Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE SAAD DAHLEB DE BLIDA 1

Faculté des Sciences

Département d'Informatique

MÉMOIRE DE MASTER INFORMATIQUE

CLASSIFICATION AUTOMATIQUE ET APPRENTISSAGE APPROFONDI DES IMAGES: APPLICATION SUR LE DOMAINE DE L'AGRICULTURE.

Réalisé par :

MADOUR Amina
MEKERKEB ABERRANE Abir

Proposé et encadré par :

M CHEMCHEM Mohamed Amine
Mme OUKID Lamia

Composition de jury :

Mme GUESSOUM Dalila Président
Mme ZAHRA Fatma Zohra Examinateur

Soutenu le :
26/09/2018

Résumé

Notre PFE consiste à réaliser une étude de recherche portant sur l'application des méthodes d'analyse de données et d'apprentissage automatique sur des images récoltées dans le domaine d'agriculture. Une application fort intéressante de ce projet est de donner plus d'information sur un champ de plantation en indiquant la qualité de ses fruits et feuilles. On va appliquer une classification des fruits (les pommes) et des feuilles atteintes d'une maladie dans deux classes : saines ou infectées.

La première partie consiste à réaliser une classification en utilisant les réseaux de neurones à convolutions (Convolutional Neural Network CNN), commençant par la classification binaire dont le but est de classer les zones infectées d'un fruit ou/et une feuille de fruit selon les deux classes : fruit-infecté, fruit-sain.

Puis, effectuer une classification multiple comme dans l'exemple qu'on a pris : (bananes, fraises, oranges, poires) qui classifie les différentes images en entrée dans une des classes déjà entrainées.

La deuxième partie consiste à réaliser une étude comparative entre les résultats du CNN et des différents algorithmes de classification supervisée (Multilayer Perceptron MLP, Support Vector Machines SVM, K Nearest Neighbors K-NN).

En dernier, nous présentons l'outil YOLO (You Only Look Once), une nouvelle approche de la détection d'objets. Nous l'avons adapté pour la détection d'une classe des fruits (Oranges) en temps réel, en utilisant la caméra d'un device quelconque dans notre cas la caméra d'un pc portable. Le résultat sera l'encadrement de l'objet reconnu dans la zone de l'image captée à l'instant 't'.

Mots clé : Classification, détection, prédiction, apprentissage automatique, reconnaissance d'objets, description d'images, base d'apprentissage.

تلخيص

مشروع دراستنا يتمحور في البحث عن أساليب الاكتساب ومعالجة الصور المجمعة للتصنيف الآلي والتعلم العميق لصور الفاكهة، للإجابة على مختلف القضايا الزراعية.تطبيقنا الأساسي لهذا المشروع يتمثل في كشف المناطق المصابة من الفاكهة، داخل حقل زراعي.

الجزء الأول يتمثل في إجراء تصنيف باستخدام الشبكات العصبية التلافيفيةCNNبدءا من تصنيف ثنائي الذي غرضه تصنيف المناطق المصابة من الفاكهة و / أو ورقة الفاكهة وفقا لفئتين:الفواكه المصابة و غير المصابة.

ثم،قمنا بإجراء تصنيف متعدد لأنواع مختارة من الفاكهة (الموز؛الفراولة ؛البرتقال والأجاص)، تصنف الصور المختلفة كمدخل في واحدة من الفئات المدربة.

أما في الجزء الثاني، فقدقمنابإجراء دراسة مقارنة بين النتائج المختلفة المحصلة من CNN و مختلف الخوارزميات المستعملة (متعدد الطبقات المستقبلاتMLP ، المتجهات الداعمة للآلات SVM ، K-NNأقرب الجيران).

وأخيرا، نقدمYOLO نهجا جديدا للكشف ، للكشف عن فئة من الفواكه (البرتقال) في الوقت الحقيقي، والتي سوف تتعرف على وجود البرتقال بين مختلف الأشياء معطية نسبتها المئوية.

الكلمة المفتاح : تصنيف، تنبؤ، التعلم الأوتوماتيكي، رصد، التعرف على الأشياء ، قاعدة التعلم ، وصف الصورة .

Dédicace

Nous dédions ce modeste travail à

Nos chers parents ; Pour leur soutien, leur patience, leur

sacrifice, leur amour et Leurs conseils judicieux.

Nous espérons qu'un jour,

Nous pourrons leurs rendre un peu de ce qu'ils ont

Fait pour nous, que dieu leur prête bonheur et longue vie.

Nous dédions aussi ce travail à nos frères et

Sœurs, nos familles, nos amis,

Pour leurs affections et leurs encouragements,

Tous nos professeurs qui nous ont enseigné

Et à tous ceux qui nous sont chers.

Liste des tableaux

Liste des figures

Sommaire

Introduction générale

I. INTRODUCTION

1. Contexte du PFE

Depuis la nuit des temps, l'homme pratique la classification dans son quotidien, quand il essaie d'affecter des objets à leur classe (en observant leurs formats, couleurs, tailles … etc). De nos jours les informations sont facilement récupérables avec un coup minimal dans un système informatique qui les stocke, traite afin de mieux les comprendre. Cependant en ce qui concerne la classification d'image on applique des méthodes de classification, des algorithmes d'apprentissage automatique, et aussi les nouvelles approches de l'apprentissage profond. L'une des applications intéressantes de ce domaine est la détection automatique des fruits: ceci peut être réalisé en appliquant des approches de détection automatique, et aussi des approches d'apprentissage approfondi. Une autre application est la classification automatique des feuilles et fruits malades.

A l'heure actuelle, les systèmes de vision automatique sont de plus en plus répandus, les caméras sont installées partout dans notre quotidien. Elles sont utilisées pour réaliser de la vidéosurveillance (dans les espaces publiques, aéroports ...etc), elles aident à la conduite, la détection d'obstacles et bien d'autres applications. Malgré les avancées de la vision par ordinateur, les systèmes développés sont très loin d'égaler les performances de l'œil et du cerveau humain. Pour l'être humain, voir est une tâche innée et nous ne mesurons souvent pas la difficulté pour obtenir les mêmes performances artificiellement. Notre projet se déroule dans ce cadre complexe dans le domaine d'agriculture.

2. Problématique

Notre problématique est la classification et la détection des fruits. Nous voyons après l'étude de l'état de l'art que plusieurs approches se dégagent pour la constitution d'un tel système,

l'un des problèmes est de trouver la meilleure façon pour prédire la classe des fruits automatiquement ainsi que faire la distinction entre fruit et feuille malade ou sain à partir d'images aléatoires pour fournir plus d'informations sur la qualité de rendement à l'agriculteur.

Pour se faire deux facteurs doivent être pris en considération

- Pour les réseaux de neurones convolutifs l'apprentissage est indispensable, sachant qu'il n'existe pas de bases disponibles avec des images convenables de fruits et les algorithmes d'apprentissage ont besoin d'un nombre considérables d'images pour pouvoir réaliser un entrainement et un teste fiable.
- Le choix de l'architecture du réseau pour les CNN : les modèles qui s'adaptent avec nos bases d'images,

  i.e, fixer les différents paramètres : ceci est connu comme la phase la plus difficile pour n'importe quelle approche d'apprentissage automatique.

3. Objectif du PFE

L'objectif de ce projet de fin d'études est d'élaborer un système qui imite la perception humaine de la détection et la classification des différents fruits. Il consiste à réaliser de classification binaire qui classifie les pommes et les feuilles sous deux classes (saine ou infectéem), et une autre multiple pour classer les différents fruits dans leur classe.

Une étude comparative est élaborée, afin de comparer les performances et les résultats des approches que nous avons proposées.
Concernant la détection des fruits, il s'agit de mettre au point un système capable de détecter en temps réel des fruits (on a choisi les oranges) à partir d'images provenant d'une simple caméra.

4. Structuration de document

Notre mémoire est constituée de trois chapitres :

- Dans le premier chapitre, nous présentons les travaux antérieures, les notions de base de la classification des images et leurs différents types, la description des méthodes de classification et d'apprentissage automatique, les différentes caractéristiques des réseaux de neurones à convolutions et d'autres algorithmes utilisés aves leur intérêt dans le domaine de la classification et la détection des images.
- Dans le deuxième chapitre, nous mettons en évidence le coté conceptuel de notre projet qui constitue une étape fondamentale qui précède l'implémentation, Cette dernière permet de détailler les différentes architectures à implémenter dans la phase suivante.

- Dans le troisième chapitre nous présentons les résultats obtenus et nous les discutons. Le mémoire est clôturé par une conclusion générale et des perspectives.

CHAPITRE 1
ETATS DE L'ART

II. ETAT DE L'ART

Dans ce chapitre, nous présentons les notions de bases sur les images et leur traitement, la relation de ces dernières avec l'apprentissage automatique, aussi les approches de la classification supervisée. Par la suite, nous exposons quelques travaux antérieurs qui sont en relation avec notre étude.

1. Notions de bases

D'un point de vue informatique, une image de luminance est simplement un tableau de nombres, dont chacun est la valeur de la luminance du pixel correspondant dans l'image. Il n'apparaît aisément qu'à partir de tous ces nombres, il est difficile de reconnaître non seulement une forme complexe (comme une personne) mais même des informations de plus bas niveau comme une simple forme géométrique. C'est pourquoi des descripteurs d'images sont couramment utilisés. Ceux-ci permettent de calculer pour une image donnée une information plus riche à partir d'une méthode de calcul adaptée. Dans cette section nous allons voir quelques notions de bases.

1.1.Définition d'image

Une image est une représentation planaire d'une scène ou d'un objet situé en général dans un espace tridimensionnel. Son élaboration résulte de la volonté de proposer une entité observable par l'œil humain. Ceci explique d'une part son aspect planaire et d'autre part le fait que l'information élémentaire associée à chaque point de l'image soit transcrite en niveau de gris ou en couleur.[1]

Une image est une collection d'information qui se présentait sur un support photographique qui permettait le traitement en différé d'un phénomène fugace, une analyse fine des phénomènes enregistrés et bien sûr l'archivage et l'illustration [2].

Elle est issue du contact des rayons lumineux provenant des objets formants la scène avec un capteur (caméra, scanner, rayons X, ...). Il ne s'agit en réalité que d'une représentation spatiale de la lumière.

1.2. Les différents types de format d'image

Il existe plusieurs façons de décrire les images en informatique. On présente ici les plus utilisés : [3]

- Image couleur RVB : l'image est décomposée en trois couleurs de base, similaire aux courbes de réponses des trois types de cônes de l'œil humain. L'addition des trois composantes permet de retrouver la couleur réelle.
  Le codage RVB (Rouge - Vert - bleu) ou en anglais RGB (Red - Green - Blue) est une méthode utilisée en infographie pour coder une couleur. Cette solution est utilisée en affichage des images et traitements des photos.
- Image d'intensités : c'est une matrice dans laquelle chaque élément est un réel compris entre 0 (noir) et 1 (blanc). On parle aussi d'image en niveaux de gris, car les valeurs comprises entre 0 et 1 représentent les différents niveaux de gris.
- Image binaire : une image binaire est une image pour laquelle chaque pixel ne peut avoir pour valeur que 0 ou 1. La manipulation de telles images regorge d'outils spécialisés ainsi que de théories mathématiques pour plusieurs raisons :
  - Les débuts du traitement des images numériques ne permettaient pas le traitement d'images complexes (problème de temps de calcul, d'espace mémoire disponible et qualité des périphériques de sortie). De plus, les premières applications (reconnaissance de caractères, analyse de traces laissées dans les chambres à bulles parades particules) vers 1950 s'adaptaient bien à ce type d'images.
  - Les images binaires sont un contexte simple permettant une formalisation mathématique des problèmes par des outils tels que la topologie.

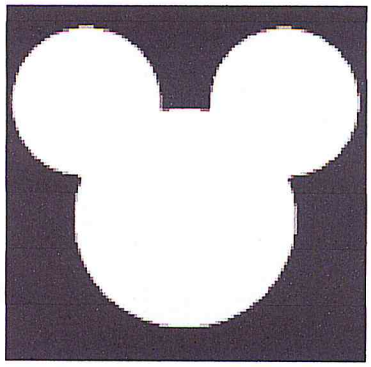


Figure 1: Image (pixels à 1) sur un fond (pixels à 0).[23]

1.3.Caractéristiques de l'image

L'image est un ensemble structuré d'information caractérisé par les paramètres suivants :

a. Pixel

Le pixel est le point élémentaire d'une image numérisée (Picture element). Un pixel peut être représenté par un seul bit (noir ou blanc) ou plus souvent par 8, 16, voire 32 bits (qui peuvent contenir des informations sur la couleur, la texture, la transparence …etc).

Le pixel correspond donc à un point de nuance de couleur qui, assemblé à beaucoup d'autres, forme une image dite matricielle.[4]

b. Niveau de gris

Une série de nuances allant du blanc pur au noir pur, utilisé pour afficher des images monochromes.

Le niveau de gris minimum est 0. Le niveau de gris maximum dépend de la profondeur de numérisation de l'image. Dans une image en niveaux de gris ou en couleur, un pixel peut prendre n'importe quelle valeur entre 0 et 255.

Dans une image couleur, le niveau de gris de chaque pixel peut être calculé à l'aide de la formule suivante:

Niveau de gris = 0.299 * composante rouge + 0.587 * composante verte + 0.114 * composante bleue[5].

Cette formule prend en compte la sensibilité des couleurs de l'œil humain rendant la présentation des niveaux de gris indépendants de la couleur et limitée uniquement à la luminosité des pixels individuels.

c. Histogramme

Un histogramme est une courbe statistique indiquant la répartition des pixels selon leur valeur.[6]

L'histogramme d'une image h(x) est la fonction qui associe `a une valeur d'intensité x le nombre de pixels dans l'image ayant cette valeur.[1]

Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci.

Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant.[7]

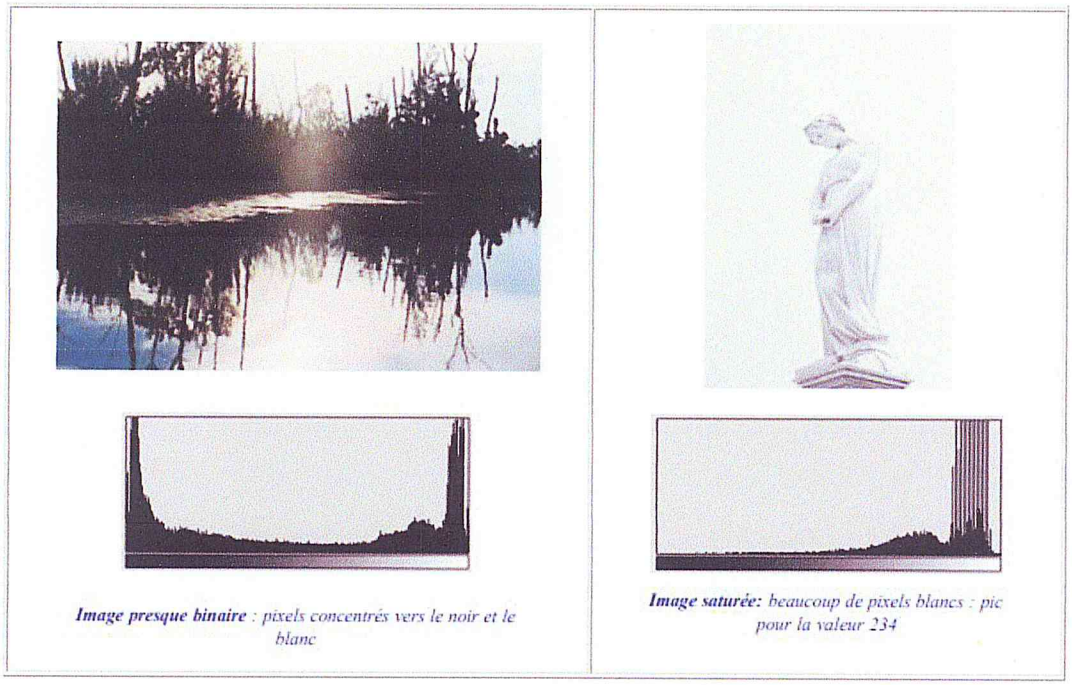


Figure 2: Histogramme[6]

d. Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C=\frac{L1-L2}{L1+L2}$$ [7]

e. Luminance

La luminance, également nommée luminance lumineuse1 ou luminance visuelle, est l'intensité lumineuse dI d'une surface élémentaire source dans une direction donnée, divisée par l'aire apparente de cette source dans cette même direction.

Autrement dis, est une grandeur correspondant à la sensation visuelle de luminosité d'une surface. Une surface très lumineuse présente une forte luminance, tandis qu'une surface parfaitement noire aurait une luminance nulle.

Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes)
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir, ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites [7].

f. L'Espace de Couleur HSV

Le système HSV (Hue, Saturation, Value) est un mode de coloration des images qui se révèle souvent plus efficace que le système classique RGB (Red, Green, Blue), notamment pour les images fractales.

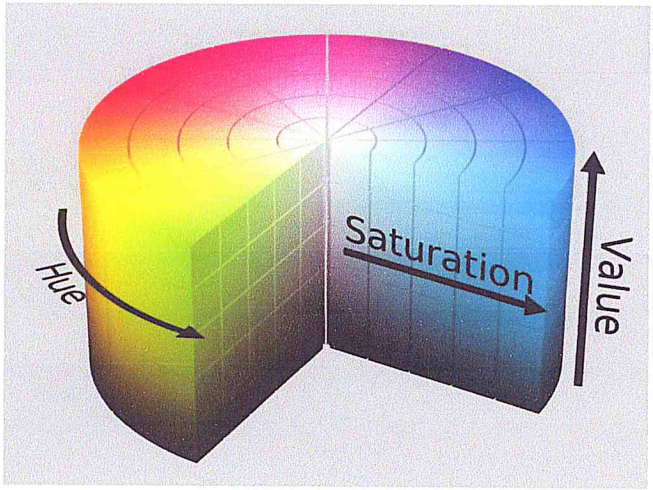


Figure 3 : Teinte, Saturation, et Valeur [8]

La Figure 3 illustre l'intervalle des teintes, H, comme un cercle représenté par des valeurs de 0 à 360.

En fixant la saturation et la luminosité (Value) à leurs valeurs maximales, on a les correspondances suivantes :

H	R	G	B	Couleur
0	255	0	0	Rouge
60	255	255	0	Jaune
120	0	255	0	Vert
180	0	255	255	Turquoise (Cyan)
240	0	0	255	Bleu
300	255	0	255	Magenta

Tableau 1: La saturation et la luminosité[8]

Dans les mêmes conditions (S = V = 1), en faisant varier H de 0 à 360° on obtient un gradient de couleurs comme montré dans la ci-dessous.

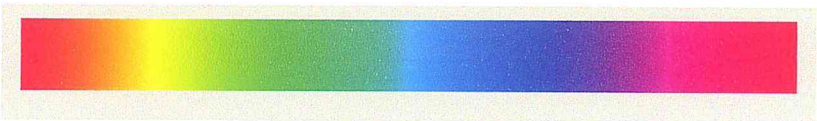


Figure 4: Variétés de H de 0 à 360

La saturation est codée par un nombre entre 0 et 1 (ou par un pourcentage). Elle traduit la vivacité de la couleur. Les fortes valeurs correspondent à des couleurs vives, les valeurs intermédiaires à des tons pastels. Aux très faibles valeurs, on n'a plus que du gris plus ou moins foncé suivant la luminosité. En prenant comme exemple la couleur rouge (H = 0), en

fixant V = 1 (luminosité maximale) et en faisant varier S de 0 à 1 on obtient un gradient qui va du blanc jusqu'au rouge.

