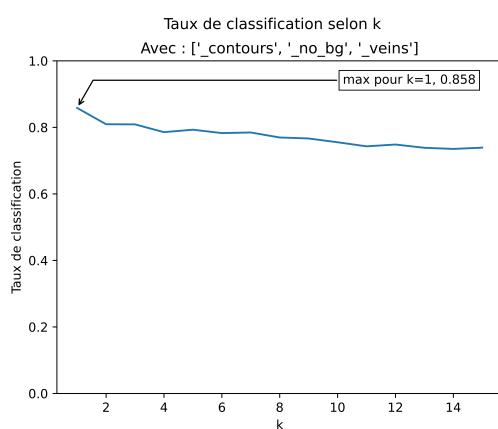


FIGURE 17 – Résultats de la classification KPPV non pondérée

FIGURE 18 – Résultats de la classification KPPV pondérée

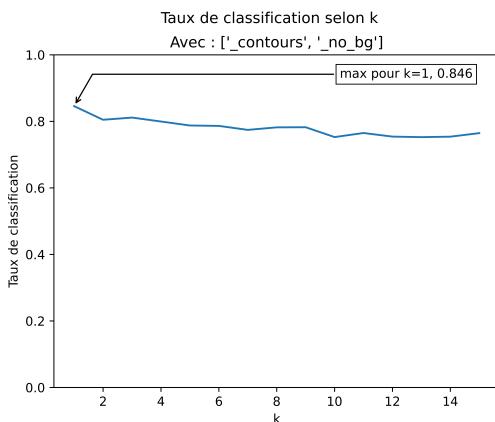


FIGURE 19 – Résultats de la classification KPPV non pondérée (sans les nervures)

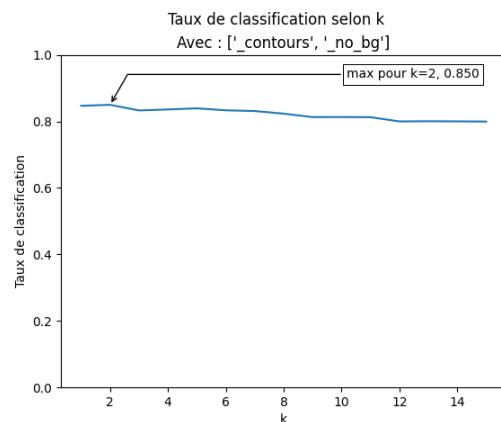


FIGURE 20 – Résultats de la classification KPPV pondérée (sans les nervures)

Les figures 19 et 20 sont les résultats obtenus pour la méthode KPPV sans descripteurs associés aux nervures de la feuille. On remarque que ces classificateurs ne sont pas très intéressants, car les résultats ne sont meilleurs que d'environ 1 %. L'information contenue dans la couleur des nervures est probablement en partie redondante ou peu utile.

L'exploitation des matrices de confusion obtenues avec les nervures pour $k=1$ (ce sont donc les mêmes résultats pour la KPPV pondérée ou non) avec les nervures pour la figure 21 et sans pour la figure 22. Montre que la présence de descripteurs pour les nervures permet à l'algorithme de moins confondre le Phosphore avec d'autres classes, mais que cela introduit de la confusion, car les carences en Azote sont plus souvent prises pour des carences en Phosphore, et les carences en Potassium plus souvent prises pour des carences en Azote.

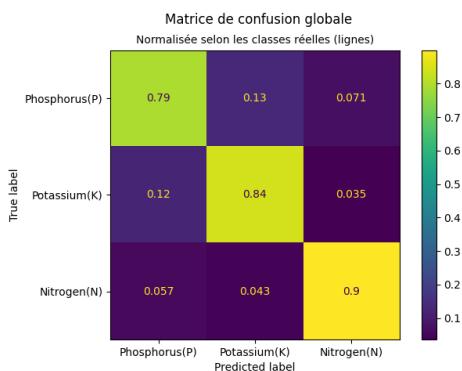


FIGURE 21 – Matrice de confusion pour $k=1$

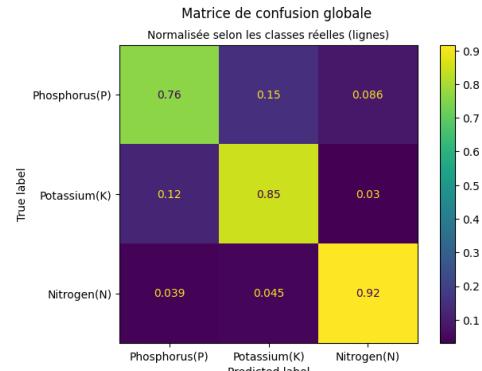


FIGURE 22 – Matrice de confusion pour $k=1$ (sans les nervures)

9.2.2 Descente du gradient

La classification par descente de gradient est une technique de machine learning qui consiste à entraîner un modèle de classification en ajustant les poids et les biais du modèle par étapes, en utilisant une fonction de coût et une méthode de gradient pour minimiser cette fonction de coût [15].

Le processus commence par initialiser les poids et les biais du modèle avec des valeurs aléatoires. Ensuite, le modèle est entraîné sur un ensemble de données d'entraînement en effectuant les étapes suivantes :

- Le modèle effectue une prédiction pour chaque exemple dans l'ensemble de données d'entraînement en utilisant les poids et les biais actuels.
- La fonction de coût est calculée pour mesurer l'erreur entre les prédictions du modèle et les étiquettes réelles des exemples dans l'ensemble de données d'entraînement.
- La méthode de gradient est utilisée pour déterminer la direction dans laquelle les poids et les biais doivent être ajustés pour minimiser la fonction de coût. Cette direction est appelée le gradient de la fonction de coût.
- Les poids et les biais sont ajustés dans la direction opposée au gradient de la fonction de coût, avec un pas de descente de gradient déterminé par un paramètre appelé le taux d'apprentissage.
- Les 4 étapes sont répétées pour un certain nombre d'itérations, jusqu'à ce que la fonction de coût soit suffisamment minimisée ou que le modèle atteigne un critère d'arrêt prédéfini.

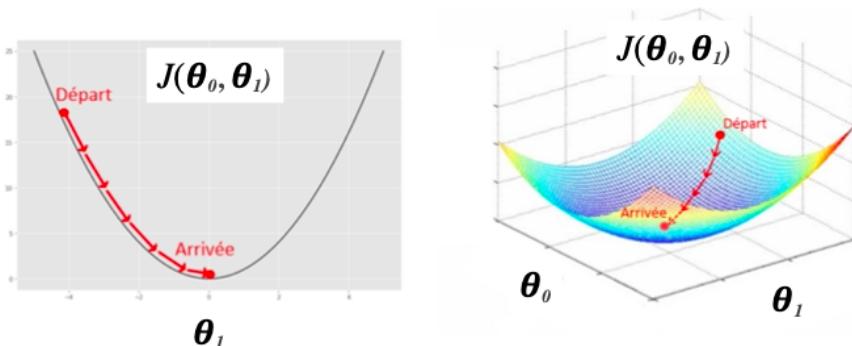


FIGURE 23 – Représentation visuelle de la descente du gradient en 2D et 3D

La figure 23 est une représentation plus visuelle de la minimisation de la fonction coût.

Lors de l'implémentation de cette méthode, comme il y a trois classes à déterminer, nous avons implémenté une méthode suivant le principe un contre tous. Le principe est le suivant :

On calcule la probabilité avec la méthode de descente du gradient pour chaque classe. Ainsi, la méthode donne donc la probabilité d'appartenir à la classe. Ainsi, lors de la prédiction, on choisit la classe ayant la plus grande probabilité.