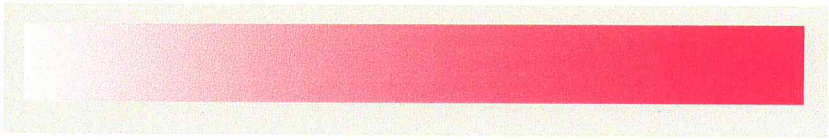


Figure 5: Variété de S entre 0 et 1

La luminosité (Value) se lit sur la coordonnée verticale, tout comme la saturation, elle s'exprime par un nombre entre 0 et 1. En augmentant la luminosité, on va du noir vers la couleur spécifiée par H. En gardant la couleur rouge (H = 0), en fixant S = 1 (saturation maximale) et en faisant varier V de 0 à 1 on obtient un gradient qui va du noir jusqu'au rouge comme montré dans la ci-dessus.

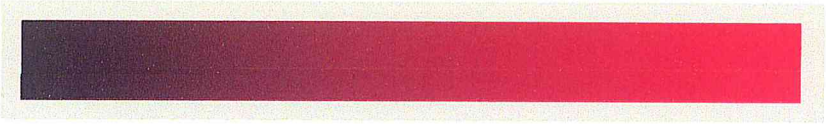


Figure 6: Variété de V entre 0 et 1

Si maintenant on se place au centre du cercle (S = 0 et H quelconque) et que l'on fasse varier V de 0 à 1 on obtient une échelle de gris

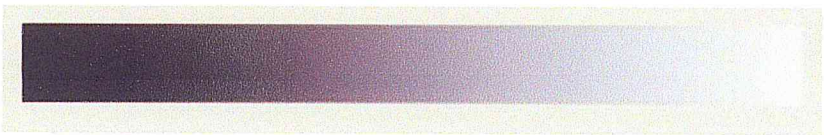


Figure 7: Variété de V entre 0 et 1 avec S = 0 et H quelconque

g. Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels).

Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image. [7]

h. Résolution

La résolution d'une image est le nombre de pixels contenus dans l'image par unité de longueur.

La résolution définit la netteté et la qualité d'une image. Plus la résolution est grande (c'est à dire plus il y a de pixels dans une longueur de 1 pouce), plus l'image est précise dans les détails.

i. Voisinage

Un voisinage de P, noté V(P), se définit comme un ensemble de pixels P0 connectés à P. On distingue deux types de voisinage :

Voisinage à 4 « 4-connexité » : On ne prend en considération que les pixels qui ont un coté commun avec le pixel considéré.

	P(i,j-1)	
P(i-1,j)	P(i,j)	P(i+1,j)
	P(i,j+1)	

Tableau 2: Voisinage à 4 [9]

Voisinage à 8 « 8-connexité » : On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré.

P(i-1,j-1)	P(i,j-1)	P(i+1,j-1)
P(i-1,j)	P(i,j)	P(i+1,j)
P(i-1,j+1)	P(i,j+1)	P(i+1,j+1)

Tableau 3: Voisinage à 8 [9]

j. Bruit

Le bruit en photo ou vidéo numériques est un défaut parasite dégradant la qualité de l'image, le bruit existe dans tout système électronique, dans le cas de la photo le signal total est la

photo finale, le signal utile est la scène réelle photographiée et le bruit est composé des pixels parasites.[10]

Il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

k. Contour

Un contour se matérialise par une rupture d'intensité dans l'image suivant une direction donnée. Ces changements d'intensités permettent de décrire des variations importantes qui se traduit par des discontinuités dans la profondeur, dans l'orientation d'une surface, dans les propriétés d'un matériau et dans l'éclairage d'une scène. Et par convention nous chercheront à représenter les contours blanc sur fond noir (la limite entre deux pixels). [11]

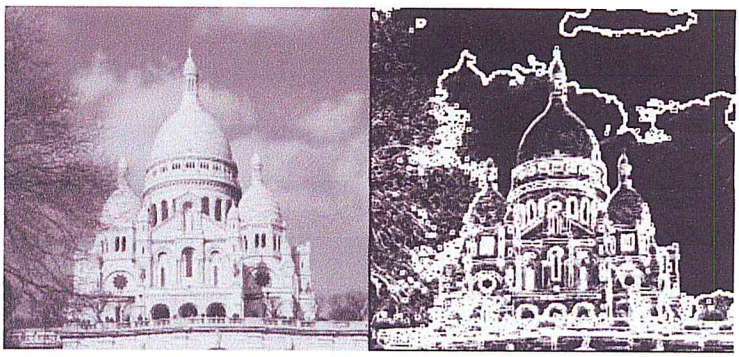


Figure 8: Contour d'image

1.4.Précision et rappel

Dans la reconnaissance de formes, pour la recherche d'information et la classification binaire, la précision (aussi appelée valeur prédictive positive) est la fraction des instances pertinentes parmi les instances récupérées, tandis que le rappel (aussi appelé sensibilité) est la fraction des instances pertinentes récupérées sur le total quantité d'instances pertinentes. La précision et le rappel sont donc basés sur une compréhension et une mesure de la pertinence.

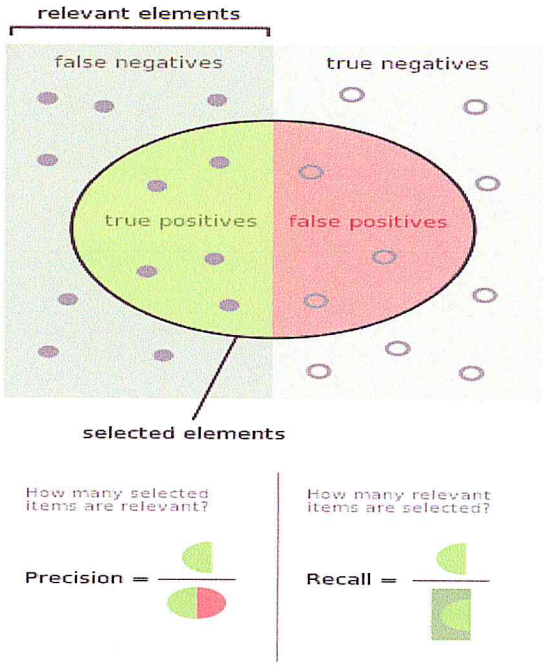


Figure 9: Précision et rappel [12]

En termes simples, la haute précision signifie qu'un algorithme a renvoyé des résultats significativement plus pertinents que des résultats non pertinents, alors qu'un rappel élevé signifie qu'un algorithme a renvoyé la plupart des résultats pertinents [13]

1.5.F-score

Dans l'analyse statistique de la classification binaire, le score F1 (également F-score ou F-mesure) est une mesure de la précision d'un teste. Il considère à la fois la précision p et le rappel r du teste pour calculer le score: p est le nombre de résultats positifs corrects divisé par le nombre de tous les résultats positifs renvoyés par le classificateur, et r est le nombre de résultats positifs corrects divisé par le nombre de tous les échantillons pertinents (tous les échantillons qui auraient dû être identifiés comme positifs). Le score F1 est la moyenne harmonique de la précision et du rappel, où un score F1 atteint sa meilleure valeur à 1 (précision parfaite et rappel) et le pire à 0.

$$F_1 = 2.\frac{Précision*Rappel}{Précision+Rappel}$$ [14]

Prenons l'exemple d'un classifieur binaire, c'est-à-dire, qui prédit 2 classes notées classe 0 et classe 1.

Pour mesurer les performances de ce classifieur, il est d'usage de distinguer 4 types d'éléments classés pour la classe voulue :

1. Vrai positif VP : Élément de la classe 1 correctement prédit
2. Vrai négatif VN : Élément de la classe 0 correctement prédit
3. Faux positif FP : Élément de la classe 1 mal prédit
4. Faux négatif FN : Élément de la classe 0 mal prédit

Ces informations peuvent être rassemblés et visualisés sous forme de tableau dans une matrice de confusion. Dans le cas d'un classifieur binaire, on obtient :

		Classe prédite	
		Classe 0	Classe 1
Classe réelle	Classe 0	VN	FN
	Classe 1	FP	VP

Figure 10 Performance de classifieur [12]

En particulier, si la matrice de confusion est diagonale, le classifieur est parfait.

Notons que la matrice de confusion est aussi généralisable lorsqu'il y a $k > 2$ classes à prédire. Il est possible de calculer plusieurs indicateurs résumant la matrice de confusion. Par exemple si nous souhaitons rendre compte de la qualité de la prédiction sur la classe 1, on définit :

Précision. Proportion d'éléments bien classés pour une classe donnée :

$$\boldsymbol{Précision_{de\ la\ classe\ 1}} = \frac{\boldsymbol{VP}}{\boldsymbol{VP+FP}} \text{ [12]}$$

Rappel. Proportion d'éléments bien classés par rapport au nombre d'éléments de la classe à prédire :

$$\boldsymbol{Rappel_{de\ la\ classe\ 1}} = \frac{\boldsymbol{VP}}{\boldsymbol{VP+FN}} \quad \text{[12]}$$

F-mesure. Mesure de compromis entre précision et rappel :

$$\boldsymbol{F-mesure_{de\ la\ classe\ 1}} = \frac{\boldsymbol{2*(Précision*Rappel)}}{\boldsymbol{Précision+Rappel}} \text{ [12]}$$

Il est possible de calculer tous ces indicateurs pour chaque classe. La moyenne sur chaque classe de ces indicateurs donne des indicateurs globaux sur la qualité du classifieur.

$$\textbf{Précision} = \frac{1}{k}\sum_{i=1}^{k}\frac{VP_i}{VP_i+FP_i}$$

$$\textbf{Rappel} = \frac{1}{k}\sum_{i=1}^{k}\frac{VP_i}{VP_i+FP_i}|$$

$$\textbf{F - mesure} = \frac{2*(Précision*Rappel)}{Précision+Rappel}$$ [12]

1.6.Matrice de confusion

Dans le domaine de l'apprentissage automatique et en particulier du problème de la classification statistique, une matrice de confusion, également connue sous le nom de matrice d'erreur[15] est généralement utilisée comme méthode quantitative de caractérisation de la précision de la classification des images. C'est un tableau qui montre la correspondance entre le résultat de la classification et une image de référence.

		Reference image			
		A	B	C	Total
Classified image	a	37	3	7	∑a = 47
	b	9	25	5	∑b = 39
	c	11	2	43	∑c = 56
	Total	∑A = 57	∑B = 30	∑C = 55	N = 142

Tableau 4: Matrice de confusion[16]

Une illustration de la matrice de confusion est présentée dans le tableau 4. Chaque ligne de la matrice représente les instances d'une classe prédite tandis que chaque colonne représente les instances d'une classe réelle (ou vice versa)[13]. Les cellules de la table montrent le nombre de pixels pour toutes les corrélations possibles entre la vérité de terrain et l'image classifiée.

Dans le tableau 4, les éléments diagonaux de la matrice sont surlignés en bleu. Les cellules diagonales contiennent le nombre de pixels correctement identifiés. Si nous divisons la somme de ces pixels par le nombre total de pixels, nous obtiendrons la précision globale de la classification (OvAc). Pour la matrice de confusion présentée dans le tableau 4, cet indice sera égal à:

$$OvAc = (aA + bB + cC) / N = (37 + 25 + 43) / 142 \approx 0,74$$

2. Approches de classifications

La classification est une discipline reliée de près ou de loin à plusieurs domaines, elle est connue aussi sous différents noms (classification, *Clustering*, segmentation,. . .) selon les objets qu'elle traite et les objectifs qu'elle vise à atteindre.

Les méthodes de classification sont des algorithmes qui permettent de classer des objets observés dans des classes (appelées clusters), les objets d'une même classe doivent être "similaires" et les objets de deux classes différentes doivent être "distincts". Sont séparées en deux grandes catégories : les méthodes de classification supervisée et les méthodes de classification non supervisée.

2.1. Apprentissage non supervisées

L'apprentissage non supervisé consiste à apprendre sans superviseur et à représenter un nuage des points d'un espace quelconque en un ensemble de groupes appelé Cluster.

Un «Cluster» est une collection d'objets qui sont «similaires» entre eux et qui sont «dissemblables » par rapport aux objets appartenant à d'autres groupes.

La méthode non supervisée consiste à extraire des classes ou groupes d'individus présentant des caractéristiques communes. La qualité d'une méthode de classification est mesurée par sa capacité à découvrir certains ou tous les motifs cachés. [17]

2.2. Approches supervisées

A partir d'un échantillon fini d'objets étiquetés ou classés, on désire construire une fonction de prédiction capable d'étiqueter ou classer au mieux de nouveau objets (ne faisant pas partie de l'échantillon initial).

En principe son objectif est de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. On utilise aussi

un deuxième échantillon appelé souvent 'teste' ou validation pour faire un teste de fiabilité des règles.[18]

Dans le cadre de notre travail nous nous intéressons à la classification supervisée. Dans la section si dessous nous présentons les principales approches.

2.3. Approches de la classification supervisée

Nous allons citer ci-dessous les différents algorithmes utilisés.

2.3.1. Séparateur de Vaste Marges (SVM)

Séparateur de Vaste Marges ou (Suport Vector Machine) en anglais, est un algorithme d'apprentissage supervisé basé sur le noyau qui classe les données en deux classes ou plus. Au cours de la phase d'apprentissage, SVM construit un modèle, cartographie la limite de décision pour chaque classe et spécifie l'hyperplan qui sépare les différentes classes. Augmenter la distance entre les classes en augmentant la marge de l'hyperplan permet d'augmenter l'exactitude de la classification.
Son principe est de trouver un classificateur, ou une fonction de discrimination* , dont la capacité de généralisation (qualité de prévision) est la plus grande possible [3].

Pour un ensemble de données constitué d'un jeu de fonctions et d'un jeu d'étiquettes, un classificateur SVM crée un modèle pour prévoir des classes pour de nouveaux exemples. Il affecte de nouveaux exemples (points de données) à l'une des classes. S'il n'y a que 2 classes, il peut être appelé en tant que classificateur SVM binaire. Il existe deux types de classificateurs SVM:

- Classificateur SVM linéaire
- Classificateur SVM non linéaire

2.3.1.1. Classificateur SVM linéaire

Dans le modèle de classificateur linéaire, nous avons supposé que des exemples d'apprentissage étaient tracés dans l'espace. Ces points de données devraient être séparés par un écart apparent. Il prédit un hyperplan droit divisant 2 classes. L'objectif principal lors du dessin de l'hyperplan est de maximiser la distance entre l'hyperplan et le point de données le plus proche de l'une ou l'autre classe. L'hyperplan dessiné appelé comme un hyperplan à

marge maximale. Dans le cas où le problème est linéairement séparable, le choix de l'hyperplan séparateur n'est pas évident. Il existe en effet une infinité d'hyperplans séparateurs, dont les performances en phase d'apprentissage sont identiques, mais dont les performances en phase de teste peuvent être très différentes. Pour résoudre ce problème, il a été montré [19], qu'il existe un unique hyperplan optimal, défini comme l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur. Il existe des raisons théoriques à ce choix. Vapnik a montré [19]que la capacité des classes d'hyperplans séparateurs diminue lorsque leur marge augmente.

2.3.1.2. Classificateur non linéaire SVM

Dans le monde réel, notre ensemble de données est généralement dispersé dans une certaine mesure. Pour résoudre ce problème, la séparation des données en différentes classes sur la base d'un hyperplan linéaire ne peut pas être considérée comme un bon choix. Pour cela, Vapnik a suggéré de créer des classificateurs non linéaires en appliquant l'astuce du noyau aux hyperplans à marge maximale, qui doit respecter les conditions du théorème de Mercer[20]. Dans la classification SVM non linéaire, les points de données sont tracés dans un espace dimensionnel supérieur.

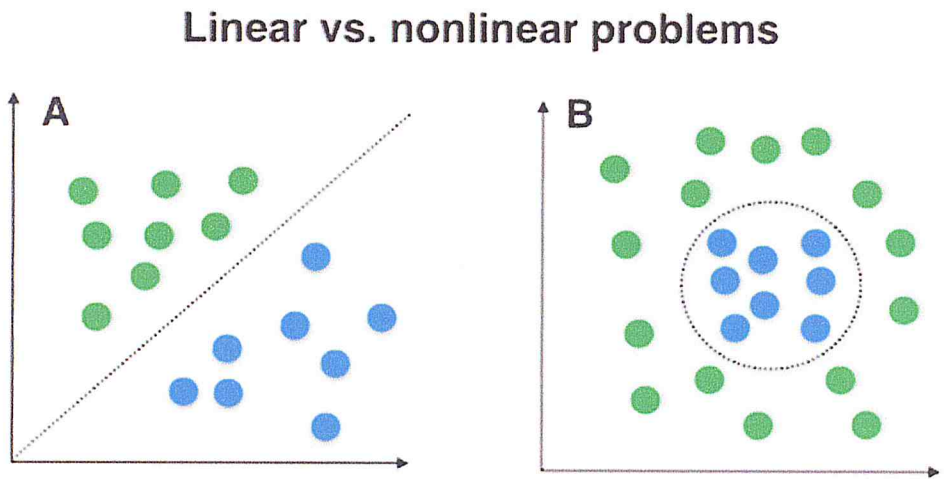


Figure 11: classificateur SVM linéaire et non linéaire[21]

2.3.1.3. **Principe de la technique SVM pour la classification binaire**

Cette technique est une méthode de classification à deux classes qui tente de séparer les exemples positifs des exemples négatifs dans l'ensemble des exemples. La méthode cherche

alors l'hyperplan qui sépare les exemples positifs des exemples négatifs, en garantissant que la marge entre le plus proche des positifs et des négatifs soit maximale. Cela garantit une généralisation du principe car de nouveaux exemples pourront ne pas être trop similaires à ceux utilisés pour trouver l'hyperplan mais être situés d'un coté ou l'autre de la frontière. L'intérêt de cette méthode est la sélection de vecteurs supports qui représentent les vecteurs discriminant grâce auxquels est déterminé l'hyperplan. Les exemples utilisés lors de la recherche de l'hyperplan ne sont alors plus utiles et seuls ces vecteurs supports sont utilisés pour classer un nouveau cas, ce qui peut être considéré comme un avantage pour cette méthode.

- Avantages
  - — Il permet de traiter des problèmes de classification non linéaire complexe.
  - — Les SVM constituent une alternative aux réseaux de neurones car plus faciles à entraîner.

2.3.2. Algorithme K plus proches voisins

L'algorithme K plus proches voisins (K-NN) est un algorithme de classification supervisée, ont été utilisés depuis 1970 dans de nombreuses applications telles que l'estimation statistique et la reconnaissance des formes ...etc. Son principe est similaire à l'expression **"Dites-moi qui sont vos voisins, et je vous dirai qui vous êtes"** A l'intérieur, cet algorithme se base simplement sur la distance entre les vecteurs de caractéristiques, tout comme la construction d'un moteur de recherche d'images seulement cette fois, nous avons les étiquettes associées à chaque image pour pouvoir prédire et retourner une catégorie réelle pour l'image.

Autrement dit, l'algorithme K-NN classe les points de données inconnus en trouvant la classe la plus commune parmi les exemples les plus proches de k. Chaque point de données dans les k exemples les plus proches émet un vote et la catégorie avec le plus de votes gagne [3].

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad [22]$$

Notation et Algorithme :

k-Nearest Neighbor
Classify (X, Y, *x*) // X: training data, Y: class labels of X, *x*: unknown sample
for $i = 1$ **to** m **do**
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set I containing indices for the k smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$

Figure 12: Étapes de L'algorithme KNN[23]

2.3.3. Réseaux de neurones multi couche

Un réseau de neurones multi couche ou est un système composé de neurones, généralement répartis en plusieurs couches connectées entre elles, la sortie d'une couche correspond à l'entrée de la suivante. Les connexions entre les différents nœuds ont des valeurs numériques, appelées pondérations, et en modifiant ces valeurs de façon systématique, le réseau peut éventuellement approximer la fonction désirée.

La dernière couche calcule les probabilités finales en utilisant pour fonction d'activation la fonction logistique (classification binaire) ou la fonction softmax (classification multi-classes)

Une fonction de perte (loss function) est associée à la couche finale pour calculer l'erreur de classification.

Les valeurs des poids des couches sont appris par rétropropagation du gradient : on calcule progressivement (pour chaque couche, en partant de la fin du réseau) les paramètres qui minimisent la fonction de perte.

Cet empilement de couches définit la sortie finale du réseau comme le résultat d'une fonction différentiable de l'entrée

Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires. Les principaux réseaux se distinguent par l'organisation du graphe (en couches, complets. . .), c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones

(leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques...[24]

Le perceptron multicouches (multilayer perceptron MLP) est en fait un type de réseau neuronale formel qui s'organise en couches, l'information circule dans une seule direction c'est de la couche d'entré vers la couche de sortie en passant par une ou plusieurs couches ; c'est ce qu'on appelle un réseau à propagation directe. Les neurones de la dernière couche sont considérés comme étant les sorties du système global.

Les MLP sont largement utilisés pour la reconnaissance, la classification, l'approximation et la prédiction des patterns.

- Avantages :

— Capacité à découvrir les dépendances par lui même.

— Résistance aux bruits[25].

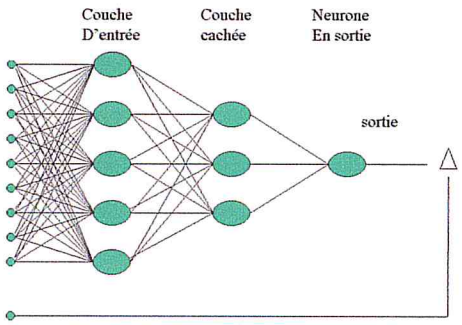


Figure 13: Modèle d'MLP [25]

2.3.4. Les réseaux de neurones convolutifs

Les réseaux de neurones à convolutions ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé : ils reçoivent des images en entrée, détectent les features de chacune d'entre elles, puis entraînent un classifieur dessus.

Cependant, les caractéristiques sont apprises automatiquement, les CNN (Convolutional Neural Networks) réalisent eux-mêmes tout le boulot fastidieux d'extraction et description de caractéristiques :

Lors de la phase d'entraînement, l'erreur de classification est minimisée afin d'optimiser les paramètres du classifieur et les caractéristiques. De plus, l'architecture spécifique du réseau permet d'extraire des caractéristiques de différentes complexités, des plus simples au plus sophistiquées. L'extraction et la hiérarchisation automatiques des caractéristiques, qui s'adaptent au problème donné, constituent une des forces des réseaux de neurones à convolutions : plus besoin d'implémenter un algorithme d'extraction "à la main". [26]

Précédemment aux techniques d'apprentissage supervisé, les réseaux de neurones convolutifs apprennent les caractéristiques de chaque image. C'est là que réside leur force : les réseaux font tout le boulot d'extraction de caractéristiques automatiquement, contrairement aux techniques d'apprentissage.[26]

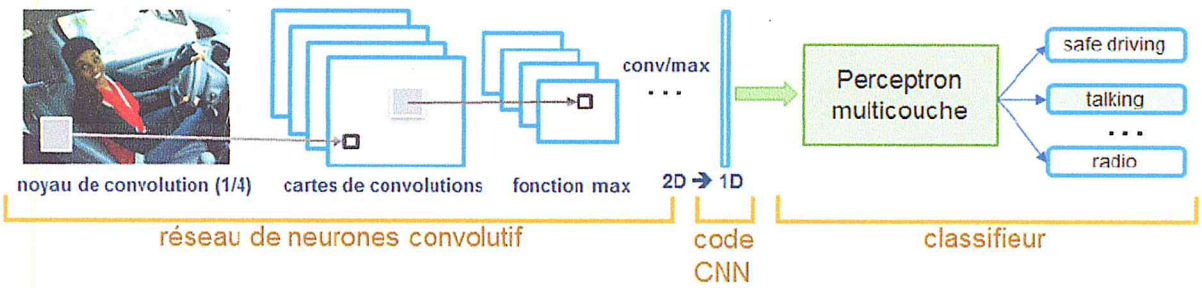


Figure 14: Architecture standard d'un réseau de neurone à convolutions[27]

Ils ont de larges applications dans la reconnaissance de l'image et de la vidéo, les systèmes de recommandations[28] et le traitement du langage naturel[29]

2.3.4.1. Architecture de réseaux de neurone à convolution

Les réseaux de neurones à convolutions comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].[27]

La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.[27]

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

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories.

Cet assemblage de couches de traitement forme l'architecture de réseau de neurones à convolutions:

- La couche de convolution (CONV) qui traite les données d'un champ récepteur ;
- La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage) ;
- La couche de correction (ReLU), souvent appelée par abus « ReLU » en référence à la fonction d'activation (Unité de rectification linéaire) ;
- La couche « entièrement connectée » (FC), qui est une couche de type perceptron ;
- La couche de perte (LOSS).

Ces opérations sont les éléments de base de chaque réseau de neurones à convolution, de sorte que la compréhension de leur fonctionnement est une étape importante pour développer une bonne compréhension des ConvNets. Nous essaierons de comprendre l'intuition derrière chacune de ces opérations ci-dessous.

2.3.4.2. Couche de convolution (CONV)

La couche de convolution est la composante clé des réseaux de neurones à convolutions, et constitue toujours au moins leur première couche.

Quand on lui présente une nouvelle image, le CNN ne sait pas exactement si les caractéristiques seront présentes dans l'image ou où elles pourraient être, il cherche donc à les trouver dans toute l'image et dans n'importe quelle position.

En calculant dans toute l'image si une caractéristique est présente, nous faisons un filtrage. Les mathématiques que nous utilisons pour réaliser cette opération sont appelés une convolution.

Pour calculer la correspondance entre une caractéristique et une sous-partie de l'image, il suffit de multiplier chaque pixel de la caractéristique par la valeur que ce même pixel contient dans l'image. Ensuite, on additionne les réponses et divise le résultat par le nombre total de pixels de la caractéristique. Si les 2 pixels sont blancs (de valeur 1) alors 1 * 1 = 1. Si les deux sont noires, alors (-1) * (-1) = 1. Dans tous les cas, chaque pixel correspondant a pour résultat 1. De manière similaire, chaque décalage donne -1. [30]

Si tous les pixels dans une caractéristique correspondent, alors leur addition puis leur division par le nombre total de pixels donne 1. De la même manière, si aucun des pixels de la caractéristique ne correspond à la sous-partie de l'image, alors la réponse est -1.

Pour compléter on prend alors le résultat de chaque convolution et créé avec un nouveau tableau de 2 Dimensions, basé sur où dans l'image le patch se trouve. Cette map des correspondances est aussi une version filtrée de l'image d'origine. C'est une map indiquant où la caractéristique a été trouvée dans l'image. Les valeurs proches de 1 montrent une forte correspondance, celles proches de -1 montrent une forte correspondance pour le négatif photographique de la fonctionnalité et les valeurs proches de 0 ne correspondent à aucun type. [30]

Pour simplifier les choses, nous allons parcourir une image qui se situe entre 1 et 0. Ceci est basé sur une image en noir et blanc donc 0 étant blanc, 1 étant noir.

Dans cet exemple, notre détecteur de caractéristiques a une taille de 3x3. Les détecteurs de caractéristiques peuvent être plus grands ou plus petits et peuvent également être connus sous le nom de noyau ou de filtre. Le détecteur de caractéristiques fait correspondre les 0 et les 1 aux carrés de l'image d'entrée pour ensuite les stocker dans une carte de caractéristiques. Le détecteur traverse chaque colonne, 1 colonne à la fois, c'est ce qu'on appelle une foulée. Dans cet exemple, la foulée est de 1px. La foulée peut être modifiée et est couramment utilisée avec 2px.

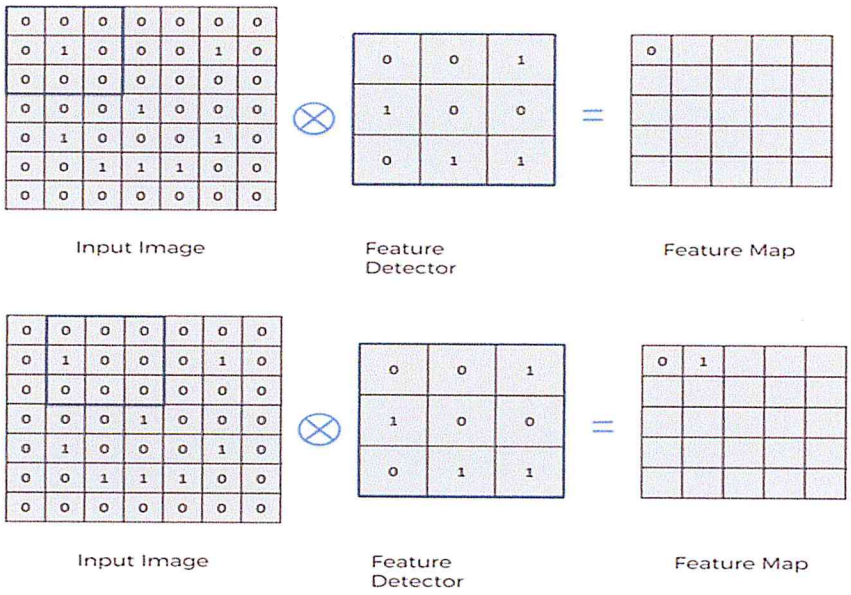


Figure 15: filtres de convolutions [54]

Une fois que le détecteur de caractéristiques a fini de parcourir toute l'image, la carte des caractéristiques est terminée. Voici l'exemple complet:

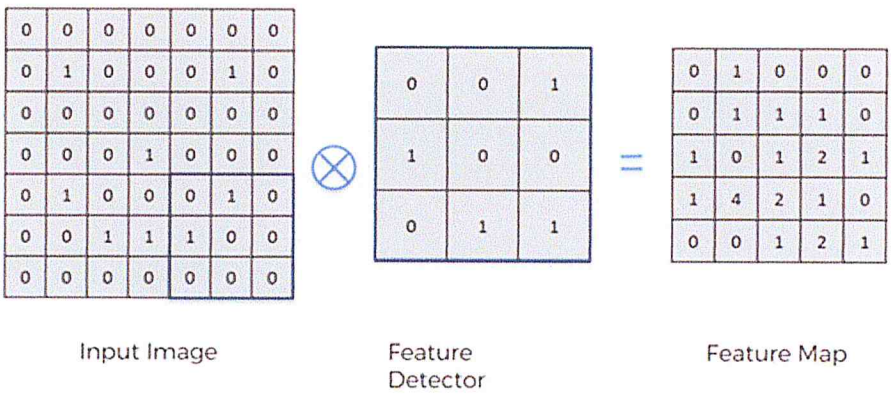


Figure 16: Résultats de convolutions [54]

Une opération de convolution est signifiée par un X dans un cercle. Une carte d'entités peut également être appelée carte d'activation. Le but du détecteur de caractéristiques est le suivant:

Compressez l'image dans une carte de caractéristiques, ce qui la rend plus facile à traiter (plus vous avez de foulées, plus la carte des caractéristiques est petite)

Cela perd certaines informations mais le but des détecteurs de fonctions est de se concentrer sur les parties de l'image qui sont intégrales.

Par exemple le détecteur de caractéristiques a un certain motif sur lui, le nombre le plus élevé dans la carte de caractéristiques est quand ce motif correspond.

Cela nous aide à préserver les caractéristiques de l'image.

Une couche à convolution consiste en plusieurs mappages de fonctions utilisant différentes caractéristiques de l'image. Grâce à la formation du réseau neuronal, il décide quelles caractéristiques sont importantes.

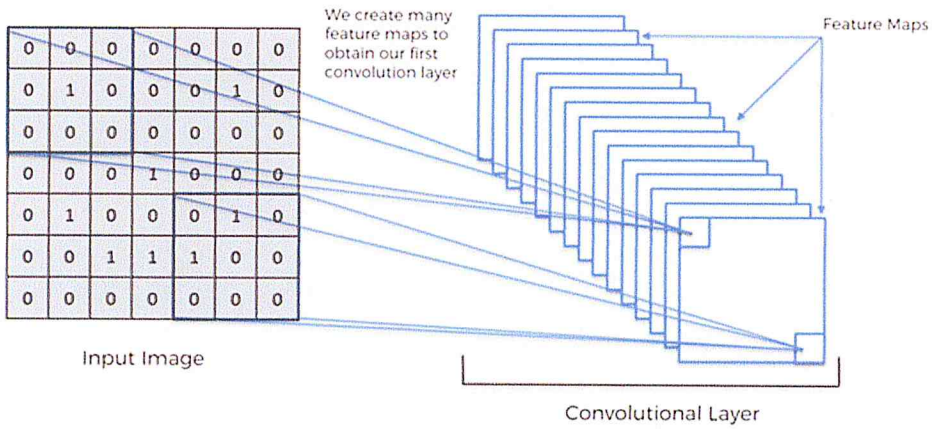


Figure 17: Couche de convolutions 3 [54]

Certains types de fonctionnalités comprennent les éléments suivants:

La carte de caractéristiques de chaque entité est sur la gauche et la représentation de l'image est sur la droite.

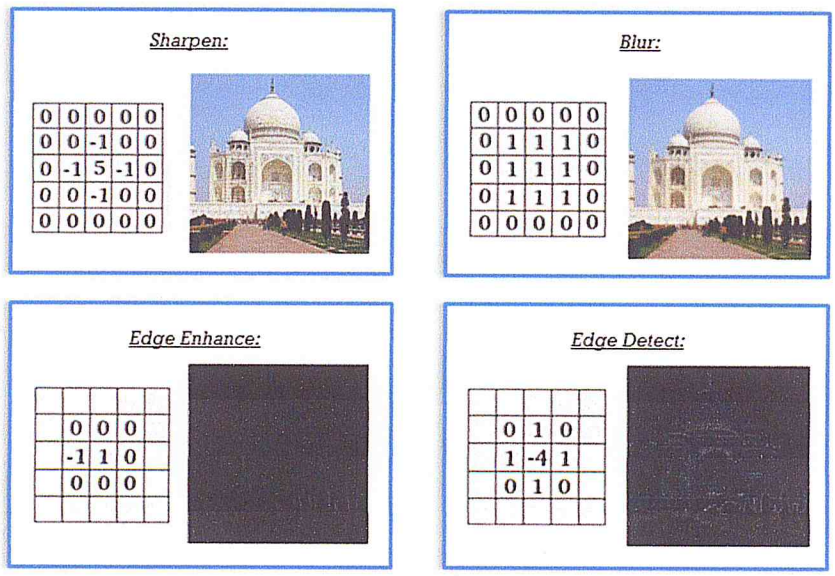


Figure 18: différents filtres de convolution [54]

2.3.4.3. Couche de pooling (POOL)

Un autre outil très puissant utilisé par les CNNs s'appelle le Pooling. Le Pooling est une méthode permettant de prendre une large image et de réduire la taille tout en préservant les informations les plus importantes qu'elle contient. Elle est aussi dite la couche de regroupement.

Intuitivement, une couche de pooling semble créer une perte d'informations. Cependant, associée à une couche de Padding, qui ajoute des pixels à zéro autour de l'image, elle peut, au contraire permettre d'éliminer ou de limiter l'impact de ces rajouts. Le tout, en gardant un degré d'information optimal. Par exemple, une étape de maxpooling éliminera l'ensemble des pixels à zéro précédemment ajoutés.

Néanmoins, son intérêt principal réside dans la prévention de l'*Overfitting* et la réduction du coût de calcul, en réduisant le nombre de paramètres pour l'apprentissage.

L'output aura le même nombre d'images mais chaque image aura un nombre inférieur de pixels. Cela permettra ainsi de diminuer la charge de calculs (l'image n'a plus qu'un quart du nombre de ses pixels de départ).

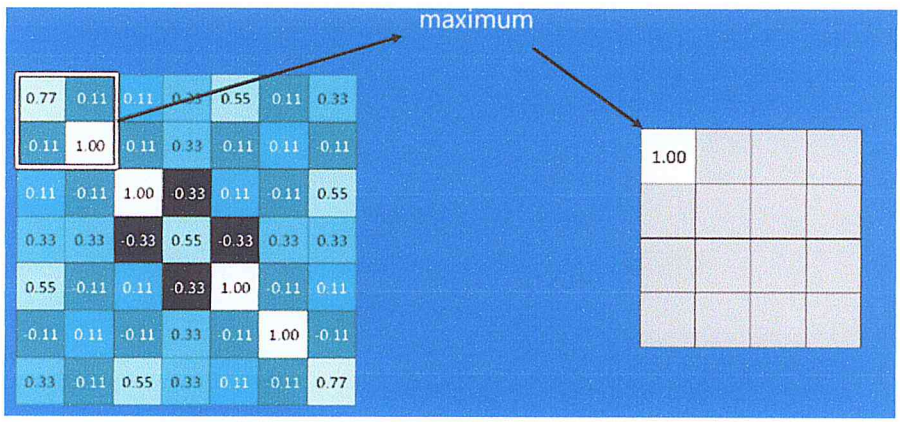


Figure 19: Illustration de la couche de Pooling [30]

Le *Padding* : est l'ajout de pixels artificiels autour de l'image, afin de faire des convolutions complètes. La taille des images n'étant pas nécessairement un multiple de la taille du stride, cela conduit à ne pas perdre d'information.

2.3.4.4. Couche de correction (RELU)

Aussi dite Unités Rectifié Linéaire, son rôle est de remplacer toutes les valeurs négatives dans un pixel par un 0. Ainsi, on permet au CNN de rester en bonne santé (mathématiquement parlant) en empêchant les valeurs apprises de rester coincer autour de 0 ou d'exploser vers l'infinie.

Le résultat d'une couche ReLU est de la même taille que ce qui lui est passé en entrée, avec simplement toutes les valeurs négatives éliminées.[30]

Nous appliquons un redresseur pour augmenter la non-linéarité dans notre CNN. Le redresseur agit comme la fonction qui brise la linéarité. Nous voulons augmenter la non-linéarité pour que les images elles-mêmes soient hautement non-linéaires.

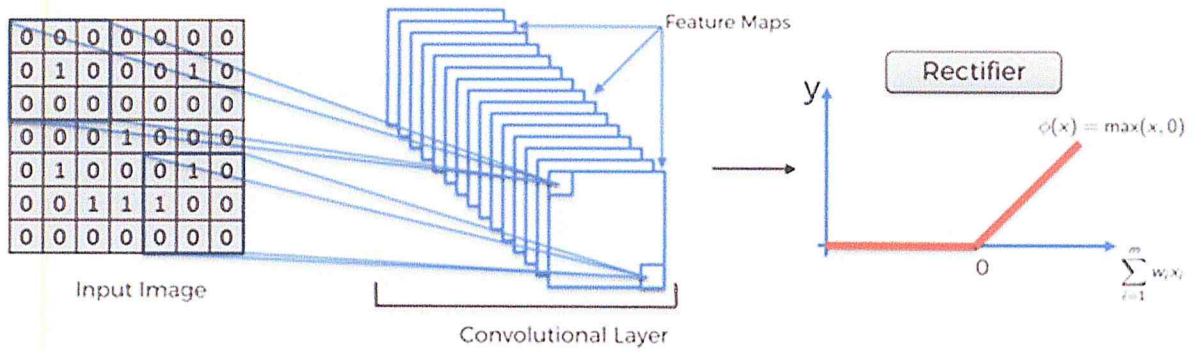


Figure 20: La fonction ReLU [54]

En termes de linéaire, il se réfère à l'ombrage. Par exemple: il va du blanc au noir dans les étapes ombragées.