De plus, nous avons utilisé la fonction sigmoïde pour faire une prédiction de la

classe. Il faut donc augmenter la probabilité que les échantillons (y) sachant les variables d'entrée (x) sont la valeur 1 si l'échantillon appartient à la classe 0 sinon (on note ce vecteur d). Pour augmenter, cette probabilité, on applique à chaque échantillon, des poids et des biais (le vecteur θ). Soit, la fonction à maximiser est : $p(y = d/x; \theta)$ (la probabilité que y soit égale à d sachant les entrées x en appliquant les poids θ). On cherche donc à minimiser l'opposé de cette probabilité avec la méthode de descente du gradient, en ajustant le vecteur θ .

Pour ajuster les poids et les biais (θ), il est important de bien choisir le taux d'apprentissage et le nombre d'itérations. Lorsque le taux d'apprentissage est trop important, cela peut entraîner une instabilité numérique et une divergence du processus d'optimisation. Le modèle peut sauter d'un minimum local à l'autre sans jamais converger vers le minimum global souhaité.

D'un autre côté, un taux d'apprentissage trop faible peut ralentir le processus d'optimisation, ce qui peut prendre beaucoup de temps pour atteindre une solution satisfaisante.

Il est donc important de trouver un taux d'apprentissage approprié pour chaque modèle et chaque problème.

Pour réaliser cela, on a effectué un grand nombre de tests avant de trouver les hyper-paramètres les plus intéressants. Dans un premier temps, on réalise quelques tests sur des plages bien distinctes. Cependant, la méthode pouvant prendre un temps important avant de converger, étant donné le nombre d'échantillons et le nombre de descripteurs, on a limité le nombre d'itérations à 10 000. Ceci entraînait déjà des programmes lents. D'autre part, pour s'assurer que les résultats n'étaient pas dus à l'échantillon choisi, on a réalisé une validation croisée. Cela consiste à séparer les échantillons d'apprentissage en k ensemble appelé *folds*. Pour chacun de ces groupes, on teste le modèle qu'elle aura entraîné avec les $k-1$ groupes restants. Le k choisi était 5.

On a ainsi obtenu avec les nervures, le tableau 24 et sans les nervures, le tableau 25, sur la première colonne les différents taux d'apprentissage, et sur la première ligne les différents nombres d'itérations.

	100	500	1000	5000	10000
0.1	0.6703	0.7443	0.79	0.7744	0.811
0.05	0.7178	0.7982	0.7991	0.8009	0.8183
0.01	0.7571	0.7808	0.7954	0.7973	0.8192
0.005	0.7525	0.7735	0.7808	0.7991	0.8128
0.001	0.7196	0.7525	0.7571	0.7808	0.8037

	100	500	1000	5000	10000
0.1	0.7507	0.7799	0.7772	0.8064	0.8119
0.05	0.7662	0.8055	0.7963	0.8091	0.8174
0.01	0.7479	0.779	0.7982	0.7927	0.8027
0.005	0.7397	0.7662	0.779	0.8	0.7927
0.001	0.7242	0.7406	0.7479	0.779	0.7982

FIGURE 24 – Tableau de recherche des hyper-paramètres optimaux avec les nervures

On observe ainsi que dès 500 itérations, il est possible d'obtenir des résultats de

FIGURE 25 – Tableau de recherche des hyper-paramètres optimaux sans les nervures

presque 80 %. Cependant, si on cherche à optimiser les résultats, on obtient jusqu'à 82 % avec un taux d'apprentissage de 0.01 et 10 000 itérations. Ceci prend cependant un temps assez important (environ 5 min) avant de finir. Pour pouvoir comparer avec les autres solutions, on a réalisé avec un échantillon de 40 tests, et sur 100 itérations différentes la solution optimale. On a ainsi obtenu un taux de classification de 82 % avec les nervures et 84 % sans les nervures. Les matrices de confusion obtenues sont présentées figures 26 et 27. On remarque que la discrimination du Phosphore et du Potassium est toujours difficile. De plus, le Phosphore est la classe la moins bien prédite.

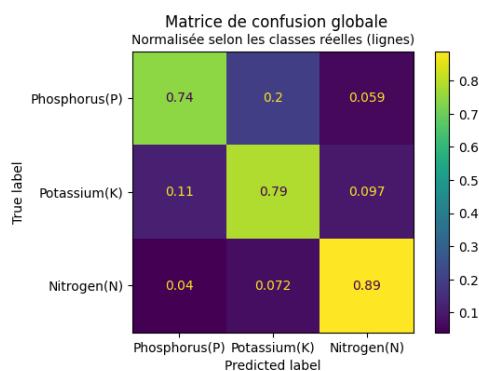


FIGURE 26 – Matrice de confusion avec les
nervures

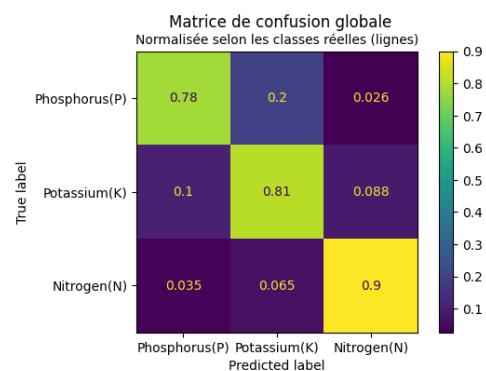


FIGURE 27 – Matrice de confusion sans les
nervures

9.2.3 Réseau de neurones

La classification par réseau de neurones est une technique très utilisée pour faire des classifications (*cf* état de l'art). Les réseaux de neurones sont inspirés du fonctionnement du cerveau humain et sont constitués de nombreux neurones interconnectés qui effectuent des calculs pour produire des sorties. [15]

Le processus d'apprentissage des réseaux de neurones commence par l'initialisation de poids aléatoires. Ensuite, les données sont introduites dans le réseau et les sorties sont calculées par propagation avant à travers les couches de neurones, jusqu'à ce que la sortie finale soit produite. La sortie produite est comparée à la sortie attendue, et une fonction de coût est utilisée pour mesurer l'écart entre les deux.

Ensuite, l'algorithme de rétro-propagation de l'erreur est utilisé pour ajuster les poids du réseau en fonction de l'erreur de sortie. L'objectif est de minimiser la fonction de coût en ajustant les poids pour améliorer la précision du réseau. Ce processus est répété plusieurs fois jusqu'à ce que la précision du réseau atteigne un niveau satisfaisant ou que l'algorithme atteigne un nombre prédéfini d'itérations.

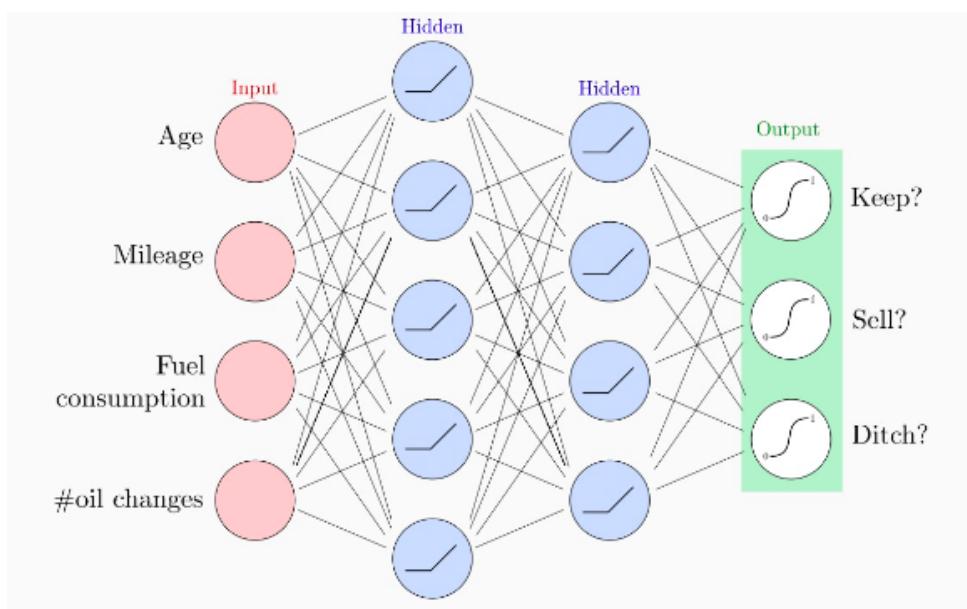


FIGURE 28 – Exemple de réseau de neurones

À chaque couche, on applique une fonction avec les paramètres de la couche précédente, ajustés de manière à minimiser la fonction coût établie. Le choix de la fonction coût est donc très important, ainsi que le nombre de couches intermédiaires et de neurones pour chaque couche. Ces paramètres influent directement sur les résultats de la classification.