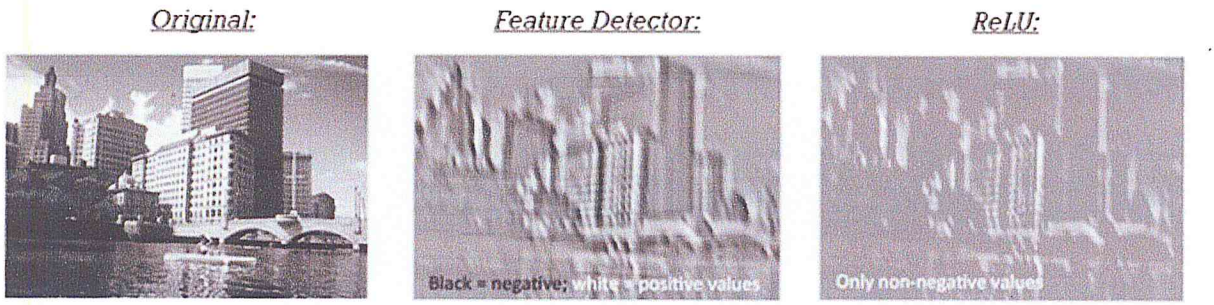


Figure 21: Etapes de la fonction ReLU [54]

2.3.4.5. Aplanissement (Flatten)

En utilisant la carte des entités regroupées, nous l'aplatissons en une colonne. En partant de la rangée du haut, de gauche à droite.

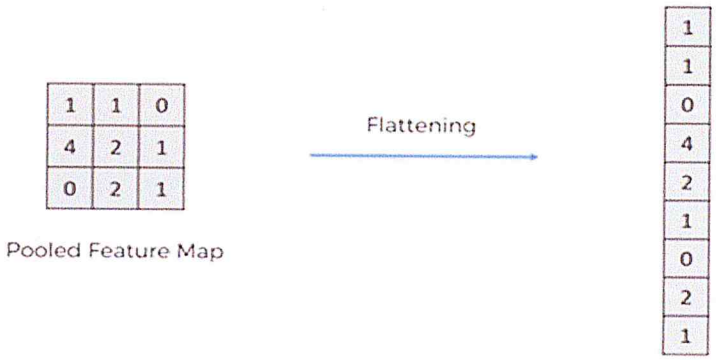


Figure 22: Applanissement d'une Map [54]

Ceci est fait pour que nous puissions l'intégrer dans un réseau de neurones artificiels pour un traitement ultérieur.

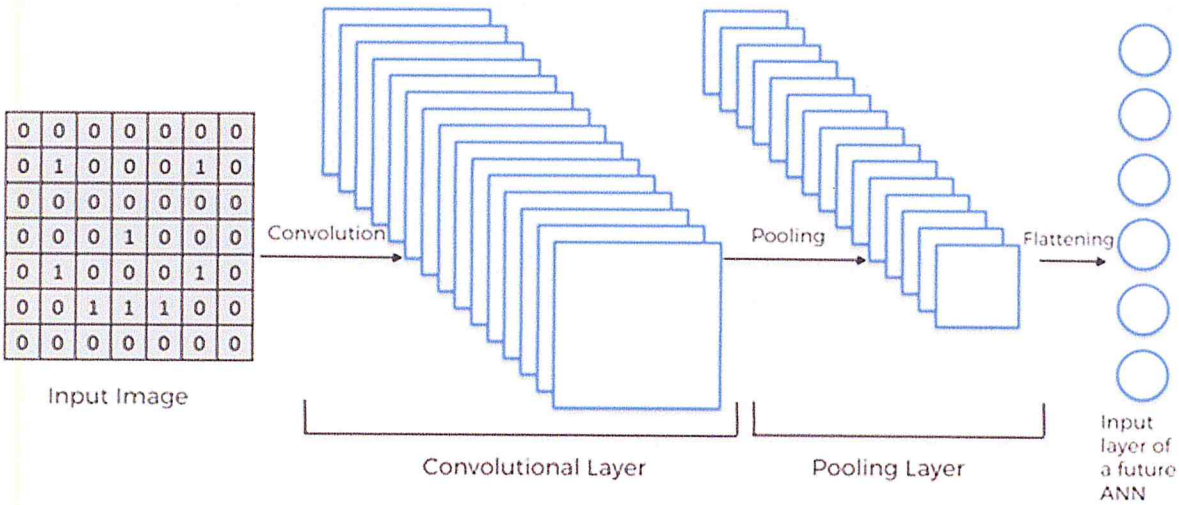


Figure 23: architecture globale [54]

2.3.4.6. Couche entièrement connectée (fully connected)

Après avoir utilisé des couches de convolution pour extraire les caractéristiques spatiales d'une image, nous appliquons des couches entièrement connectées pour la classification finale. Premièrement, nous aplatissons la sortie des couches de convolution. Par exemple, si les cartes d'entités finales ont une dimension de 4x4x512, nous l'aplatirons en un tableau de

4096 éléments. Nous appliquons 2 autres calques cachés ici avant que nous effectuions le classement final. Les techniques nécessaires ne sont pas différentes d'un réseau FC dans l'apprentissage en profondeur.[55]

La couche entièrement connectée constitue toujours la dernière couche d'un réseau de neurones, les neurones dans cette couche ont des connections vers tout les neurones du couche précédente.

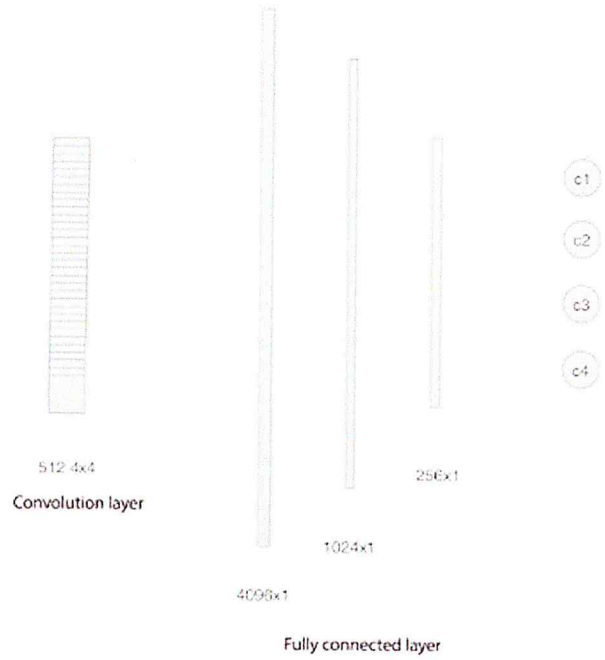


Figure 24: Couche entièrement connectée [54]

Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

2.3.4.7. Couche de perte (LOSS)

C'est la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « Softmax » permet de calculer la distribution de probabilités sur les classes de sortie.

Son rôle est de spécifier comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel.

2.3.4.8. SoftMax

La fonction SoftMax est utilisée pour fournir des valeurs de sortie avec une probabilité entre chacune d'elles. Par exemple: il y a une image d'un chien qui a été passée à travers un CNN, la machine décide que l'image est 0,95 susceptible d'être un chien et 0,05 susceptible d'être un chat.

$$f_j(z) = \frac{e^{zj}}{\sum_k e^{zk}} \quad [54]$$

La valeur d'origine est écrasée en valeurs beaucoup plus petites qui s'ajoutent toutes deux jusqu'à 1 pour permettre à la machine de fournir une probabilité appropriée de chaque image.

2.3.4.9. Sur apprentissage

Fondamentalement, sur apprentissage fait référence à un modèle qui modélise les «données d'entraînement» trop bien. Le sur apprentissage se produit lorsqu'un modèle apprend le détail et le bruit dans les données d'apprentissage dans la mesure où il a un impact négatif sur la performance du modèle sur les nouvelles données.

Sur apprentissage se produit chaque fois qu'un modèle apprend de modèles qui sont présents dans les données de formation, mais ne le font pas refléter le processus de génération de données. Voir plus qu'il n'y en a réellement (Une sorte d'hallucination des données).

On doit distinguer les données d'Entrainement des données du Teste car Si le classificateur choisi est ensuite testé sur de nouvelles données, il n'est souvent pas mieux que deviner au hasard.

Il n'y a pas de réponse exacte à cette question. Parfois, nous sommes presque influencés par notre modèle en voyant le coût de la formation diminuer progressivement. Mais considérons un scénario, ci-dessous est un graphique de changement de coût que le réseau apprend

En voyant le graphique, il faut seulement dire que notre modèle apprend comme le coût diminue constamment. Mais laissez-moi vous montrer comment la précision des testes change avec le d'époque, s'il vous plaît suivez la figure suivante.

Figure 25: précision d'entrainement

En voyant le graphique, on peut que notre modèle apprend comme le coût diminue constamment. Mais regardons comment la précision des testes change avec les époques,

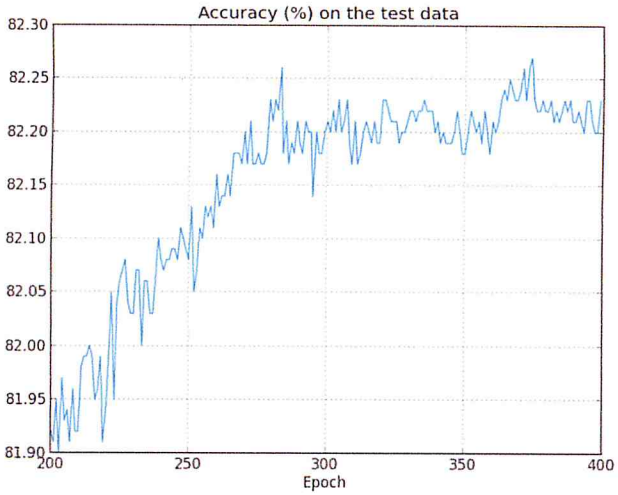


Figure 26: précision de teste

En fait, cela fait partie d'une stratégie plus générale, qui consiste à utiliser les données de validation pour évaluer différents choix d'hyper-paramètres tels que le nombre d'époques à suivre, le taux d'apprentissage, le meilleur réseau l'architecture, et ainsi de suite. Nous utilisons ces évaluations pour trouver et définir de bonnes valeurs pour les hyper-paramètres.

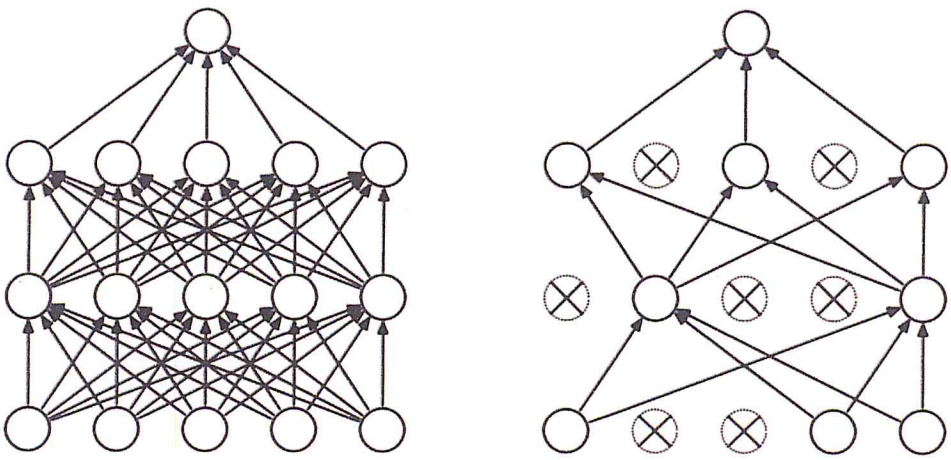
Comment cela va nous aider à prévenir les sur apprentissage. Pour comprendre pourquoi, considérons que lors de la définition des hyper-paramètres, nous sommes susceptibles d'essayer plusieurs choix différents pour les hyper-paramètres. Si nous définissons les hyper-paramètres en fonction des évaluations des données de teste, il est possible que nous finissions par superposer nos hyper-paramètres aux données de teste. C'est-à-dire que nous pouvons finir par trouver des hyper-paramètres qui correspondent à des particularités particulières des

testes data, mais où les performances du réseau ne généraliseront pas à d'autres ensembles de données.

2.3.4.10. Régularisation

Un problème central du Machine Learning est de savoir comment créer un algorithme qui fonctionne bien ; non seulement sur l'ensemble d'apprentissage, mais aussi sur l'ensemble de teste. De nombreuses stratégies utilisées dans le Deep Learning sont explicitement conçues pour réduire l'erreur de teste, éventuellement au détriment d'une augmentation de l'erreur d'apprentissage. Ces stratégies sont connues sous le nom de régularisation. Nous allons nous intéresser à une technique particulièrement puissante : le *Dropout.*

Le *Dropout* est une technique où des neurones sélectionnés au hasard sont ignorés (temporairement) pendant l'apprentissage. Cela signifie que leur contribution à l'activation des neurones qui leur succède est temporairement supprimé lors de la phase de propagation et toutes les mises à jour de poids ne sont pas appliquées au neurone lors de la phase de retro propagation. Lorsque des neurones sont supprimés au hasard du réseau pendant l'apprentissage, les autres neurones devront intervenir et gérer la représentation requise pour faire des prédictions pour les neurones manquants. Lors de la phase d'apprentissage, pour chaque itération, un neurone est gardé avec une probabilité p, sinon il est supprimé. Lors de la phase de teste, tous les neurones sont gardés, nous voulons donc que les sorties des neurones au moment du teste soient identiques à leurs sorties au moment de l'apprentissage. Par exemple, dans le cas ou p = 0.5, les neurones doivent réduire de moitié leurs sorties au moment du teste pour avoir la même sortie que pendant l'apprentissage.



(a) Standard Neural Net (b) After applying dropout.

Figure 27: Illustration du *Dropout* lors de l'apprentissage (à droite) et lors du teste (à gauche)

3. Réseaux de neurones de convolution pour la détection d'objets

Au cours des dernières années, le domaine de la détection d'objets a connu d'énormes progrès, aidés par l'avènement de l'apprentissage en profondeur. La détection d'objets est la tâche d'identifier des objets dans une image et de dessiner des cadres de délimitation autour d'eux, c'est-à-dire de les localiser. C'est un problème très important dans la vision par ordinateur en raison de ses nombreuses applications de la voiture autonome à la sécurité et au suivi. Les approches antérieures de la détection d'objets ont généralement proposé des pipelines qui sont des étapes distinctes dans une séquence. Cela provoque une déconnexion entre ce que chaque étape accomplit et l'objectif final, qui dessine un cadre étroit autour des objets dans une image. Un cadre de bout en bout qui optimise l'erreur de détection d'une manière conjointe serait une meilleure solution, non seulement pour former le modèle pour une meilleure précision, mais aussi pour améliorer la vitesse de détection.

C'est là qu'intervient l'approche «Vous ne regardez qu'une seule fois» (ou YOLO).

4.YOLO

YOLO qui signifie You Only Look Once, est un algorithme de détection d'objets basé sur l'apprentissage profond développé par Joseph Redmon et Ali Farhadi à l'Université de Washington en 2016, ce n'est pas un classificateur traditionnel qui est réutilisé pour être un détecteur d'objet, mais il adopte une approche complètement différente, ne regarde qu'une seule fois l'image (d'où son nom: You Only Look Once), mais d'une manière intelligente, il divise l'image en une grille de 13 par 13 cellules comme montre dans la figure 25[32].

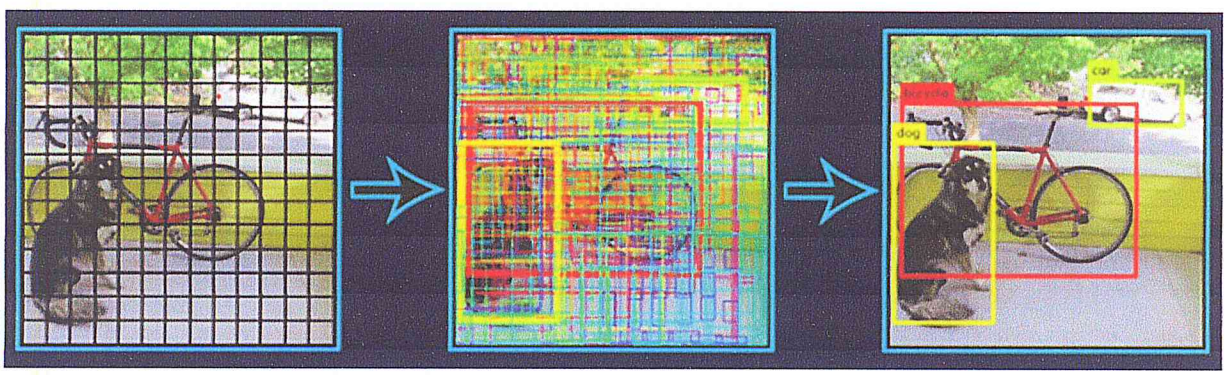


Figure 28: Yolo [32]

La façon dont cela a été fait était via un processus en deux étapes:

La première étape consistait à générer des dizaines de milliers de propositions. Ils ne sont rien mais des zones rectangulaires spécifiques sur l'image également connu sous le nom de boîtes de délimitation (bounding boxes), de ce que le système a cru être des choses semblables à des objets dans l'image. La proposition de la boîte englobante peut être soit autour d'un objet réel dans une image ou non, et filtrer ceci était l'objectif de la deuxième étape.

Dans la deuxième étape, un classificateur d'images classerait la sous-image dans la proposition de boîte englobante, et le classificateur dirait s'il s'agissait d'un type d'objet particulier ou simplement d'un non-objet ou d'un arrière-plan. Bien qu'important, ce processus en deux étapes a souffert de certaines failles telles que l'efficacité, en raison de l'immense nombre de propositions générées, et d'un manque d'optimisation conjointe sur la génération et la classification des propositions. Cela conduit à ce que chaque étape ne comprenne pas vraiment la situation dans son ensemble, mais soit réduite à son propre mini-problème et limite ainsi ses performances.

Le système YOLO passe une fois seulement dans l'image entière au système d'apprentissage en profondeur. Ensuite, il obtient toutes les boîtes englobantes ainsi que les classifications des catégories d'objets en une fois. C'est la décision de conception fondamentale de YOLO et c'est une nouvelle perspective rafraîchissante sur la tâche de détection d'objets.

4.1.Intersection sur Union (IoU)

En anglais (Intersect Over Union) est une métrique d'évaluation utilisée pour mesurer la précision d'un détecteur d'objet sur un jeu de données particulier, pour évaluer comment une sortie de détection d'objet est liée à une vérité terrain, la IoU est normalement utilisée pendant l'entrainement et les testes en comparant la boîte de délimitation donnée lors de la prédiction avec la réalité.