La dernière couche a un rôle particulier puisqu'elle permet ici de désigner à quelle classe l'échantillon appartient. Pour cela, on choisit donc une probabilité. La fonction *softmax* est particulièrement utile pour ce genre de classification, car elle permet d'avoir pour chaque classe la probabilité d'appartenir à celle-ci. De plus, celle-ci est normalisée (c'est-à-dire, en sommant toutes les sorties, on obtient 1), voir la formule ci-dessous.

$$\text{softmax}(a_k) = \frac{\exp(a_k)}{\sum_{i=1}^3 \exp(a_i)} \quad (1)$$

On a donc décidé d'utiliser une fonction *softmax* pour la dernière couche et 3 neurones, un pour chaque classe. Lorsqu'on cherche à résoudre un problème de classification multiple et qu'on utilise une fonction *softmax* la fonction coût la plus adaptée est la fonction entropie croisée (ou *cross-entropy loss* en anglais). La formule, avec y , la valeur cible et \hat{y} , la variable de prédiction et n qui est le nombre d'échantillons, est la suivante :

$$H(y, \hat{y}) = - \sum_{i=1}^n y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \quad (2)$$

Pour la fonction d'activation, par souci de simplicité, on a décidé d'utiliser la fonction ReLU (*Rectified Linear Unit*) qui est la plus simple et la plus utilisée pour des problèmes de classification [16]. Elle permet d'insérer de la non-linéarité dans le problème, toute en étant calculable rapidement. Une représentation est donnée figure 29.

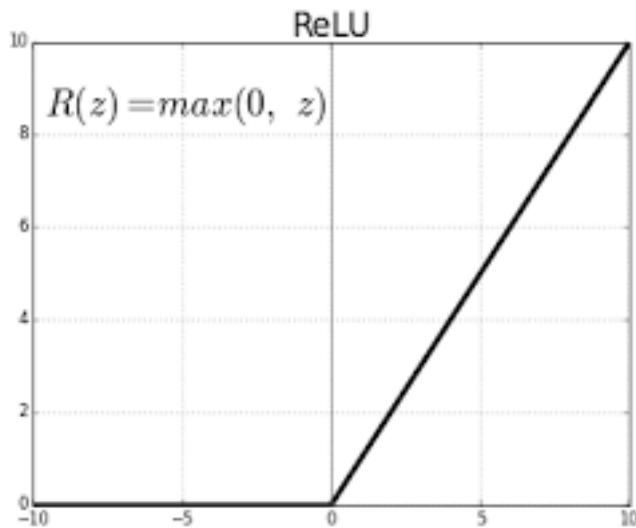


FIGURE 29 – Fonction ReLU

Pour déterminer, les autres paramètres, c'est-à-dire le nombre de couches, le nombre de neurones et les hyper-paramètres, on a réalisé un grand nombre de tests. Les hyper-paramètres sont, comme pour la méthode de descente du gradient, le nombre d'itérations et le taux d'apprentissage.

Une fois la méthode implémentée, nous avons réalisé un grand nombre de tests pour déterminer les meilleures performances possibles. Tout d'abord, nous avons essayé avec seulement une couche cachée (avec la fonction ReLU). Nous avons fait varier le nombre de neurones de 10 à 200. Les tests avec plus de 200 neurones prenaient beaucoup de temps et nous avons obtenu de bons résultats avec 200 neurones. Ensuite, de même que pour la descente du gradient, il valait trouver les bons hyper-paramètres, nous avons ainsi fait

varier le taux d'apprentissage (α) de 0.001 à 10^{-6} et le nombre d'itérations de 20 à 500. Voici les tableaux des résultats :

tableau de résultat taux de classification des tests avec alpha=0.001

	10	20	50	100	200
20	0.33	0.3	0.3	0.18	0.22
50	0.2	0.26	0.19	0.3	0.31
100	0.36	0.35	0.43	0.33	0.28
500	0.38	0.38	0.38	0.38	0.38

FIGURE 30 – taux de classification sur les tests avec alpha = 0.001

tableau de résultat taux de classification de l'apprentisaage avec alpha=0.001

	10	20	50	100	200
20	0.24	0.27	0.35	0.27	0.22
50	0.32	0.25	0.17	0.26	0.24
100	0.28	0.38	0.23	0.22	0.22
500	0.38	0.38	0.38	0.38	0.38

FIGURE 31 – taux de classification sur les données d'apprentissage avec alpha = 0.001

tableau de résultat taux de classification des tests avec alpha=0.0001

	10	20	50	100	200
20	0.57	0.61	0.7	0.69	0.74
50	0.6	0.66	0.72	0.75	0.83
100	0.65	0.71	0.78	0.81	0.82
500	0.74	0.76	0.8	0.83	0.83

FIGURE 32 – taux de classification sur les tests avec alpha = 0.0001

tableau de résultat taux de classification de l'apprentisaage avec alpha=0.0001

	10	20	50	100	200
20	0.62	0.65	0.76	0.75	0.74
50	0.68	0.7	0.83	0.84	0.95
100	0.7	0.78	0.93	1.0	1.0
500	0.86	0.98	1.0	1.0	1.0

FIGURE 33 – taux de classification sur les données d'apprentissage avec alpha = 0.0001

tableau de résultat taux de classification des tests avec alpha=1e-05

tableau de résultat taux de classification de l'apprentisaage avec alpha=1e-05

	10	20	50	100	200
20	0.56	0.56	0.6	0.61	0.63
50	0.61	0.63	0.64	0.67	0.7
100	0.64	0.64	0.66	0.68	0.71
500	0.65	0.67	0.71	0.75	0.78

FIGURE 34 – taux de classification sur les tests avec alpha = 10^{-5}

	10	20	50	100	200
20	0.58	0.6	0.64	0.67	0.67
50	0.62	0.65	0.69	0.71	0.76
100	0.64	0.68	0.74	0.79	0.83
500	0.67	0.71	0.86	0.96	1.0

FIGURE 35 – taux de classification sur les données d'apprentissage avec alpha = 10^{-5}

tableau de résultat taux de classification des tests avec alpha=1e-06

	10	20	50	100	200
20	0.38	0.4	0.44	0.47	0.52
50	0.45	0.44	0.55	0.58	0.56
100	0.52	0.55	0.54	0.59	0.59
500	0.59	0.62	0.66	0.68	0.69

 FIGURE 36 – taux de classification sur les tests avec alpha = 10^{-6}

tableau de résultat taux de classification de l'apprentissage avec alpha=1e-06

	10	20	50	100	200
20	0.4	0.41	0.46	0.46	0.52
50	0.44	0.45	0.55	0.57	0.58
100	0.51	0.55	0.58	0.6	0.62
500	0.62	0.65	0.72	0.73	0.78

 FIGURE 37 – taux de classification sur les données d'apprentissage avec alpha = 10^{-6}

On peut ainsi observer que les meilleurs résultats sont obtenus pour un taux de classification entre 0.0001 et 10^{-5} , ainsi que pour un nombre de neurones supérieur à 50 et un nombre d'itérations aussi supérieur à 50. On observe aussi que pour certaines valeurs, le taux de classification sur les données d'apprentissage sont de 1.0, c'est-à-dire que toutes les données d'apprentissage sont bien classifiées. Cependant, sur les données de tests, on obtient seulement autour de 80 % de taux de classification. Cela signifie que le modèle réalise du sur-apprentissage. Il devient trop adapté pour les données d'entraînement. Il faut donc simplifier le modèle, soit diminuer certains paramètres. Cependant, les résultats avec des paramètres moins précis du tableau n'ont pas de résultat supérieur à 83 %. On a donc essayé d'affiner les variations de paramètres dans les plages ayant les meilleurs résultats pour essayer d'obtenir de meilleurs résultats avec des modèles plus simples.

tableau de résultat taux de classification des tests avec alpha=6e-05

tableau de résultat taux de classification de l'apprentissage avec alpha=6e-05

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.573	0.665	0.69	0.686	0.7	0.712	0.689	0.716	0.718	0.747	0.729	0.73	0.777	0.755
100	0.637	0.694	0.735	0.722	0.743	0.757	0.774	0.762	0.792	0.757	0.779	0.796	0.796	0.799
200	0.664	0.698	0.733	0.766	0.767	0.785	0.792	0.799	0.831	0.799	0.805	0.805	0.784	0.816
300	0.7	0.751	0.752	0.788	0.782	0.775	0.791	0.795	0.798	0.802	0.792	0.806	0.829	0.801
400	0.709	0.742	0.769	0.783	0.802	0.79	0.802	0.79	0.786	0.796	0.795	0.827	0.783	0.807

 FIGURE 38 – taux de classification sur les tests avec alpha = 6.10^{-6}

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.607	0.669	0.768	0.738	0.751	0.775	0.761	0.816	0.807	0.848	0.779	0.841	0.861	0.851
100	0.65	0.714	0.804	0.818	0.845	0.824	0.921	0.92	0.86	0.934	0.974	0.963	0.992	
200	0.699	0.767	0.869	0.911	0.964	0.989	0.966	0.999	0.999	1.0	1.0	1.0	1.0	1.0
300	0.739	0.822	0.913	0.974	0.998	0.999	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
400	0.768	0.833	0.95	0.997	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

 FIGURE 39 – taux de classification sur les données d'apprentissage avec alpha = 6.10^{-6}

tableau de résultat taux de classification des tests avec alpha=8e-05

tableau de résultat taux de classification de l'apprentissage avec alpha=8e-05

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.608	0.675	0.699	0.732	0.738	0.73	0.763	0.742	0.727	0.77	0.755	0.748	0.774	
100	0.639	0.721	0.743	0.747	0.767	0.774	0.758	0.807	0.788	0.821	0.802	0.809	0.811	0.795
200	0.673	0.726	0.749	0.784	0.795	0.789	0.809	0.805	0.818	0.82	0.802	0.804	0.804	
300	0.726	0.742	0.768	0.779	0.772	0.78	0.795	0.808	0.789	0.808	0.837	0.821	0.812	
400	0.733	0.753	0.765	0.781	0.781	0.788	0.799	0.803	0.832	0.821	0.805	0.804	0.805	0.81

 FIGURE 40 – taux de classification sur les tests avec alpha = 8.10^{-5}

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.632	0.735	0.777	0.776	0.812	0.808	0.816	0.835	0.857	0.834	0.877	0.882	0.878	0.862
100	0.664	0.755	0.837	0.845	0.876	0.933	0.918	0.958	0.964	0.983	0.996	0.998	0.997	0.998
200	0.738	0.816	0.89	0.971	0.995	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
300	0.772	0.846	0.953	0.997	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
400	0.801	0.918	0.992	0.999	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

 FIGURE 41 – taux de classification sur les données d'apprentissage avec alpha = 8.10^{-5}

tableau de résultat taux de classification des tests avec alpha=0.0001

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.538	0.658	0.691	0.731	0.755	0.751	0.733	0.743	0.725	0.75	0.765	0.799	0.731	0.788
100	0.681	0.706	0.722	0.754	0.784	0.799	0.8	0.774	0.81	0.811	0.807	0.816	0.803	0.82
200	0.717	0.748	0.755	0.78	0.791	0.811	0.796	0.812	0.807	0.805	0.807	0.806	0.828	0.805
300	0.721	0.756	0.764	0.788	0.787	0.786	0.793	0.811	0.806	0.814	0.812	0.809	0.831	0.851
400	0.739	0.775	0.772	0.794	0.802	0.808	0.801	0.805	0.821	0.814	0.799	0.817	0.821	0.823

 FIGURE 42 – taux de classification sur les tests avec alpha = $10 \cdot 10^{-5}$

tableau de résultat taux de classification de l'apprentissage avec alpha=0.0001

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.664	0.719	0.71	0.728	0.701	0.754	0.803	0.786	0.819	0.774	0.798	0.789	0.79	
100	0.689	0.696	0.719	0.779	0.788	0.795	0.804	0.803	0.807	0.829	0.829	0.823		
200	0.72	0.748	0.783	0.772	0.784	0.788	0.816	0.796	0.812	0.823	0.82	0.811	0.816	0.828
300	0.746	0.771	0.775	0.786	0.806	0.795	0.811	0.825	0.827	0.828	0.828	0.845	0.834	
400	0.743	0.773	0.787	0.807	0.802	0.805	0.793	0.8	0.804	0.814	0.82	0.827	0.826	0.823

 FIGURE 43 – taux de classification sur les données d'apprentissage avec alpha = $10 \cdot 10^{-5}$

tableau de résultat taux de classification des tests avec alpha=0.00012

tableau de résultat taux de classification de l'apprentissage avec alpha=0.00012

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.662	0.709	0.727	0.731	0.72	0.763	0.74	0.782	0.769	0.776	0.783	0.813	0.823	0.814
100	0.689	0.696	0.719	0.779	0.788	0.795	0.804	0.803	0.807	0.829	0.829	0.823		
200	0.724	0.74	0.777	0.79	0.805	0.791	0.802	0.815	0.804	0.812	0.829	0.811	0.824	0.816
300	0.729	0.763	0.785	0.787	0.807	0.817	0.814	0.812	0.82	0.816	0.804	0.82	0.824	0.823
400	0.747	0.775	0.785	0.784	0.791	0.795	0.815	0.805	0.822	0.814	0.816	0.829	0.823	0.807

 FIGURE 44 – taux de classification sur les tests avec alpha = $12 \cdot 10^{-5}$

 FIGURE 45 – taux de classification sur les données d'apprentissage avec alpha = $12 \cdot 10^{-5}$

tableau de résultat taux de classification des tests avec alpha=0.00014

tableau de résultat taux de classification de l'apprentissage avec alpha=0.00014

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.679	0.709	0.727	0.731	0.72	0.763	0.74	0.782	0.769	0.776	0.783	0.813	0.823	0.814
100	0.689	0.727	0.753	0.777	0.791	0.788	0.799	0.791	0.824	0.81	0.823	0.824	0.826	0.816
200	0.724	0.74	0.777	0.79	0.805	0.791	0.802	0.815	0.804	0.812	0.829	0.811	0.824	0.816
300	0.729	0.763	0.785	0.787	0.807	0.817	0.814	0.812	0.82	0.816	0.804	0.82	0.824	0.823
400	0.747	0.775	0.785	0.784	0.791	0.795	0.815	0.805	0.822	0.814	0.816	0.829	0.823	0.807