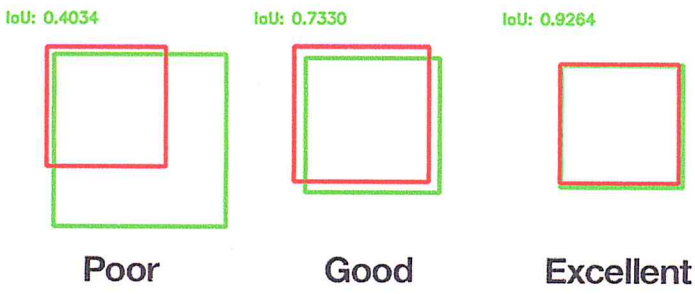


Figure 29: différent type d'IoU

5.Travaux connexes

Au cours des dernières années, diverses méthodes ont été proposées pour la classification des images. Ces méthodes peuvent être regroupées en trois catégories. La première catégorie utilise la similarité entre les images. Il est temps de commencer par extraire les composants de base d'image, puis de les utiliser pour la classification. Le travail dans la deuxième catégorie est axé sur le développement de descripteurs d'image locaux et / ou globaux. Ces descripteurs sont ensuite utilisés pour la classification. Extraire des fonctionnalités locales et globales est également un processus assez long. Enfin, les méthodes de la troisième catégorie utilisent CNN pour apprendre et extraire automatiquement des entités des images de document qui sont ensuite classées. [33]

Les réseaux convolutifs sont une forme particulière de réseaux neuronaux multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères, chaque élément n'est connecté qu'à un petit nombre d'éléments voisins dans la couche précédente. En 1995, **Yann** [34] ont développé un système automatique de lecture de chèques qui a été déployé largement dans le monde. À la fin des années 90, ce système lisait entre 10 et 20 % de tous les chèques émis aux États-Unis. Mais ces méthodes étaient plutôt difficiles à mettre en œuvre avec les ordinateurs de l'époque, et malgré ce succès, les réseaux à convolutionts et les réseaux neuronaux plus généralement ont été délaissés par la communauté de la recherche entre 1997 et 2012[35].

La détection d'objets dans des images, est une thématique ancienne de la Vision par Ordinateur, qui a connu des progrès impressionnants depuis les années 2000, grâce à des approches fondées sur l'apprentissage des caractéristiques de modèles de classes d'objets[36]. Depuis 2010 ces approches sont concurrencées par les méthodes fondées sur les réseaux de neurones de convolution (CNN), dont les capacités d'apprentissage sont bien supérieures, à

condition de disposer de bases d'apprentissage de très grandes dimensions, et de moyens de calcul très puissants : aussi il a été proposé d'exploiter des approches à plusieurs étapes, en faisant varier la complexité des architectures des CNN en utilisant une capacité minimal d'apprentissage. L'une des difficultés de la comparaison des différentes méthodes de détection de fruits, est qu'il n'existe pas de base de données d'images acquises sur des oranges par exemple dans un verger. Les images utilisées pour la détection de fruits ne les représentent pas dans des arbres, mais de façon isolée.

Plusieurs algorithmes détecteurs d'objets ont été testeés dans le cadre de la détection de fruits, une méthode repose sur l'algorithme de Viola et Jones [37] été proposée par **Wachs** [38] elle est appliquée sur des images multi-spectrales (visible achrome et thermique) acquises séparément sur un arbre. Les régions positives classées par AdaBoost, sont ensuite analysées par plusieurs réseaux de neurones votant pour confirmer ou infirmer ces premières détections. D'autres méthodes s'appuyant sur une segmentation par la couleur des pixels donnant de bons résultats si les images sont de très bonne qualité[39][40],ce qui nécessite en général des techniques de photographie inutilisable de manière automatique, celles ci reposent sur une détection des pommes visibles grâce au gradient et un apprentissage de la couleur des objets détectés, le principal problème de cette méthode est le fait que la première détection suppose qu'une ou plusieurs pommes soient assez visibles, mais aussi que toutes les pommes soient éclairées de manière uniforme, ce qui est impossible sans utiliser un éclairage puissant et bien positionné. Certaines méthodes utilisent aussi une classification en utilisant la texture[41].

S.Bargoti [42] utilisent une méthode de détection avec des réseaux de neurones de convolution. Elle se base sur l'algorithme Faster R-CNN, cette méthode permet de différencier différents fruits (pommes, mangues et amandes), le problème majeur des CNNs basés sur les régions aient été coûteux en termes de calcul. De nombreux chercheurs se soient penchés sur le problème de la détection des fruits, comme les travaux présentés dans [43] et [44] le problème de la création d'un système de détection rapide et fiable persiste [45]. Cela est dû à la forte variation de l'apparence des fruits dans les paramètres du champ, y compris les propriétés de la couleur, de la forme, de la taille, de la texture et de la réflectance. De plus, dans la plupart de ces environnements, les fruits sont partiellement abstraits et soumis à des conditions d'éclairage et d'illumination en constante évolution. Divers travaux présentés dans la littérature abordent le problème de la détection des fruits en tant que problème de segmentation de l'image **Wang** [46] ont examiné la question de la détection des pommes pour la prédiction du rendement. Ils ont mis au point un système qui détecte les pommes en

fonction de leur couleur et de leur motif de réflexion spéculaire distinctif. D'autres informations, telles que la taille moyenne des pommes, ont été utilisées pour supprimer les détections erronées ou pour séparer les zones pouvant contenir plus d'une pomme. Une autre heuristique employée consistait à accepter comme détections uniquement les régions les plus arrondies.

Bac [47] ont proposé une approche de segmentation pour les poivrons. Ils ont utilisé une caméra multi spectrale à six bandes et ont utilisé une gamme de fonctionnalités, y compris des données multi spectrales brutes, des indices de différence standardisés, ainsi que des caractéristiques de texture basées sur l'entropie. Des expériences dans un environnement de serre hautement contrôlé ont montré que cette approche produit des résultats de segmentation raisonnablement précis. Cependant, les auteurs ont noté qu'il n'était pas assez précis pour constituer une carte des obstacles fiable.

De plus, la vision par ordinateur en agriculture[45], a été explorée dans de multiples publications des études à des fins de détection de fruits et d'estimation du rendement[48]. Le traitement des images dans les vergers couvre une grande variété de fruits tels que le raisin[49], les mangues[50], les pommes[51], agrumes[52], les kiwis[53] et les pêches[54].

La classification des fruits est généralement effectuée en transformant des régions d'image en espaces caractéristiques discriminants et en utilisant un classificateur pour les associer à des régions de fruits ou à des objets d'arrière-plan tels que feuillage, branches, sol, etc. adjacente à chaque pixel de l'image. la sortie est une image densément segmentée. Des techniques de post-traitement peuvent ensuite être appliquées pour différencier les objets d'intérêt particuliers. Une approche spécifique de la détection réduit l'espace de recherche de région en détectant initialement les points clés. Ici, des régions d'image intéressantes (candidats possibles pour les fruits) sont d'abord extraites en utilisant des contraintes accordées manuellement conçues pour le jeu de données particulier. Ceci est suivi par l'extraction des caractéristiques et la classification comme précédemment.

La détection des fruits par extraction et la classification des points clés sont souvent appliquées aux vignes et aux vergers. Par exemple, [43] **Nuske** en (2014) exploitent les symétries radiales dans la réflexion spéculaire des baies individuelles pour extraire les points clés, qui sont ensuite classés comme baies ou non-baies. Par ailleurs, la segmentation d'image renvoie une carte de vraisemblance riche des fruits, sur laquelle un seuil peut être appliqué pour obtenir un masque de fruits binaire détaillant les régions de l'image contenant des

fruits[55], conçoivent un ensemble de mesures heuristiques basées sur des couleurs et des textures locales afin de classer les pixels individuels sous forme de mangues ou de non-créneaux.

Linkel [51] incorpore un post-traitement supplémentaire pour la détection des pommes, où les gouttes individuelles sont développées, segmentées et combinées pour gérer les grappes de fruits et de fruits fermés. **Stajnko**[56] qui ont été proposés en (2009), utilise plutôt l'analyse de forme et la correspondance de modèles pour extraire des pommes circulaires de l'image segmentée. **Yamamoto** qui ont été proposés en (2014) implémente un second composant de classification sur les taches de tomates extraites par segmentation d'image pour supprimer toute détection d'arrière-plan[44].

Les méthodes proposées dans la littérature utilisent la segmentation, les réseaux de neurones ou même les réseaux de neurones convolution ... etc. Dans notre mémoire nous présentons une variante de l'apprentissage automatique (KNN, SVM) ainsi que l'apprentissage profond: différents algorithmes d'apprentissage en profondeur. Réseau de neurones convolutifs. Nous élargissons encore cette étude en la comparant modèles CNN appliqué sur une base de données avec et sans augmentation de donnés (Data augmentation).

Le détecteur Faster R-CNN créé par S.Ren [57] qui repose sur une détection entièrement faite avec des réseaux de neurones de convolution d'où un premier réseau de neurones de convolution prend en entrée une image de taille quelconque et donne en sortie des régions dans lesquelles pourraient se trouver les objets à détecter, le second réseau prend en entrée les régions proposées par le premier réseau et recherche si elles contiennent l'objet à détecter. Parmi les inconvénients de Faster R-CNN est qu'il se déroule en plusieurs étapes, il est couteux en espace. par conséquence la détection d'objet est lente. Une autre solution serait l'utilisation de l'algorithme YOLO développé par **J.Redmon** [32], cet algorithme prédit directement les boîtes englobantes (un quadrillage de l'image où l'on prédit la classe de l'objet s'il existe dans notre cas soit une orange soit rien) et les probabilités de classe avec un seul réseaux de neurones de convolution, en une seule évaluation. La simplicité du modèle YOLO permet des prévisions en temps réel. Initialement, le modèle prend une image en entrée.

Il le divise en une grille SxS. Chaque cellule de cette grille prédit les cases B avec un score de confiance. Cette confiance est simplement la probabilité de détecter l'objet multiplié par l'IoU entre les boîtes de vérité prédites et au sol[58].

L'avantage de cette prédiction est qu'elle peut se faire indépendamment de la première détection mais qu'elle ne cible pas les objets en particulier ; il est donc plus difficile de détecter les plus petits objets.

Conclusion

Nous venons de faire un tour de littérature où nous avons présenté les principaux domaines et approches chevauchant avec notre projet ainsi que les définitions nécessaires pour la compréhension de la suite. Nous avons consacré ce chapitre à la présentation des notions de la classification ainsi que leurs intérêts dans le domaine d'imagerie. Nous avons également abordé l'utilisation des réseaux de neurones dans ce domaine ainsi que les réseaux de neurones convolutionnels. Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée. Nous avons présenté également d'autres approches d'apprentissage automatique tel que l'approche SVM, l'approche K-NN, et finalement la méthode YOLO pour la détection d'objets. En se basant sur cette étude, nous présentons dans les chapitres suivants nos modèles et leurs implémentations. Ensuite, nous exposons les résultats obtenus dans la phase d'apprentissage et de teste.

CHAPITRE 2

Classification et détection des fruits: approches retenues

III. INTRODUCTION

Dans ce chapitre nous mettons en évidence le coté conceptuel de notre application , permet de détailler les différentes architectures utilisées avec les différents algorithmes choisi pour faire la classification et la détection des fruits et qui serons implémenter dans la phase suivante. Nous avons choisi 5 algorithmes à tester (SVM, K-NN, CNN, ANN, YOLO) par la suite nous allons exposer l'architecture de chacun d'eux.

1. Conception des approches pour la classification des fruits

1.1. Réseaux de neurones à convolution(CNN)

La définition de nouvelle configuration se fait de manière expérimentale, pour cela on a choisi quatre modèles avec différentes architectures (configurations) deux pour faire une classification multiple et deux autres pour une classification binaire:

- Modèle 1 : il est appliqué sur une base de données multiple contient quatre types de fruits (banane, fraise, orange, poire) nommée **Dataset1**.
- Modèle 2 : il est appliqué sur une base de données binaire contient deux types de classes : feuille infectée ou feuille normale nommée **Dataset2**.
- Modèle 3: il est appliqué sur une base de données binaire contient deux types de classes : pomme infectée ou pomme normale (saine) nommée **Dataset3**.
- Modèle 4 : il est appliqué sur une base de données multiple avec un nombre considérable de classes (60) nommée **Fruit 360 dataset**.

1.1.1 Architecture de notre réseau de neurones à convolution

Nous avons testé de manière expérimentale différentes configurations et nous avons abouti à quatre modèles où nous obtenons des meilleurs résultats.

Dans ce qui suit nous présentons les architectures des 4 modèles :

1.1.1.1 Architecture du Modèle 1

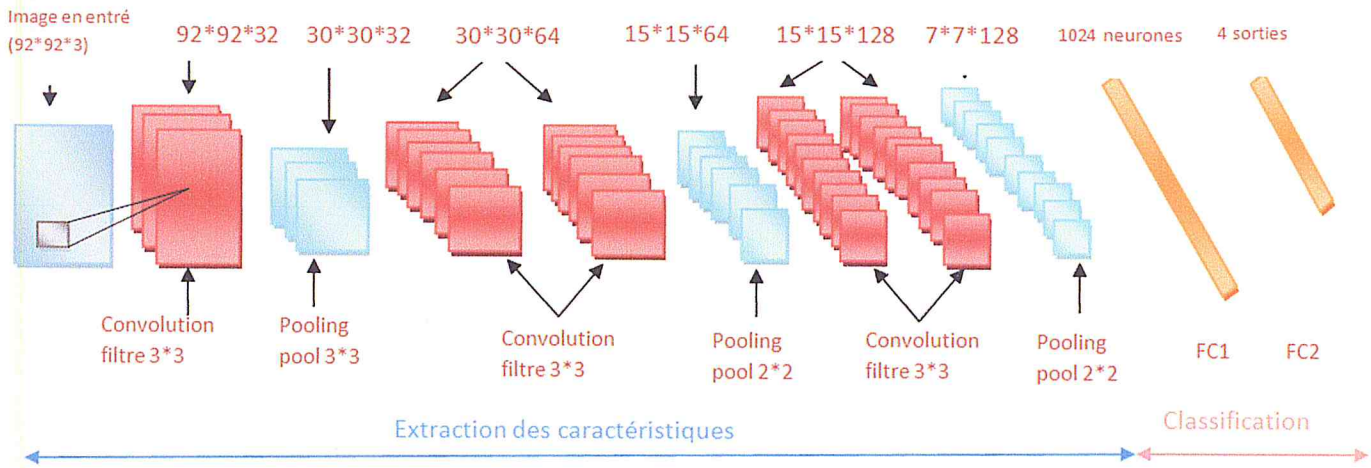


Figure 30 : Architecture du Modèle 1

La **Figure 29** montre le premier modèle, il est composé de cinq **couches de convolution**, trois couches de **Maxpooling** et deux couches de **fully connected**.

La couche de convolution a 32 filtres avec un noyau 3 x 3. Nous utilisons **ReLU** la fonction d'activation qui force les neurones à retourner des valeurs positives, suivie de la normalisation par lots **BatchNormalization**.

Notre couche **Maxpooling** utilise une taille POOL 3 x 3 pour réduire rapidement les dimensions spatiales de 92 x 92 à 30 x 30 (nous utilisons des images d'entrée de 92 x 92 x 3 pour former notre réseau comme nous le verrons dans la section suivante) .

Nous utiliserons également l'abandon (*Dropout*) dans notre architecture de réseau. L'abandon supprime aléatoirement et temporairement des liens entre les neurones, en fixant les poids de sortie à zéro. Cette procédure permet au CNN d'apprendre même s'il manque de l'information. Cette stratégie est implémentée afin de limiter le sur apprentissage (*Overfitting*). Le pourcentage de liens à déconnecter, est fixé dans *keras*.

De là, nous allons ajouter deux couches de convolutions successives suivi de la fonction **ReLU** avant d'appliquer une autre couche **Maxpooling** pour réduire la taille de l'image ainsi

la quantité des paramètres et de calcul. À la sortie de cette couche, nous aurons 64 feature maps de taille 15*15.

On répète la même chose avec les couches de convolutions quatre et cinq, ces couches sont composées de 128 filtres, la fonction d'activation **ReLU** est appliquée toujours sur chaque convolution. Une couche de **Maxpooling** est appliquée après la couche de convolution cinq. À la sortie de cette couche, nous aurons 128 feature maps de taille 7*7.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé de deux couches **fully connected**. La première couche contient 1 024 neurones où la fonction d'activation utilisée est le **ReLU**, et la deuxième couche est un **Softmax** qui permet de calculer la distribution de probabilité des 4 classes (nombre de classe dans la base d'image Dataset1).

Le tableau suivant montre l'architecture du modèle 1.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 92, 92, 32)	896
activation_1 (Activation)	(None, 92, 92, 32)	0
batch_normalization_1 (Batch	(None, 92, 92, 32)	368
max_pooling2d_1 (MaxPooling2	(None, 30, 30, 32)	0
dropout_1 (Dropout)	(None, 30, 30, 32)	0
conv2d_2 (Conv2D)	(None, 30, 30, 64)	18496
activation_2 (Activation)	(None, 30, 30, 64)	0
batch_normalization_2 (Batch	(None, 30, 30, 64)	120
conv2d_3 (Conv2D)	(None, 30, 30, 64)	36928
activation_3 (Activation)	(None, 30, 30, 64)	0
batch_normalization_3 (Batch	(None, 30, 30, 64)	120
max_pooling2d_2 (MaxPooling2	(None, 15, 15, 64)	0
dropout_2 (Dropout)	(None, 15, 15, 64)	0

conv2d_4 (Conv2D)	(None, 15, 15, 128)	73856
activation_4 (Activation)	(None, 15, 15, 128)	0
batch_normalization_4 (Batch	(None, 15, 15, 128)	60
conv2d_5 (Conv2D)	(None, 15, 15, 128)	147584
activation_5 (Activation)	(None, 15, 15, 128)	0
batch_normalization_5 (Batch	(None, 15, 15, 128)	60
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 128)	0
dropout_3 (Dropout)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 1024)	6423552
activation_6 (Activation)	(None, 1024)	0
batch_normalization_6 (Batch	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 4)	4100
activation_7 (Activation)	(None, 4)	0

Total params: 6,710,236
Trainable params: 6,707,824
Non-trainable params: 2,412

Tableau A.1 Configuration du modèle 1

1.1.1.2 Architecture du Modèle 2

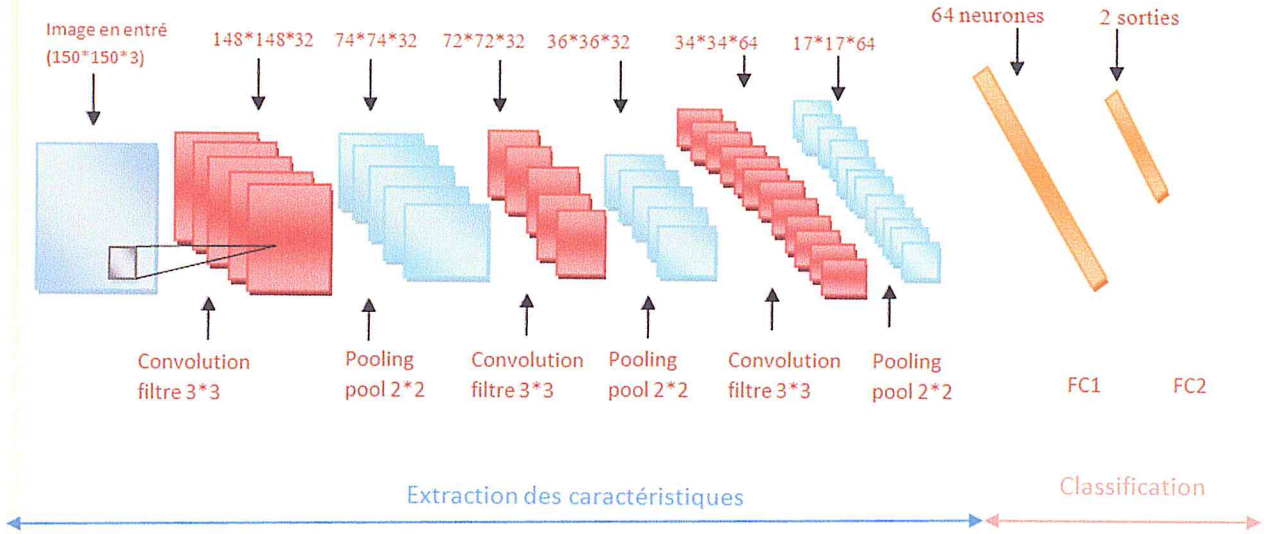


Figure 31 : Architecture du Modèle 2

Le deuxième modèle montré dans la **Figure 30** est composé de 3 couches de convolution et trois couches de **Maxpooling** et deux couches de **fully connected**.