 FIGURE 46 – taux de classification sur les tests avec alpha = $14 \cdot 10^{-5}$

 FIGURE 47 – taux de classification sur les données d'apprentissage avec alpha = $14 \cdot 10^{-5}$

tableau de résultat taux de classification des tests avec alpha=0.0002

tableau de résultat taux de classification de l'apprentissage avec alpha=0.0002

	10	20	30	40	50	60	70	80	90	100	120	130	140	150
50	0.607	0.624	0.743	0.676	0.768	0.784	0.725	0.777	0.796	0.799	0.824	0.75	0.741	
100	0.689	0.746	0.773	0.784	0.731	0.817	0.788	0.815	0.753	0.793	0.727	0.783	0.804	0.799
200	0.722	0.773	0.785	0.78	0.781	0.803	0.771	0.815	0.81	0.811	0.769	0.794	0.822	0.824
300	0.724	0.797	0.801	0.782	0.803	0.788	0.812	0.81	0.815	0.813	0.818	0.819	0.819	0.794
400	0.768	0.745	0.784	0.798	0.802	0.811	0.798	0.823	0.81	0.818	0.823	0.824	0.805	0.792

 FIGURE 48 – taux de classification sur les tests avec alpha = $20 \cdot 10^{-5}$

 FIGURE 49 – taux de classification sur les données d'apprentissage avec alpha = $20 \cdot 10^{-5}$

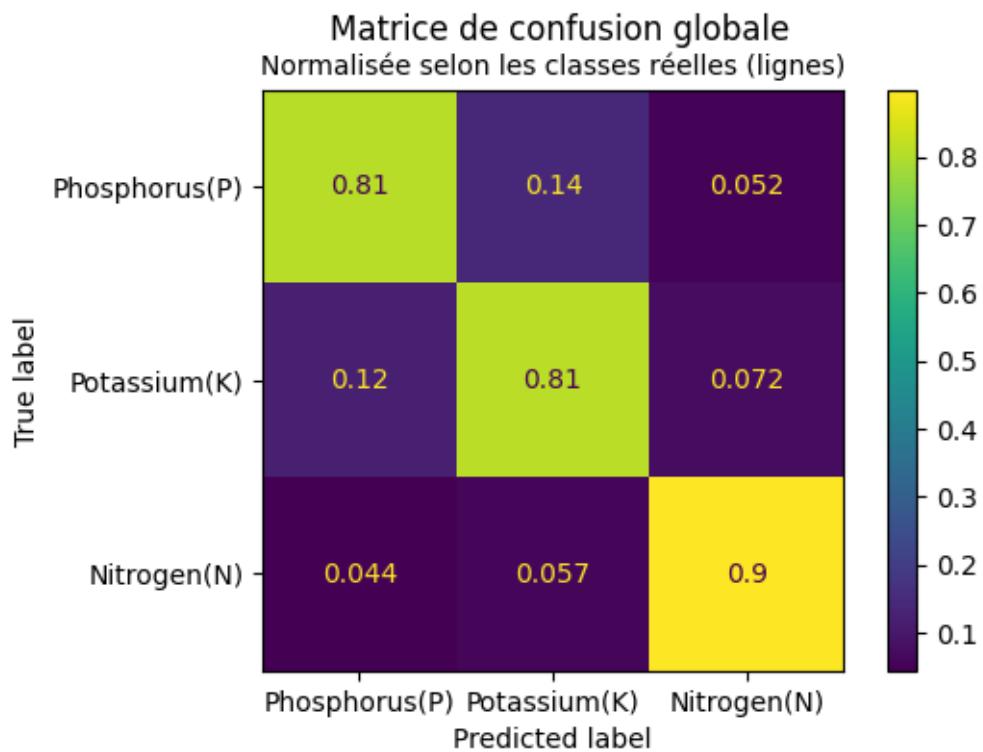


FIGURE 50 – Matrice de confusion

On observe que de nombreux tests ont des taux de classification sur l'apprentissage de 1.0, même avec un modèle plus simple. Cependant, les taux de classification sur les tests ne dépassent jamais les 85 %. Ceci signifie que pour obtenir un meilleur taux de classification, il faudrait posséder une base de données plus importante où avoir des descripteurs plus discriminants.

Pour comparer cette solution avec les autres, on a réalisé une phase de tests dans les mêmes conditions, c'est-à-dire un échantillon de 40 et sur 100 réalisations différentes. On obtient ainsi un résultat de 84,4 % et la matrice de confusion présentée figure 50.

9.2.4 Machines à vecteurs de support (SVM)

Les SVM (Machines à vecteurs de support) sont un ensemble de techniques qui permettent de discriminer et de classifier des vecteurs de descripteurs à l'aide de frontières [17]. Pour cela, on utilise des fonctions, appelées "noyaux" qui ont pour objectif de transformer ces vecteurs pour pouvoir faire des frontières linéaires (hyperplans). La figure 51 présente le problème plus visuellement.

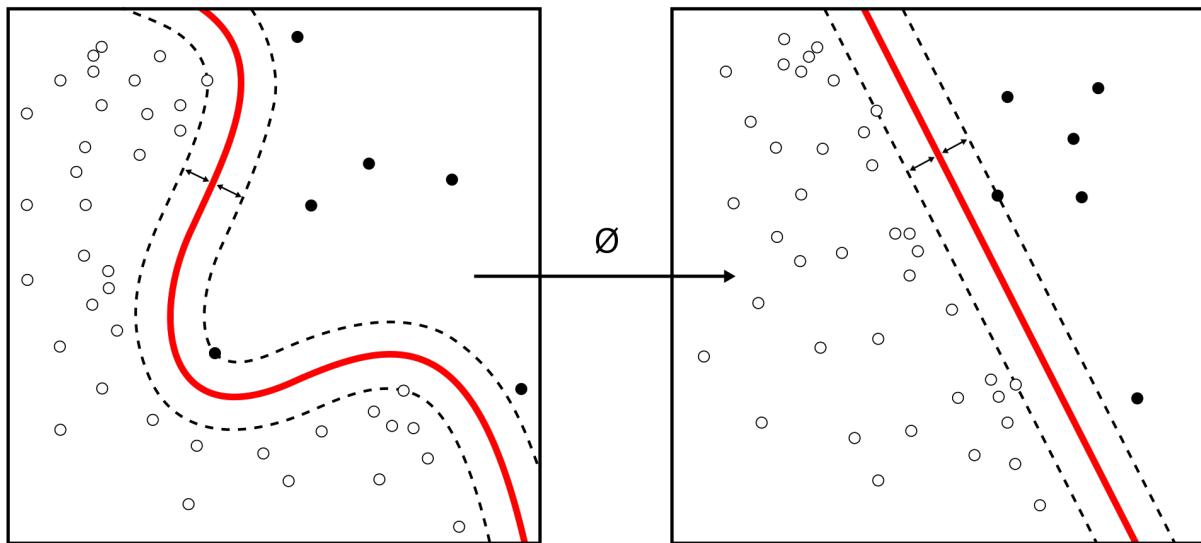


FIGURE 51 – Représentation visuelle d'une application de fonction noyau (d'après *wikipedia*)

Pour mettre en place cette méthode, nous avons utilisé la librairie python *sklearn* qui est accompagnée d'une documentation très exhaustive [18]. Pour avoir une estimation laquelle des fonctions noyau utiliser, nous avons commencé par essayer tous les noyaux disponibles sur la librairie, à savoir : linéaire, polynomiale (degré 3), rbf, sigmoïde. Ces essais ont été faits avec les paramètres par défaut : coefficient de régularisation (C) à 1 et γ (pour rbf, polynomiale et sigmoïde) à $1/(N_{desc} \times Var(X))$. Ces tests ont été faits avec et sans la série de descripteurs de la couleur des nervures. Les résultats de ce premier jet sont présentés figure 52.

On voit finalement que les meilleurs résultats sont obtenus pour le noyau rbf. On remarque aussi que les résultats sont moins bons avec les nervures que sans, sauf pour le noyau rbf où les résultats sont à peine meilleurs avec.

Nous pouvons aussi nous intéresser aux matrices de confusion de ces différents noyaux présentées figure 53. Nous pouvons faire les observations suivantes :

- La fonction polynomiale de degré 3 est la seule à mieux classifier les carences en Phosphore que celles en Potassium, malgré que dans 20% des cas, lorsqu'on est en présence d'une carence de Phosphore, la prédiction soit Potassium.
- La fonction sigmoïde a des performances médiocres sauf pour l'Azote.
- De manière générale, discriminer Phosphore et Potassium est une tâche peu réussie (environ 20% d'échec).

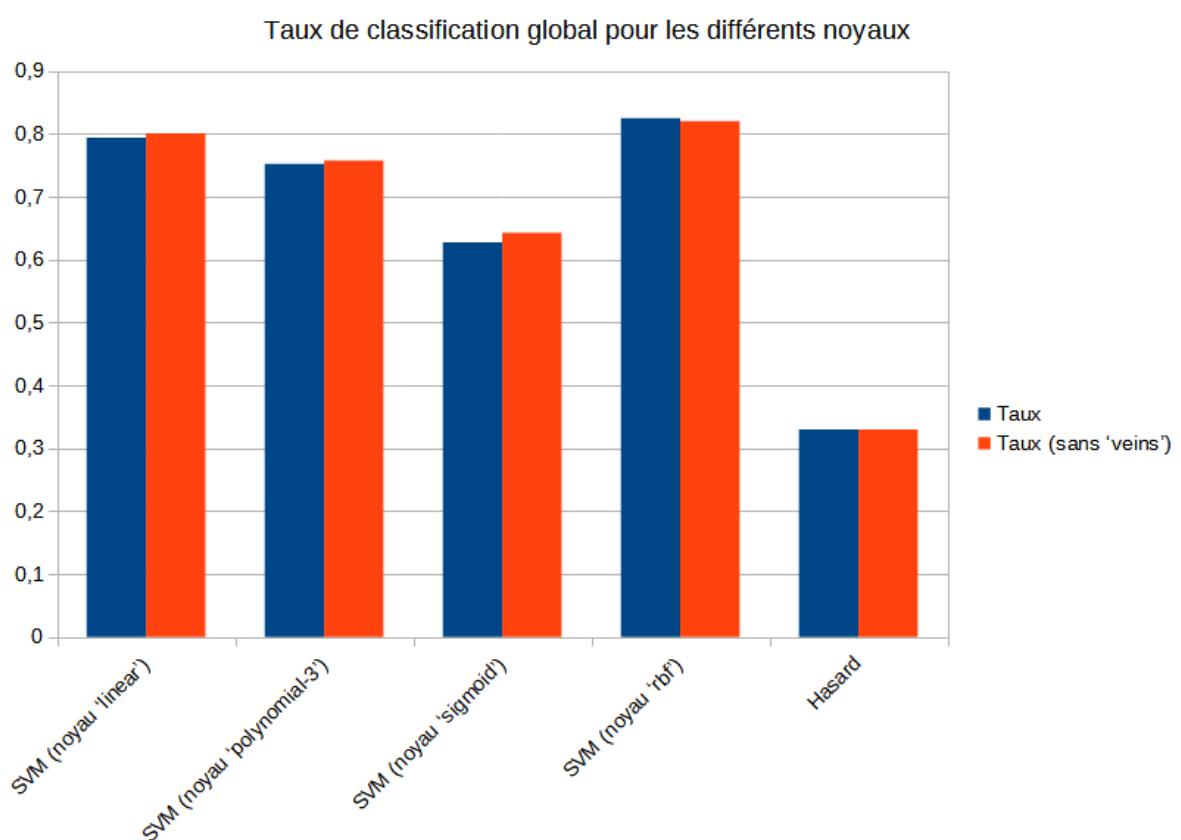


FIGURE 52 – Comparaison des différents noyaux par défaut (avec et sans les nervures)

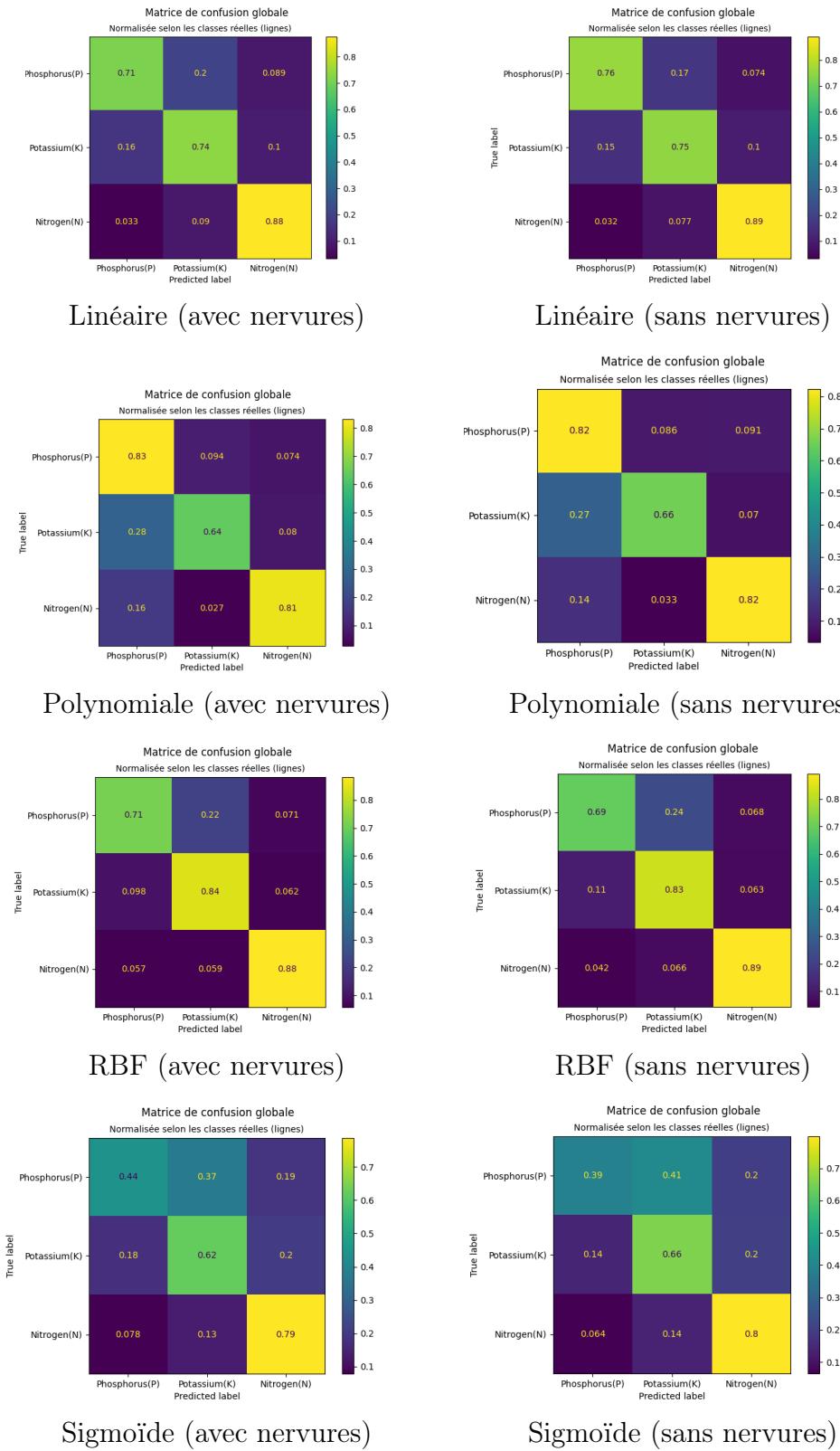


FIGURE 53 – Matrices de confusion pour chacun des noyaux, avec et sans les nervures