L'image en entrée est de taille 150*150, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, la fonction d'activation **ReLU** est utilisée pour forcer les neurones à retourner des valeurs positives, après cette convolution 32 **features maps** de taille 32*32 seront créés.

Une couche de **Maxpooling** est appliquée après la première couche de convolutions pour réduire autant que possible la taille des images et des paramètres. À la sortie de cette couche, nous aurons 32 **feature maps** de taille 74*74.

La deuxième couche de convolution est composées de 32 filtres de taille 72*72 la fonction d'activation ReLU est appliquée toujours sur chaque convolution, ensuite on applique la couche Maxpooling réduire la taille de l'image. À la sortie de cette couche, nous aurons 32 feature maps de taille 36*36.

La troisième couche de convolution est composée de 64 filtres de taille 34*34 en utilisant la fonction d'activation ReLU, après la dernière couche Maxpooling est appliquée, en sortie donne 64 feature maps de taille 17*17.

Après ces trois couches de convolution nous utilisons un réseau de neurones composé de deux couches fully connected. La première couche contient 64 neurones où la fonction d'activation utilisée est le **RELU**, et la deuxième couche est une fonction sigmoïde non-linéarité la raison principale pour laquelle nous utilisons la fonction sigmoïde est parce qu'elle existe entre (0 à 1). Par conséquent, il est particulièrement utilisé pour les modèles où nous devons prédire la probabilité comme une sortie. Puisque la probabilité de n'importe quoi existe seulement entre la gamme de 0 et 1, sigmoïde est le bon choix pour la classification binaire.

Le tableau suivant montre l'architecture du modèle 2.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
activation_1 (Activation)	(None, 148, 148, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 32)	9248
activation_2 (Activation)	(None, 72, 72, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 32)	0
conv2d_3 (Conv2D)	(None, 34, 34, 64)	18496
activation_3 (Activation)	(None, 34, 34, 64)	0
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 64)	0
flatten_1 (Flatten)	(None, 18496)	0
dense_1 (Dense)	(None, 64)	1183808
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
activation_5 (Activation)	(None, 1)	0

Total params: 1,212,513

Trainable params: 1,212,513
Non-trainable params: 0

Tableau A.2 Configuration du modèle 2

1.1.1.3 Architecture du Modèle 3

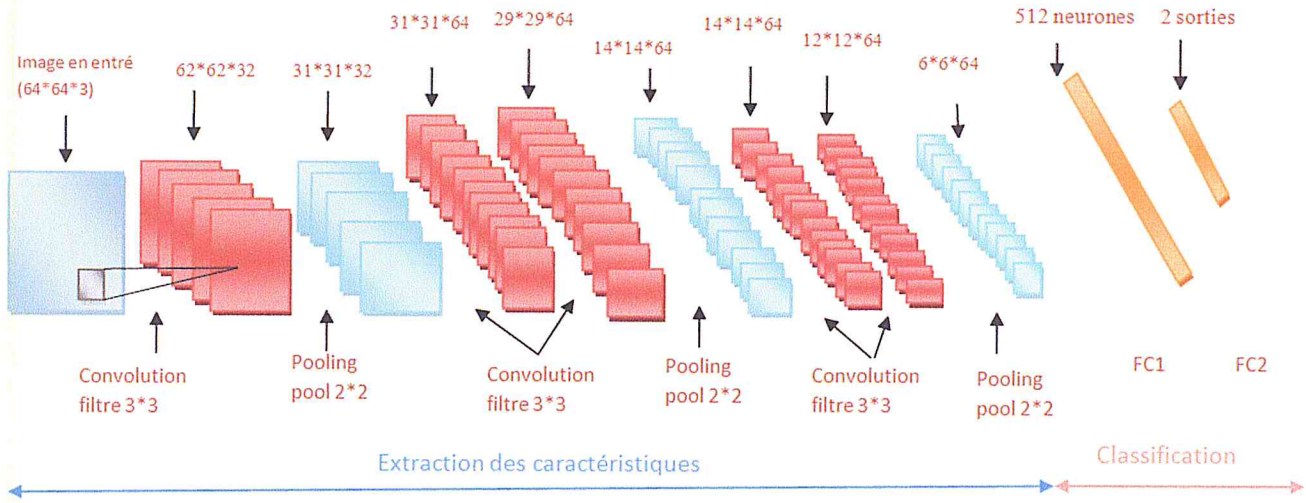


Figure 32 : Architecture du Modèle 3

L'architecture du modèle 3 que nous présentons dans la **Figure 31** est composée de cinq couches de convolution et trois couches de **Maxpooling** et deux couches de **fully connected**. L'image en entrée est de taille 64*64, l'image passe d'abord à la première couche de convolution composée de 32 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation **ReLU** après cette convolution 32 **features maps** de taille 32*32 seront créés.

Ensuite, on applique la fonction **Maxpooling** pour réduire la taille de l'image ainsi la quantité des paramètres et de calcule. À la sortie de cette couche, nous aurons 32 feature maps de taille 31*31.

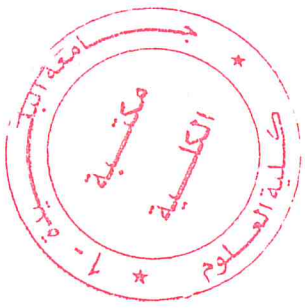
On applique les couches de convolutions deux et trois ces couches sont composées de 64 filtres, la fonction d'activation **ReLU** est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution trois. À la sortie de cette couche, nous aurons 64 feature maps de taille 14*14.

On répète la même chose avec les couches de convolutions quatre et cinq qui sont composées de 64 filtres, la fonction d'activation **ReLU** est appliquée toujours sur chaque convolution. Ensuite on applique la couche de Maxpooling après la couche de convolution quatre. À la sortie de cette couche, nous aurons 64 feature maps de taille 6*6.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé de deux couches fully connected. La première couche a 512 neurones où la fonction d'activation utilisée est le ReLU, et la deuxième couche est une fonction sigmoïde qui permet de calculer la distribution de probabilité de deux classes (nombre de classes dans la base d'image Dataset3).

Le tableau suivant montre la configuration de Modèle 3.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
dropout_1 (Dropout)	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 31, 31, 64)	18496
conv2d_3 (Conv2D)	(None, 29, 29, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
conv2d_4 (Conv2D)	(None, 14, 14, 64)	36928
conv2d_5 (Conv2D)	(None, 12, 12, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 64)	0
dropout_3 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 512)	1180160



dropout_4 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 2)	1026
activation_1 (Activation)	(None, 2)	0

Total params: 1,311,362
Trainable params: 1,311,362
Non-trainable params: _____

Tableau A.3 Configuration du modèle 03

1.1.1.4 Architecture du Modèle 04

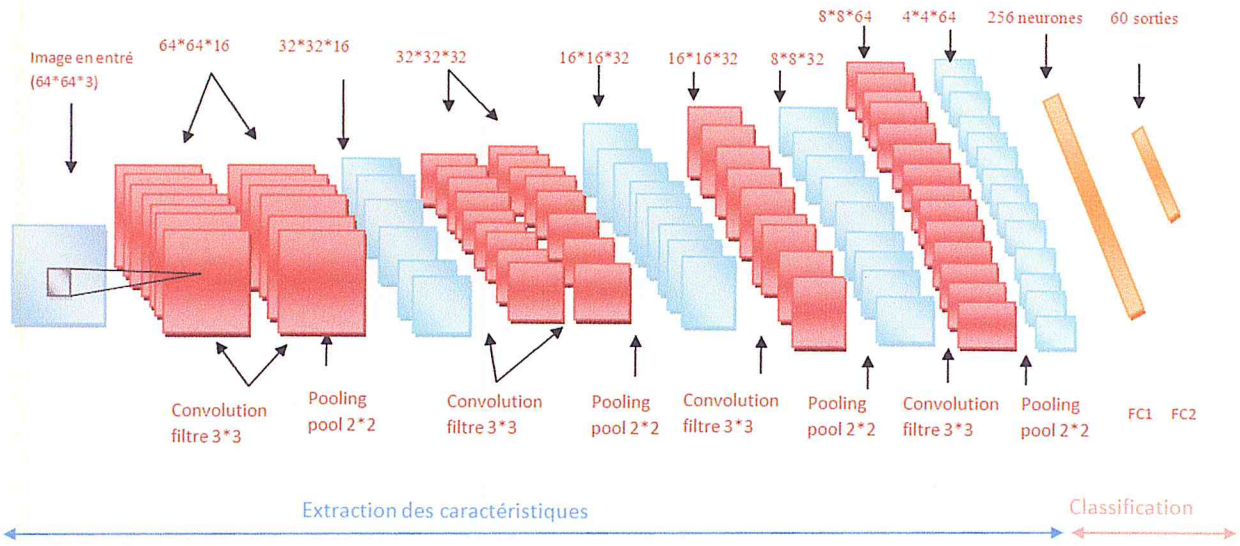


Figure 33 Architecture du Modéle4

Notre Modèle 4 montré dans la **Figure 32** est composé de six couches de convolution et quatre couches de Maxpooling et deux couches de fully connected.

Notre modèle commence par deux couches de convolution successives les deux couches ayont 16 filtres avec un noyau 3 x 3. Nous utilisons **Leaky ReLU** la fonction d'activation qui force les neurones à retourner des valeurs positives, contrairement à **ReLU**, **Leaky ReLU** est plus «équilibrée» et peut donc apprendre plus rapidement, suivie de la normalisation par lots **Batch Normalization**.

Après on utilise la couche **Maxpooling** avec le taille POOL 2 x 2 pour réduire les dimensions spatiales de 64 x 64 à 32 x 32.

De là, nous allons ajouter deux autres couches de convolutions successives avec 32 filtres la fonction **Leaky ReLU** est utilisée, avant d'appliquer une autre couche **Maxpooling.** À la sortie de cette couche, nous aurons 32 feature maps de taille 16*16.

On répète la même chose avec la couche de convolution trois, elle est composée de 32 filtres toujours, avec l'utilisation de la fonction d'activation **Leaky ReLU**, ensuite on applique la couche **Maxpooling** à la sortie on a 32 feature maps de taille 8*8, même chose avec la couche de convolution quatre qui contient 64 filtres on applique la couche de **Maxpooling** nous aurons 64 feature maps de taille 4*4 à la sortie.

Après ces quatre couches de convolution, nous utilisons un réseau de neurones composé de deux couches fully connected. La première couche contient 1 024 neurones où la fonction d'activation utilisée est le RELU, et la deuxième couche est un softmax qui permet de calculer la distribution de probabilité des 60 classes (nombre de classe dans la base d'image **Fruit 360 dataset**).

Le tableau suivant montre la configuration de Modèle 4.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 16)	448
conv2d_2 (Conv2D)	(None, 64, 64, 16)	2320
leaky_re_lu_2 (LeakyReLU)	(None, 64, 64, 16)	0
batch_normalization_1 (Batch	(None, 64, 64, 16)	64
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 16)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	4640
conv2d_4 (Conv2D)	(None, 32, 32, 32)	9248
leaky_re_lu_3 (LeakyReLU)	(None, 32, 32, 32)	0
batch_normalization_2 (Batch	(None, 32, 32, 32)	128
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_5 (Conv2D)	(None, 16, 16, 32)	9248
leaky_re_lu_4 (LeakyReLU)	(None, 16, 16, 32)	0

batch_normalization_3 (Batch	(None, 16, 16, 32)	128
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_6 (Conv2D)	(None, 8, 8, 64)	18496
leaky_re_lu_5 (LeakyReLU)	(None, 8, 8, 64)	0
batch_normalization_4 (Batch	(None, 8, 8, 64)	256
max_pooling2d_4 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 256)	262400
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 60)	15420
activation_1 (Activation)	(None, 60)	0

Total params: 322,796
Trainable params: 322,508
Non-trainable params: 288

Tableau A.4 Configuration du modèle 4

1.2. Réseaux de neurones simples

Nous avons utilisé une base d'image (Dataset1) de quatre classes chaque échantillon contient des images en couleur de tailles déférentes, les valeurs de pixels vont de 0 à 255. Nous appliquerons le prétraitement suivant aux données avant de les transmettre au réseau.

1.2.1. Phase prétraitement

Tout d'abord convertir chaque matrice d'image, exemple image en couleur de taille 64 x 64 (64 × 64 × 3) en un tableau (64 * 64 *3 = 12288 dimensions) qui seront alimentés au réseau comme une caractéristique unique. Ensuite nous convertirons les étiquettes d'un codage entier en un codage catégoriel car c'est le format requis par Keras pour effectuer une classification multi classe. Le codage à chaud unique est un type de représentation booléenne de données entières. Il convertie l'image en un tableau de tous les zéros sauf un 1 à l'index de classe. Par exemple, en utilisant un codage une chaud pour 4 classes, la deuxième classe sera codé 0100.

La **Figure ci-dessous** montre la l'architecture de notre réseau de neurones.

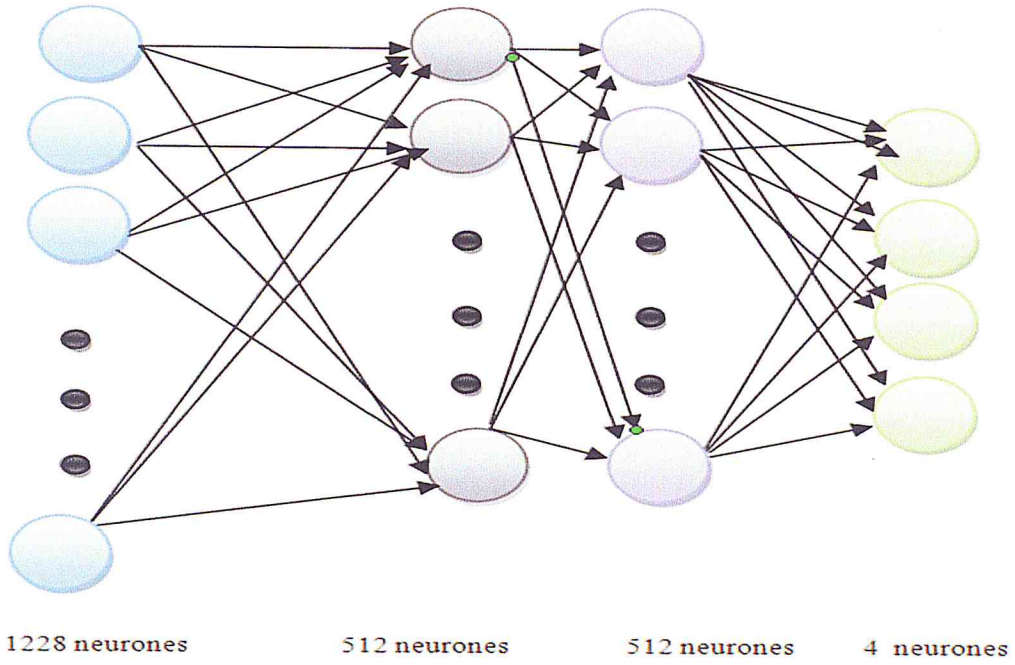


Figure 34 : Architecture de réseau de neurones

1.2.2. Architecture de notre réseau de neurones

Nous utilisons un réseau avec 2 couches cachées et une couche de sortie avec 4 unités. Le nombre d'unités dans la première couche cachée est maintenu à 512 et la deuxième couche est maintenue à 512. L'entrée dans le réseau est le tableau à 12288 dimensions converti à partir de l'image 64 x 64 x3.

Nous utilisons le modèle *Sequential* pour construire le réseau. Dans le modèle *Sequential*, nous pouvons simplement empiler les couches en ajoutant la couche souhaitée une par une. Nous utilisons la couche *Dense*, également appelée couche entièrement connectée, car nous construisons un réseau *Feedforward* dans lequel tous les neurones d'une couche sont connectés aux neurones de la couche précédente. Après nous ajoutons la fonction d'activation ReLU qui est nécessaire pour introduire la non-linéarité dans le modèle. Cela aidera le réseau à apprendre les limites de décision non linéaires. La dernière couche est une couche Softmax car c'est un problème de classification multiple (multiclasses).

Dans la troisième étape, nous configurons notre modèle on utilisant l'optimiseur à rmsprop. Nous spécifions également le type de perte qui est l'entropie croisée catégorielle utilisée pour la classification multiple. Nous spécifions également les métriques (précision dans ce cas) que nous voulons suivre pendant le processus de formation. Pour entrainer notre modèle on utilise la fonction fit () dans Keras. Nous spécifions le nombre d'époques comme 100. Cela signifie que tout le jeu de données sera transmis au réseau 100 fois. Nous utiliserons les données de test pour la validation.

Le diagramme montré dans la figure suivante recense la méthodologie d'entrainement et de classification de notre réseau de neurones.

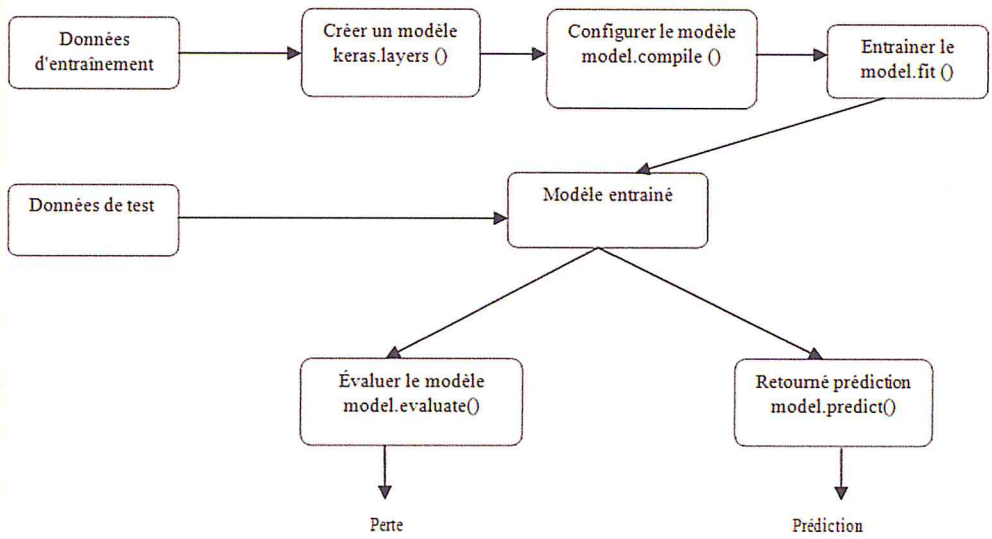


Figure 35 Méthodologie d'entrainement et de classification de notre réseau de neurones

1.3. Vecteur à Support Machine (SVM)

Dans la **Figure 35,** l'organigramme de l'exécution élabore chaque étape de la méthodologie globale et représente la manière dont les caractéristiques sont extraites et la classification effectuée.