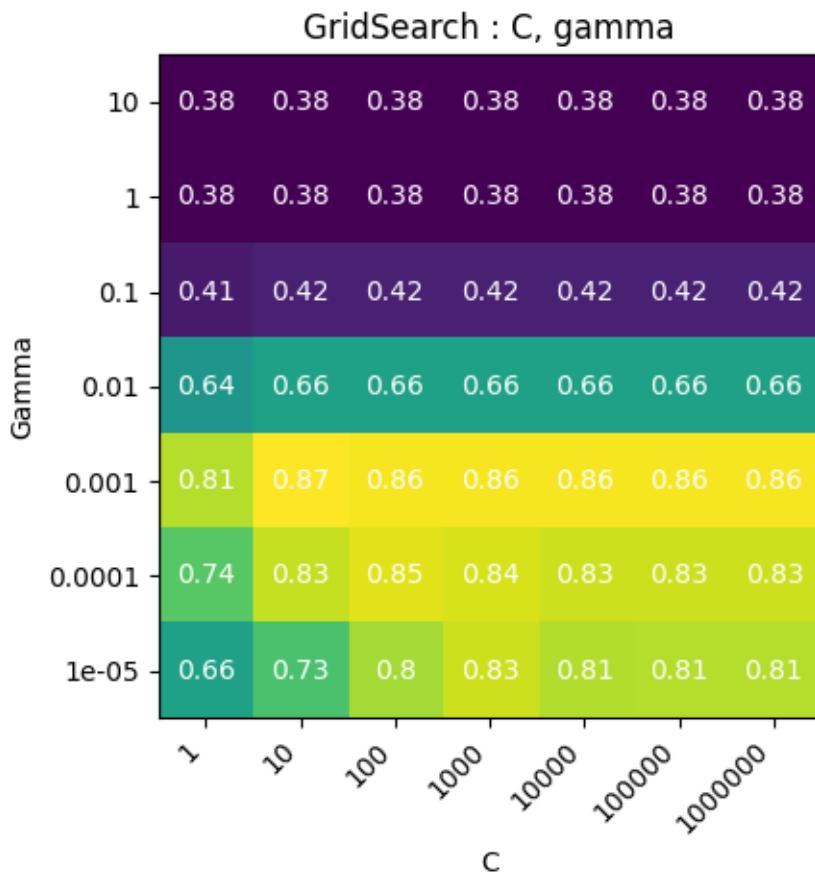


FIGURE 54 – Recherche par maillage : amélioration des résultats obtenus avec le noyau rbf

- De manière générale, identifier les carences en Azote est une tâche réussie ($> 70\%$).

Nous avons voulu améliorer les résultats rendus par la fonction rbf en modifiant les hyper-paramètres de celle-ci : C et γ . Pour diminuer les coûts de calcul, nous avons utilisé la méthode *5-fold* qui divise la base de données en 5 groupes : Pour chacun de ces groupes, elle va tester son modèle qu'elle aura entraîné avec les quatre groupes restants. Cela permet une bonne approximation des performances sans demander autant de temps de calcul. Nous avons utilisé des gammes de valeurs que nous retrouvions dans la littérature ([18] en particulier). Le résultat de ce maillage en utilisant la fonction *GridSearch* de *scikit-learn* est représenté visuellement figure 54.

On voit que les meilleurs hyper-paramètres sont $C = 10$ et $\gamma = 0.001$. En utilisant le même processus de test que les algorithmes précédents (voir tout début de cette partie), nous arrivons à un taux de classification de 90 %, et donc environ 8 % meilleur que la rbf non optimisée. Cela est donc efficace de modifier ces paramètres.

9.2.5 Analyse et interprétation des résultats : comparaison des performances des algorithmes

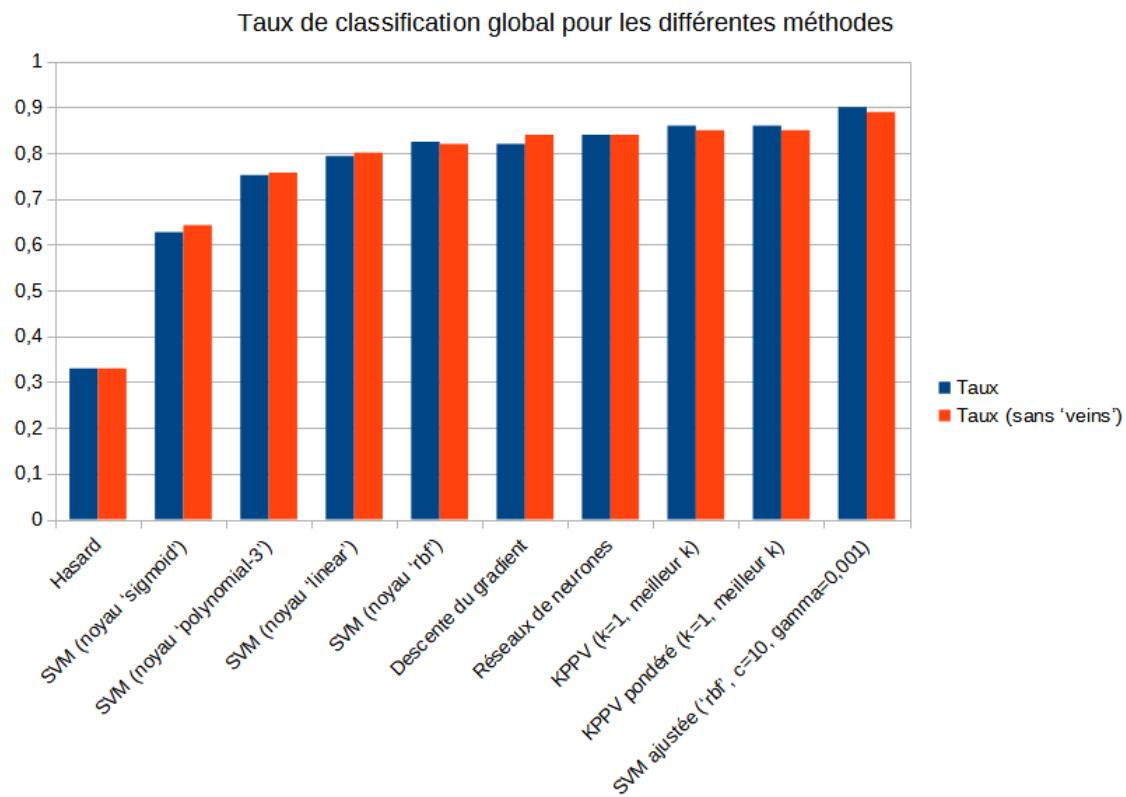


FIGURE 55 – Comparaison des différentes méthodes

Sur le graphe 55, on peut voir et comparer les différents résultats des méthodes de classification. Cependant, ces résultats ont été obtenus avec la base de données disponible : celle du riz carencé. Or, celle-ci possède uniquement trois classes, ainsi si certaines méthodes de classification peuvent paraître très bonnes ici, peut-être que sur une autre base de données bien plus fournie ce ne serait pas le cas. On peut notamment penser à la méthode KPPV qui a obtenu des bons résultats et était la plus rapide des quatre méthodes. Cependant, s'il y avait plus de classes, cette méthode devient moins adaptée, car elle peut devenir plus lente. En effet, il faut parcourir et calculer les distances entre le vecteur de descripteurs considéré et les vecteurs de la base de donnée d'entraînement.

À l'inverse, la méthode avec un réseau de neurones est plus compliquée et demande une grande quantité de calcul, mais ne semble pas apporter des meilleurs résultats. Cependant, si la base de données utilisée s'enrichissait, il serait possible que les performances de celle-ci soient démultipliées. On a vu en effet que le taux de classification sur les données d'apprentissage était bien supérieur à celui sur les données de tests.

La méthode de descente de gradient était une méthode particulièrement lente pour les résultats obtenus. De plus, une augmentation du nombre de classes pourrait conduire à des chevauchements entre les différentes classes. Cette méthode ne semble donc pas particulièrement adaptée pour ce problème.