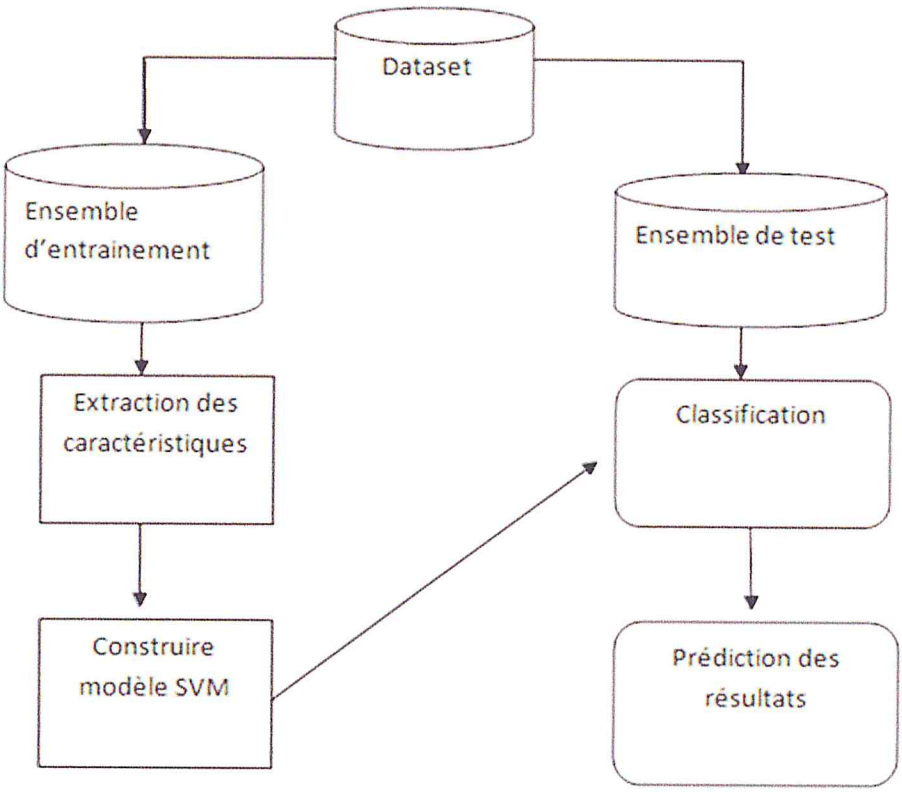


Figure 36: Méthodologie SVM

La méthodologie SVM se déroule en trois étapes :

1.3.1. Partitionner l'ensemble de données

Nous avons utilisé l'ensemble de données en quatre classes de fruits (banane, fraise, orange, poire) on partitionne d'abord l'ensemble de données en deux sous-ensembles principaux ensemble d'entrainement et ensemble de test.

1.3.2. Extraire les caractéristiques

Chaque échantillon de l'ensemble de données correspond à une "cible", c'est-à-dire la "réponse". Chaque pixel d'image prend une valeur comprise entre 0 et 255. Comme ce sont des images en couleurs, chaque pixel est représenté par trois valeurs distinctes R, G, B (3 canaux). Nous allons convertir cela en utilisant la fonction de refonte (*Reshape*) de numpy à

(64x64x3). C'est-à-dire une image qui mesure 64 pixels de largeur et 64 pixels d'hauteur, avec 3 canaux et qui est représenté comme une matrice (4744 échantillons, 64px,64px, 3ch)

Nous changeons la classe de sortie (étiquette) en format catégoriel ou un format chaud en utilisant la fonction **to_categorial**. C'est-à-dire au lieu d'une seule valeur de 0 à 3(classes), nous convertissons ceci en un tableau de taille 4, par exemple. y = [3] devient y = [0, 0, 1,0]. De plus, nous adaptons toutes les fonctionnalités (valeurs de pixels) d'une plage de 0 à 255, à une plage de 0 à 1. Ceci est fait en divisant chaque valeur de la matrice de caractéristiques par 255.

1.2.3. Entrainement et Classification

Nous avons utilisé le modèle SVM avec les deux paramètres C et gamma, le paramètre C, commun à tous les noyaux SVM, remplace la classification erronée des exemples d'apprentissage par la simplicité de la surface de décision. Un faible C rend la surface de décision lisse, tandis qu'un C élevé vise à classer correctement tous les exemples d'entraînement. Gamma définit l'influence d'un seul exemple d'entraînement. Plus le gamma est grand, plus les autres exemples doivent être proches. Nous avons utilisé C avec la valeur égale à 100 et gamma avec la valeur égale à 0.001 qui on donnée des meilleur résultats.

1.4. Plus proche voisin (K-NN)

L'algorithme K-NN repose simplement sur la distance entre les vecteurs de caractéristiques, nous avons les étiquettes associées à chaque image pour prédire et retourner une catégorie réelle pour l'image donné, nous devons définir une métrique de distance ou une fonction de similarité. Nous utilisons la distance euclidienne pour comparer les images à des fins de similarité.

$$d(x,y) = \sqrt{\sum_{i-1}^{N}(x_i - y_i)^2} \qquad [22]$$

Dans ce qui suit nous présentons la phase de prétraitement qui sert à extraire les différentes caractéristiques de l'image, et voir par la suite phase de classification.

1.4.1. Phase prétraitement

1.4.1.1. Construire un vecteur de caractéristiques

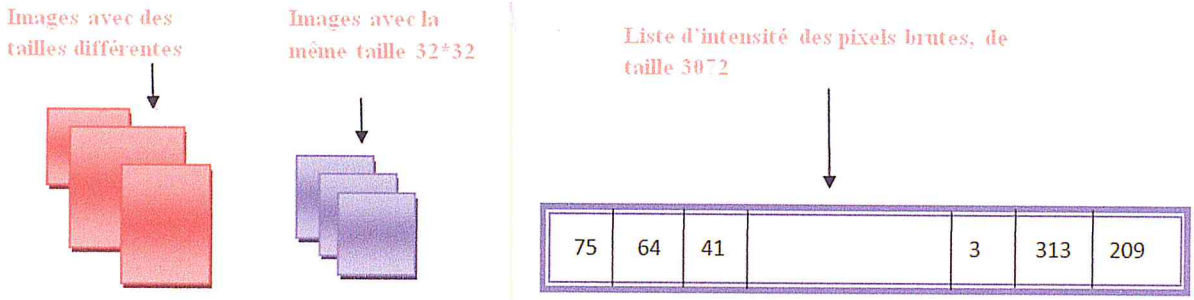


Figure 37 : Construction du vecteur de caractéristiques d'une image

Nous avons utilisé la méthode ***image_to_feature_vector*** elle est extrêmement naïve qui prend simplement une image d'entrée et la redimensionne à une largeur et une hauteur (taille) fixes, puis aplatit les intensités des pixels RVB (rouge, bleu, vert) en une seule liste de nombres.

Cela signifie que notre image d'entrée sera rétrécie à 32 x 32 pixels, et trois canaux pour chaque composante Rouge, Vert et Bleu respectivement, notre "vecteur de caractéristiques" en sortie sera une liste de 32 x 32 x 3 = 3 072 nombres.

Strictement parlant, la sortie de ***image_to_feature_vector*** n'est pas un vrai "vecteur de caractéristiques" puisque nous avons tendance à considérer les "caractéristiques" et les "descripteurs" comme des quantifications abstraites du contenu de l'image.

En outre, l'utilisation d'intensités de pixels brutes comme entrées pour les algorithmes d'apprentissage automatique a tendance à produire des résultats médiocres car même de petits changements de rotation, de translation, de point de vue, d'échelle, etc. peuvent influencer considérablement sur l'image elle-même.

1.4.1.2. Construire un histogramme de couleur

Nous définissons ensuite notre seconde méthode appelée **extract_color_histogram.** Comme son nom l'indique, cette fonction accepte une image d'entrée et construit un histogramme de couleur pour caractériser la distribution des couleurs de l'image.

Tout d'abord, nous convertissons notre image en espace couleur HSV. Ensuite pour trouver la similarité Nous appliquons la fonction cv2.calcHist pour calculer un histogramme 3D pour l'image en couleur.

Compte tenu de notre histogramme calculé, nous le normalisons ensuite, en prenant soin d'utiliser la signature de fonction cv2.normalize le but et de transformer l'image d'une certaine façon pour que la comparaison avec d'autres images ait un sens.

1.4.1.3. Méthodologie

Nous utiliserons cinq étapes pour former les classificateurs d'images :

Étape 1 - Structuration de notre ensemble de données initial:

Notre ensemble de données initial est composé de 5600 images représentant quatre type de fruits (banane, fraise, orange, poire). Ces images sont en couleurs, ont également été fortement prétraités ce qui rend notre travail de classification plus facile.

Étape 2 - Diviser l'ensemble de données:

Nous utiliserons deux divisions pour notre expérience. Le premier ensemble est notre ensemble d'entraînement, utilisé pour former notre classificateur k-NN. Nous utiliserons également notre classificateur en utilisant l'ensemble de test.

Étape 3 - Extraction des caractéristiques:

Nous utiliserons les intensités de pixels brutes, de l'image en couleur pour extraire les caractéristiques de chaque image, ensuite nous construisons l'histogramme, puis nous passerons à la normalisation.

Étape 4 - Formation de notre modèle de classification:

Notre classificateur K-NN formé sur les intensités de pixels brutes des images dans l'ensemble

Notre classificateur k-NN sera formé sur les intensités de pixels brutes des images dans l'ensemble d'apprentissage. Nous déterminerons ensuite la meilleure valeur de k en utilisant l'ensemble de test.

Étape 5 - Évaluation de notre classificateur:

Une fois que nous avons trouvé la meilleure valeur de k, nous pouvons ensuite évaluer notre classificateur k-NN sur notre ensemble de tests.

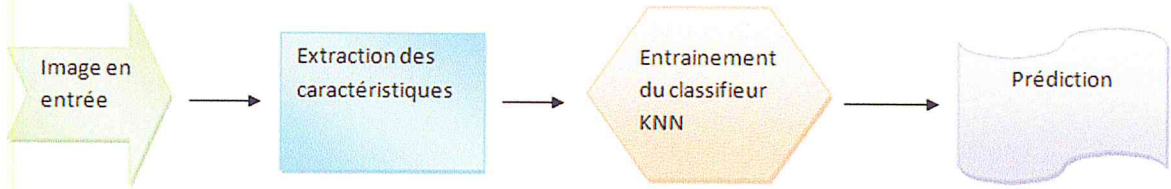


Figure 38: Méthodologie K-NN

2. Conception des approches pour la détection des fruits

Tous les systèmes de détection antérieurs réutilisent les classificateurs ou les localisateurs pour effectuer la détection. Ils appliquent le modèle à une image à plusieurs emplacements et échelles. Les régions à haut score de l'image sont considérées comme des détections.

Nous utilisons une approche totalement différente. Nous appliquons un seul réseau de neurones à l'image complète. Ce réseau divise l'image en régions et prédit les limites et les probabilités pour chaque région. Ces limites sont pondérées par les probabilités prédites.

2.1. Annotation de données

D'abord on crée pour chaque image (.jpg-image) le fichier (.xml-extention) correspondant, ce dernier porte le nom de l'image relative, chaque fichier .xml contient le nom d'image, sa taille et les coordonnées de(s) l'objet(s) sur cette image (dans notre cas les objets de la classe sont les oranges), la figure suivante donne un aperçu sur un fichier .xml d'une image donnée dans l'ensemble d'apprentissage :

Figure 39: Représentation de fichier XML

2.2. Création de données d'entraînement

L'un des principaux problèmes lors de l'entraînement de YOLO est le nombre d'images d'orange nécessaire pour l'apprentissage, ainsi que la qualité des images. Dans notre cas, les données utilisées pour l'entraînement sont composées de 1208 images annotées. Ces 1208 images contenant des images aléatoire des oranges.

2.3. Modèle Yolo

YOLO est un réseau de neurones à convolution, ce réseau n'utilise que des types de couches standard comme la couche de convolution avec un noyau 3x3 et maxpooling avec un noyau 2x2. Il n'y a pas de couche entièrement connectée. Le modéle utiliser est composée de 9 couches de convolution avec des activations ReLU perméables. Après la couche maxpool6 l'image d'entrée 416x416 devient une image 13x13, cette dernière est la taille de la grille qui divise l'image.

La sortie de ce modèle est une taille de lot tenseur 13x13x125. Dans ce tenseur, nous nous retrouvons donc avec 125 canaux pour chaque cellule de la grille. Ces 125 numéros contiennent les données pour les boîtes englobantes et les prédictions de classe. Les informations suivantes sont codées:

- Définition de boîte composée de: (x, y, largeur, hauteur, "est l'objet" confiance) x et y représentent le centre de la boite englobante par rapport à l'emplacement de la cellule de la grille.
- Probabilités de classe (seulement considérées si la confiance "est l'objet" est élevée)

YOLO est simple. Il prend une image en entrée (redimensionnée à 416 × 416 pixels), elle parcourt le réseau convolutif passé, et sort l'autre extrémité comme un tenseur 13 × 13 × 125 décrivant les boîtes de délimitation pour les cellules de la grille. Il calcule les scores finaux pour les boîtes de délimitation et de jeter ceux qui marquent moins de 30%. L'architecture de réseau est montrée dans la figure ci-dessous.

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

Figure 40: Yolo architecture

Conclusion

Ce chapitre a donné une vision sur notre travail, Nous avons montré l'aspect conceptuel de notre application à travers les différents modèles qui illustrent les différentes architectures choisies, pour les différents algorithmes utilisés tel que KNN, SVM, les réseaux de neurones, et une approche de classification basée sur les réseaux de neurones convolutionnels. De plus nous avons présenté une approche adaptée à la détection des fruits en utilisant l'algorithme YOLO .Le chapitre qui suit fera l'objet de la mise en œuvre de notre application on montre les résultats obtenus dans la phase d'apprentissage et de teste et les discuter et critiquer.

CHAPITRE 3

REALISATION ET ANALYSE DES RESULTATS

IV. Introduction

Dans ce dernier chapitre, nous présentons les différentes étapes réalisées durant l'implémentation de notre application, nous commençons à décrire la création des bases de données, ensuite nous discutons les résultats obtenus par les différentes architectures. Nous avons utilisé quatre algorithmes de classification supervisée. Ces quatre classificateurs supervisés sont l'algorithme de K plus proche voisin (K-NN), les séparateurs à vastes marges (SVM), les réseaux de neurones multicouches (PMC) et les réseaux de neurones à convolution plus YOLO algorithme dédié à la détection d'images.

1. Création d'un base d'image en utilisant Google image

Les algorithmes d'apprentissage en profondeur, en particulier les réseaux de neurones à convolution, peuvent être des bêtes avides de données. Pour aggraver les choses, mettre manuellement un ensemble de données d'images peut prendre du temps, fastidieux et même coûteux.

Alors, y a-t-il un moyen de tirer parti de la puissance de Google Images pour rassembler rapidement des images et réduire ainsi le temps nécessaire à la création de jeu de données.

Pour rassembler rapidement des données pour les modèles d'apprentissage en profondeur on a utilisé Google Images, JavaScript et un peu de Python.

La première étape dans l'utilisation de Google Images pour collecter des données d'entraînement pour notre réseau de neurones à convolution consiste à accéder à Google Images et entrer une requête, dans ce cas, nous utiliserons le terme de requête 'banane'.

Dans l'exemple suivant, nous avons nos résultats de recherche. L'étape suivante consiste à utiliser un tout petit peu de JavaScript pour rassembler les URL de l'image (que nous pourrons ensuite télécharger en utilisant Python).

Dans l'exemple suivant, nous avons nos résultats de recherche. L'étape suivante consiste à utiliser un tout petit peu de JavaScript pour rassembler les URL de l'image (que nous pourrons ensuite télécharger en utilisant Python).

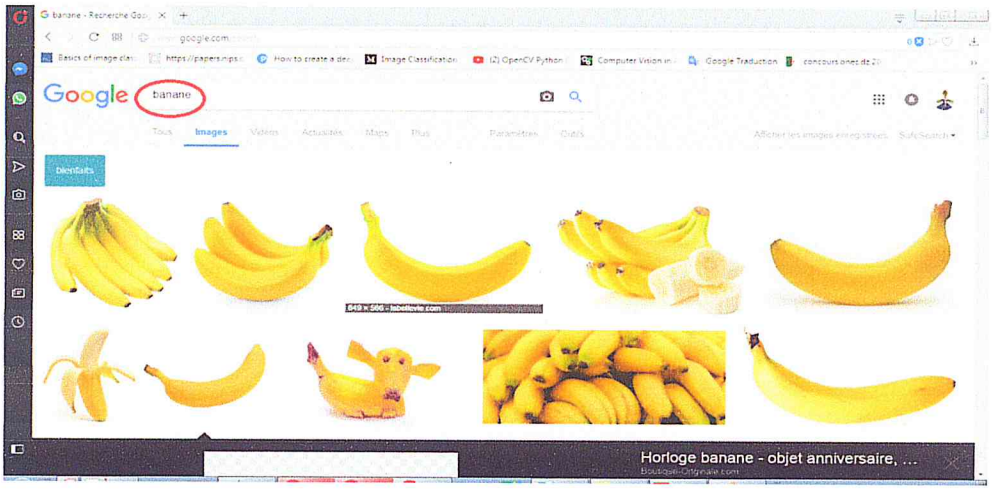


Figure 41: Résultats de recherche du terme 'banane'

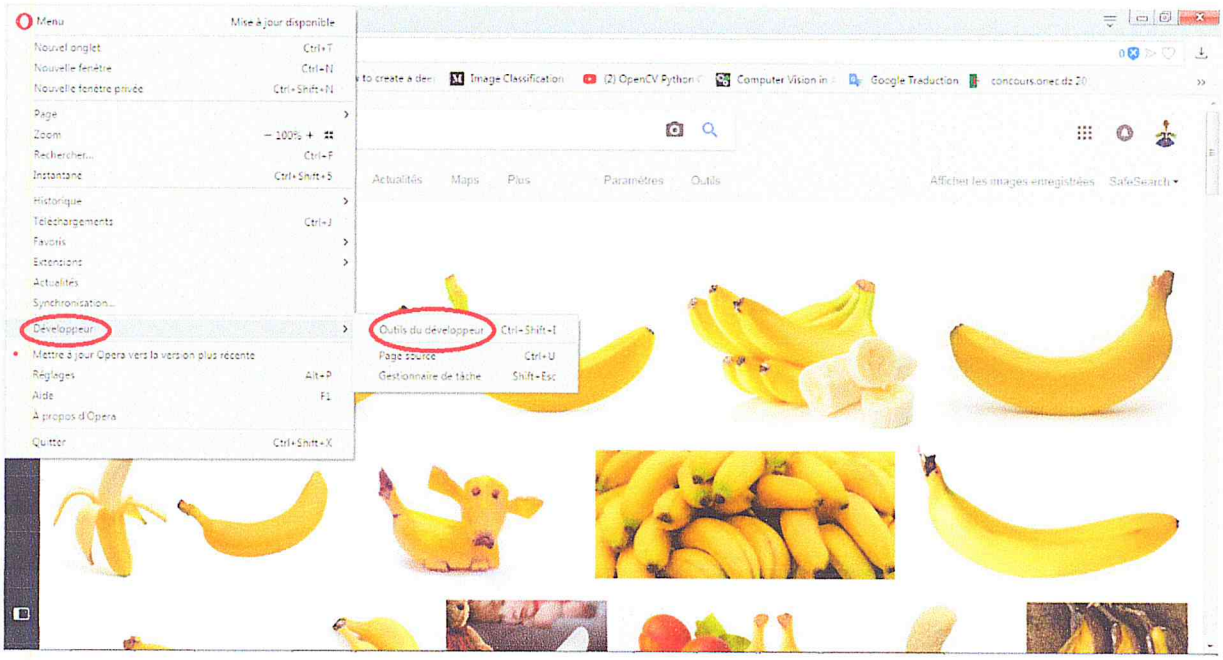


Figure 42: Lancer la console JavaScript

Après on lance la console JavaScript en cliquant sur Développeur puis Outils du Développeur Comme montré dans la **Figure 41**. De là cliquons sur Console.