La méthode SVM a obtenu les meilleurs résultats une fois optimisée, cependant celle-ci était aussi plutôt longue, et il a fallu passer par une étape d'optimisation des hyper-paramètres particulièrement chronophage.

Cependant, il est difficile de prévoir comment évolueront les méthodes si on agrandit la base de données. Il serait donc nécessaire d'ajuster les différentes méthodes pour déterminer celle qui serait la plus adaptée avec une base de données plus complète.

Il sera aussi intéressant de déterminer sur quel support notre outil serait potentiellement installé, car le choix de la méthode de classification dépend aussi beaucoup du support disponible (téléphone portable ou ordinateur, calculs sur un *cloud* ou *in situ*, librairies compatibles avec l'installation, etc.)

10 Limites et prolongement de l'étude

10.1 Discriminer Potassium et Phosphore

De ce que les matrices de confusion nous ont montré tout au long de ce rapport, le plus dur est de discriminer les carences de Potassium et Phosphore. En reprenant le tableau 3, on remarque que la meilleure différence que l'on puisse exploiter est la différence de comportement de la couleur de l'extrémité de la feuille.

On pourrait imaginer de prolonger l'étude en créant un nouveau masque "extrémité" pour faire une analyse colorimétrique de cette partie-là de la feuille. Nous avions déjà commencé ce travail. Une des difficultés est que les feuilles des images ne sont pas forcément toutes dans la même direction. Il faut donc détecter le sens de la feuille.

Une autre question est celle de la longueur. Pour avoir un résultat exploitable (c'est-à-dire qui donnerait des résultats similaires pour des feuilles similaires), il faut que cette longueur considérée comme "extrémité" soit un certain pourcentage de la feuille. Il faut donc connaître la longueur de la feuille photographiée. Ceci n'est pas si simple, car les photos de la base de données utilisée sont très recadrées et la feuille n'est pas entière. Nous pourrions utiliser par exemple la courbure des bords de la feuille pour extrapoler cette longueur.

10.2 Ajouter des classes et intégrer d'autres bases de données

Cette étude a montré des résultats intéressants puisque l'on arrive à des taux de classification de 90 % avec certaines méthodes. Cependant, ces résultats sont difficilement comparables avec les études de l'état de l'art pour deux raisons principales :

La première est simple, notre base de données ne contient seulement trois classes, ainsi les résultats obtenus sont nécessairement meilleurs que lorsque l'on compare à une classification avec six classes (une pour chaque carence du tableau 3).

La deuxième concerne le nombre d'images, on peut penser que 300 images est une quantité importante, cependant certaines méthodes (comme le réseau de neurone) ont montré qu'il serait possible d'obtenir de meilleurs résultats avec une base de données plus fournie.

Il serait ainsi intéressant de pouvoir ajouter des bases de données à la nôtre pour pouvoir augmenter le nombre d'échantillons tout en augmentant le nombre de classes.

11 Conclusion

Lors de cette étude, nous avons constaté que la plupart des logiciels d'aide au diagnostic existants utilisent une approche "deep learning" sans analyse préalable des symptômes des carences. Nous avons donc proposé une démarche qui combine une analyse des carences sur les plantes, des méthodes d'extraction de données d'une image et de classification sur ces données.

Nous avons testé plusieurs types de classification, aussi bien très classiques (KPPV) que plus avancées (SVM, descente du gradient), et nous avons constaté que nos résultats étaient concluants (90 % de taux de classification pour la SVM optimisée) néanmoins pas aussi concluants que les approches du corpus bibliographique (jusqu'à 99 %). Ces résultats sont cohérents, car nous perdons une partie de l'information contenue dans l'image lorsque nous passons aux descripteurs.

Nous avons également constaté que le choix des descripteurs est crucial. Dans notre cas, les descripteurs choisis ne sont pas assez discriminants pour le Phosphore et le Potassium. Nous pourrions également obtenir de meilleurs résultats en utilisant des méthodes d'extraction pour d'autres zones de la feuille.

12 Bibliographie

Références

- [1] Carlos Alberto GARZA-ALONSO, Emilio OLIVARES-SÁENZ, Adriana GUTIÉRREZ-DÍEZ, Rigoberto E. VÁZQUEZ-ALVARADO et Alfredo LÓPEZ-JIMÉNEZ : Visual Symptoms, Vegetative Growth, and Mineral Concentration in Fig Tree (*Ficus carica* L.) Under Macronutrient Deficiencies. *Agronomy*, 9(12):787, novembre 2019.
- [2] R UCHIDA : Essential Nutrients for Plant Growth : Nutrient Functions and Deficiency Symptoms.
- [3] Guillaume ROUSSET : Apports des méthodes d'apprentissage profond pour la reconnaissance automatique des modes d'occupation des sols et d'objets par télédétection en milieu tropical.
- [4] Ümit ATILA, Murat UÇAR, Kemal AKYOL et Emine UÇAR : Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*, 61:101182, mars 2021.
- [5] B. SAI REDDY et S. NEERAJA : Plant leaf disease classification and damage detection system using deep learning models. *Multimedia Tools and Applications*, 81(17):24021–24040, juillet 2022.
- [6] Guillaume LOBET : Image Analysis in Plant Sciences : Publish Then Perish. *Trends in Plant Science*, 22(7):559–566, juillet 2017.
- [7] Arief Rais BAHTIAR, PRANOWO, Albertus Joko SANTOSO et Jujuk JUHARIAH : Deep Learning Detected Nutrient Deficiency in Chili Plant. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–4, Yogyakarta, Indonésie, juin 2020. IEEE.
- [8] Lili Ayu WULANDHARI, Alexander Agung Santoso GUNAWAN, Arie QURANIA, Pri-hastuti HARSANI, Triastinurmiantingsih , Ferdy TARAWAN et Riska Fauzia HERMAWAN : Plant Nutrient Deficiency Detection Using Deep Convolutional Neural Network, 2019.
- [9] Ukrit WATCHAREERUETAI, Pavit NOINONGYAO, Chaiwat WATTANAPAIBOONSUK, Puriwat KHANTIVIRIYA et Sutsawat DUANGSRISAI : Identification of Plant Nutrient Deficiencies Using Convolutional Neural Networks. In *2018 International Electrical Engineering Congress (iEECON)*, pages 1–4, Krabi, Thaïlande, mars 2018. IEEE.
- [10] MUHAMMAD ASRAF HAIRUDDIN, Nooritawati MD TAHIR et SHAH RIZAM SHAH BAKI : Overview of image processing approach for nutrient deficiencies detection in *Elaeis Guineensis*. In *2011 IEEE International Conference on System Engineering and Technology*, pages 116–120, Shah Alam, Malaisie, juin 2011. IEEE.
- [11] David CONDAMINET, Albrecht ZIMMERMANN, Bastien BILLIOT, Bruno CREMILLEUX et Sylvain PLUCHON : Using Data Science to Improve the Identification of Plant Nutritional Status. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 496–505, sydney, Australie, octobre 2020. IEEE.

- [12] Donald Danforth Plant Science CENTER : documentation for PlantCV. <https://plantcv.readthedocs.io/en/stable/>.
- [13] WIKIPÉDIA : Méthode des k plus proches voisins — wikipédia, l'encyclopédie libre, 2022. [En ligne ; Page disponible le 9-novembre-2022].
- [14] Eve MATHIEU-DUPAS : Algorithme des k plus proches voisins pondérés et application en diagnostic. *42èmes Journées de Statistique*, (42), 2010.
- [15] Emmanuel DELLANDRÉA : Cours d'approfondissement : Analyse de données et reconnaissance des formes, 2022.
- [16] Machine Learning for Artists ML4A : Les réseaux de neurones, 2021.
- [17] WIKIPÉDIA : Machine à vecteurs de support — wikipédia, l'encyclopédie libre, 2023. [En ligne ; Page disponible le 4-janvier-2023].
- [18] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT et E. DUCHESNAY : Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.