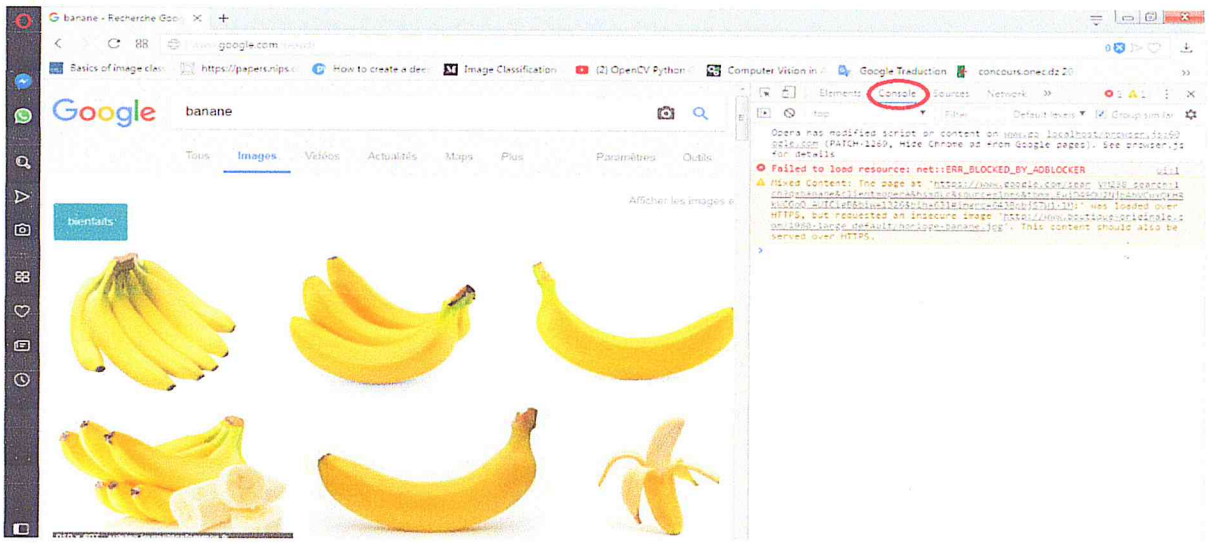


Figure 43: Console JavaScript

La figure suivante représente le déroulement de jQuery

```
// Déroule jquery dans la console JavaScript
var script = document.createElement('script');
script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);
```

Figure 44: Déroulement de jQuery

L'extraction de code JavaScript dans la console comme montrés dans la **Figure 43** montre la bibliothèque jQuery JavaScript, un paquet commun utilisé pour presque toutes les applications JavaScript. Maintenant que jQuery est arrêté, nous pouvons utiliser un sélecteur CSS pour récupérer une liste d'URL: comme montré ci-dessous

```
// récupérer les URL
var urls = $('.rg_di .rg_meta').map(function() { return JSON.parse($(this).text()).ou; });
```

Figure 45: Récupération ds URLs

Après l'exécution de l'extrait montré dans la **Figure 44** un fichier nommé ''banane.txt'' est téléchargé dans le répertoire Téléchargements par défaut, et il est utilisé par un programme Python pour charger chaque image individuellement.

La figure suivante représente l'enregistrement des URLs dans un fichier texte.

```
// écrire les URls dans le fichier (un par ligne)
var textToSave = urls.toArray().join('\n');
var hiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,' + encodeURI(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'banane.txt';
hiddenElement.click();
```

Figure 46: Enregistrement des URLs dans un fichier texte

2. Bases de données utilisées

Dataset1, Dataset2, Dataset3 et Dataset4 sont des sous-ensembles d'images étiquetés collecté depuis Google.

Fruits 360 dataset est une base d'image qui contient 60 classes de fruits.

2.1. Dataset1

La base d'image Dataset1 se compose de 5600 images couleurs, ces images sont réparties en 4 classes de fruit (banane, fraise, orange, poire). Dans cette base on a partitionné 80% de la totalité d'mages pour l'apprentissage et 20% pour le teste.

2.2. Dataset2

La base d'image Dataset2 contient 2800 images couleur, ces images sont réparties en 2 classes binaires (feuille malade, feuille normal) avec 1400 images par classe. Pour chaque classe on trouve 1000 images pour l'apprentissage et 400 images pour le teste.

2.3. Dataset3

C'est une base d'image qui contient les mêmes caractéristiques que Dataset2 sauf qu'elle traite les deux classes binaires (pomme malade, pomme seine).

2.4. Fruits 360 dataset

C'est une base d'image qui contient les fruits suivants: Pommes (différentes variétés: Golden, Golden-Red, Granny Smith, Rouge, Red Delicious), Abricot, Avocat, Avocat mûr, Banane (Jaune, Rouge), Cactus, Cantaloup (2 variétés), Carambula , Cerise (différentes variétés, Rainier), Cire de cerise (Jaune, Rouge, Noir), Clémentine, Cocos, Dattes, Granadilla, Raisin (Rose, Blanc, Blanc2), Pamplemousse (Rose, Blanc), Goyave, Huckleberry, Kiwi, Kaki , Kumsquats, Citron (normal, Meyer), Citron vert, Litchi, Mandarine, Mangue, Maracuja, Melon Piel de Sapo, Mûrier, Nectarine, Orange, Papaye, Fruit de la passion, Pêche, Pepino, Poire (différentes variétés, Abate, Monster, Williams), Physalis (normal, avec Cosse), Ananas (normal, Mini), Pitahaya Rouge, Prune, Grenade, Coing, Ramboutan, Framboise, Salak, Fraise (normale, Wedge), Tamarillo, Tangelo.

- Nombre total d'images: 49606.
- Taille de l'ensemble d'entraînement: 37101 images (un fruit par image).
- Taille de l'ensemble d'essai: 12460 images (un fruit par image).
- Nombre de classes: 74 (fruits).
- Taille de l'image: 100x100 pixels.

Format du fichier: image_index_100.jpg (par exemple 32_100.jpg) ou r_image_index_100.jpg (par exemple, r_32_100.jpg) ou r2_image_index_100.jpg. "r" signifie fruits tournés. "r2" signifie que le fruit a été tourné autour du 3ème axe. "100" provient de la taille de l'image (100x100 pixels).

Différentes variétés du même fruit (pomme par exemple) sont stockées comme appartenant à différentes classes.

3. Logiciels et librairies utilisé dans l'implémentation

Notre choix de technologie s'est basé sur des facteurs tels que l'infrastructure envisagée, l'environnement technique du projet et la pérennité de technologie pour le futur. A base de ces facteurs décrits ci-dessus Nous avons choisi le Framework **Python,** l'environnement **Jupyter Netbook** et un ensemble de bibliothèques.

Nous allons justifier notre sélection comme suit :

3.1. Python

Python est un langage script de haut niveau, structuré et open source, conçu pour être orienté objet, il n'en dispose pas moins d'outils permettant de se livrer à la programmation fonctionnelle ou impérative. Le langage Python est placé sous une licence libre et fonctionne sur la plupart des plates-formes informatiques, Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement.

3.2. Jupyter Notebook

Jupyter Notebook est une application Web open-source qui permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Les utilisations incluent: le nettoyage et la transformation des données, la simulation numérique, la modélisation statistique, la visualisation des données, l'apprentissage automatique.

3.3. TensorFlow

Tensorflow est un outil open source d'apprentissage automatique développé par Google, le code source a été ouvert le 9 novembre 2015. TensorFlow est considéré comme la première mise en œuvre sérieuse d'un cadre axé sur l'apprentissage en profondeur, depuis sa sortie, n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones.

3.4. Keras

Keras est l'une des bibliothèques Deep Learning les plus populaires en ce moment et a largement contribué à la banalisation de l'intelligence artificielle. Il est open source écrite en Pytho capable de fonctionner sur TensorFlow, Microsoft Cognitive Toolkit, Theano ou MXNet, c'est la clé pour faire de bonnes recherches, et se concentre sur être facile à utilisé il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Système

d'Exploitation de Robots Intelligents Neuroélectroniques Ouverts), et son principal auteur et mainteneur est François Chollet, un ingénieur de Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçu pour être une interface plutôt qu'un cadre autonome d'apprentissage automatique. Il offre un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilite le développement de modèles d'apprentissage en profondeur, quel que soit le système de calcul utilisé. Microsoft a également ajouté un backend CNTK à Keras, disponible dès CNTK v2.0.

3.5. SciPy

SciPy est un projet visant à unifier et fédérer un ensemble de bibliothèques Python à usage scientifique. Cette distribution de module est destinée à être utilisée avec le langage Python interprété pour créer un environnement de travail scientifique très similaire à celui offert par Scilab, GNU Octave, Matlab. Il contient par exemple des modules pour l'optimisation, l'algèbre linéaire, les statistiques, le traitement du signal ou encore le traitement d'images.

3.6. NumPy

NumPy bibliothèque logiciel open source du langage de programmation Python pour manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Elle fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

3.7. Scikit-learn

Scikit-learn est une bibliothèque d'apprentissage automatique pour le langage de programmation Python, Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support, et est conçu pour interopérer avec les librairies numériques et scientifiques Python NumPy et SciPy.

3.8. jQuery

jQuery est une bibliothèque qui permet d'agir sur le code HTML, CSS, JavaScript et AJAX. Plus précisément il permet de manipuler les éléments mis en place en HTML (textes, images, liens, vidéos, etc.) et mis en forme en CSS (position, taille, couleur, transparence, etc.) en utilisant des instructions simples qui donnent accès aux immenses possibilités de JavaScript et d'AJAX.

3.9. Darkflow

Darkflow est un outil Python 3 qui rend les réseaux de neurones open source disponibles en Python en utilisant Tensorflow.

3.10. Darknet

Darknet est un framework de réseau de neurones open source écrit en C et CUDA. Il est rapide, facile à installer et prend en charge le calcul CPU et GPU.

3.11. OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel , elle est gratuit pour une utilisation académique et commerciale. Il dispose d'interfaces C ++, Python et Java et prend en charge Windows, Linux, Mac OS, iOS et Android. OpenCV a été conçu pour l'efficacité de calcul et avec un fort accent sur les applications en temps réel. Écrit en C et C ++ optimisé, la bibliothèque peut tirer parti du traitement multicœur. Il peut tirer une parti de l'accélération matérielle de la plate-forme de calcul hétérogène sous-jacente.

Adopté dans le monde entier, OpenCV compte plus de 47 000 utilisateurs et un nombre de téléchargements estimé à plus de 14 millions. L'utilisation s'étend de l'art interactif, à l'inspection des mines, à l'assemblage de cartes sur le Web ou à la robotique avancée.

3.12. Configuration Utilisé dans l'implémentation

- La configuration du matériel utilisé dans l'implémentation du YOLO est :

Un PC portable Lenovo Core i7-4700 MQ

RAM de taille 8,00 GB

Disque dur de taille 2.40 GHz

Système d'exploitation Windows 10 64-bit

- La configuration du matériel utilisé dans l'implémentation du ANN, CNN, K-NN, SVM est :

Un PC portable TOSHIBA Core i3-3110 M

RAM de taille 4,00 Go

Disque dur de taille 2.40 GHz

Système d'exploitation Windows 7 64-bits

4. Résultats

Afin de montrer les résultats obtenus pour les quatre algorithmes utilisés, on illustre dans ce qui suit les résultats en termes de précision et d'erreur ainsi que matrice de confusion pour chacun des quatre algorithmes.

4.1. Résultats de classification obtenus avec le réseau de neurones a convolution

4.1.1. Résultat du modèle 1

La figure suivante (**Figure 46**) montre la précision et l'erreur pour le deuxième modèle 1 il est appliqué sur la base de donné Dataset1.

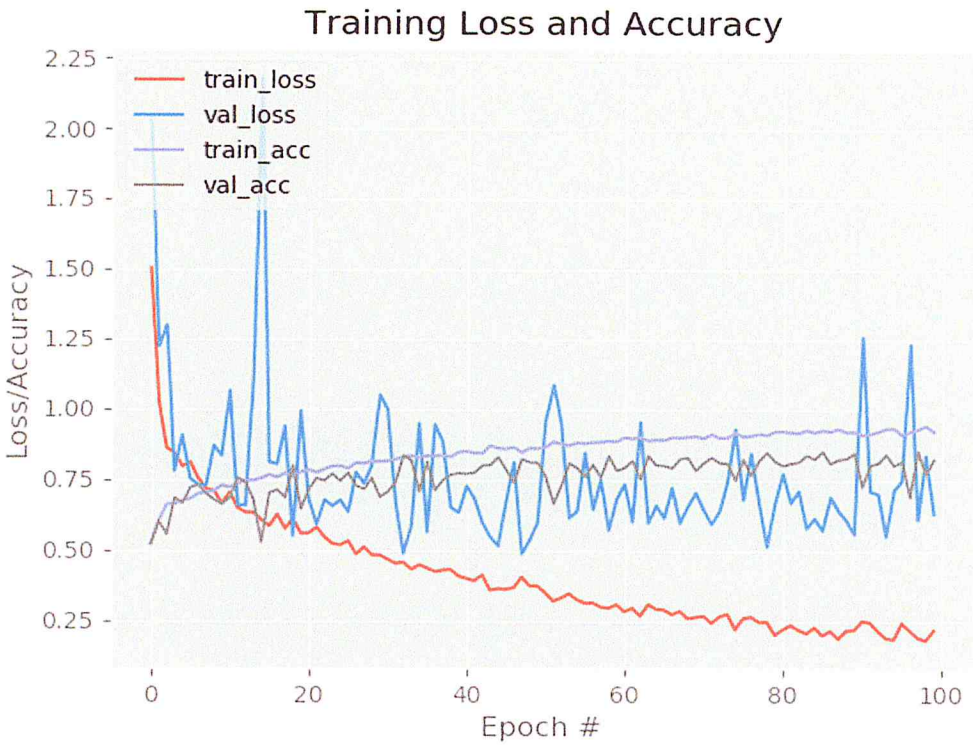


Figure 47: Précision et Erreur pour le Modèle 1

Après l'analyse des résultats obtenus, On constate les remarques suivantes :

D'après **la Figure 46.** La précision de l'apprentissage et de teste augmente avec le nombre d'époque, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations.

L'erreur d'apprentissage diminue en fonction de nombre d'époque. Par contre, l'erreur du teste a montré un sur- apprentissage, cela est du à la taille modeste de la base de teste (5600 images). Donc, pour faire apprendre notre modèle, on aura besoin de plus d'information en augmentant la taille de base d'images pour faire apprendre notre modèle.

4.1.2. Résultat du modèle 2

La figure suivante (**Figure 47**) montre la précision et l'erreur pour le deuxième modèle 2 il est appliqué sur la base de donné Dataset2.

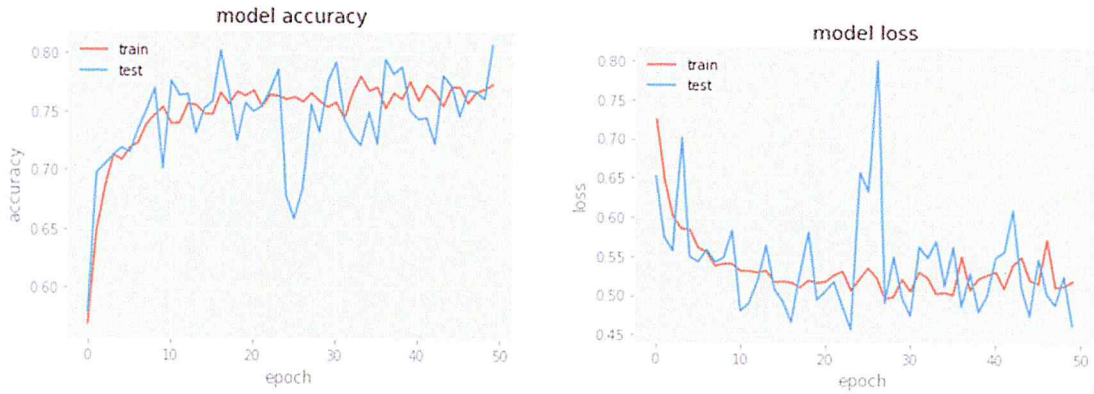


Figure 48: Précision et Erreur pour le Modèle 2

D'après la **Figure 47** nous pouvons voir qu'il y a une différence considérable entre la perte d'entraînement et de teste entre 24 e 26 époque. Cela indique que le réseau a essayé de mémoriser les données d'apprentissage et est ainsi en mesure d'obtenir une meilleure précision sur celui-ci. C'est un signe d'*Overfitting*. De même pour le graphe de la précision on voit que la précision de l'apprentissage et de teste augmente avec le nombre d'époque ceci reflète qu'à chaque époque le modèle apprenne plus d'informations, par contre entre 24 e 26 époque nous voyons qu'il y a une différence considérable entre la précision d'entraînement et de teste

4.1.3. Résultat du modèle 3

La figure suivante représente les valeurs de la précision et d'erreur pour le modèle3, il est appliqué sur la base de données Dataset3.

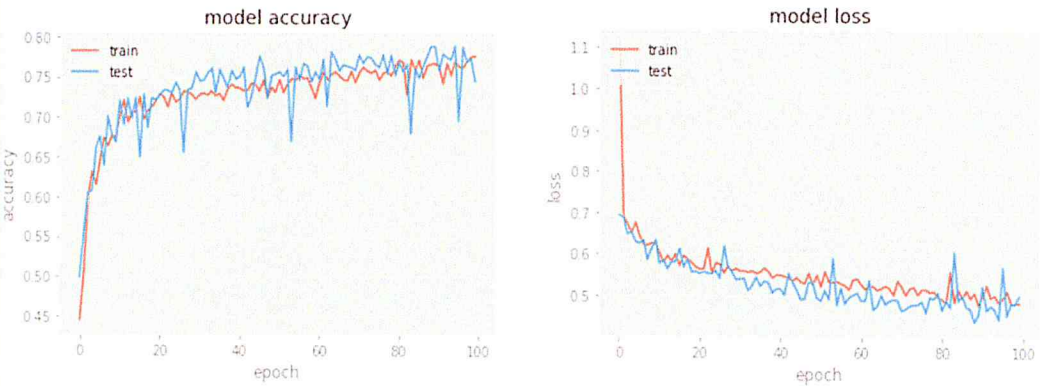


Figure 49: Précision et Erreur pour le Modèle 3

D'après **la Figure 48** la précision de l'apprentissage et de teste augmente en parallèle avec le nombre d'époque, cela veut dire qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors nous aurons besoin de plus d'information pour faire apprendre notre modèle et par conséquent nous devons augmenter le nombre d'époque.
De même, l'erreur d'apprentissage et de teste diminue en parallèle avec le nombre d'époque.

4.1.4. Résultat du modèle 4

La figure ci-dessous représente les valeurs de la précision et d'erreur pour le modèle4.

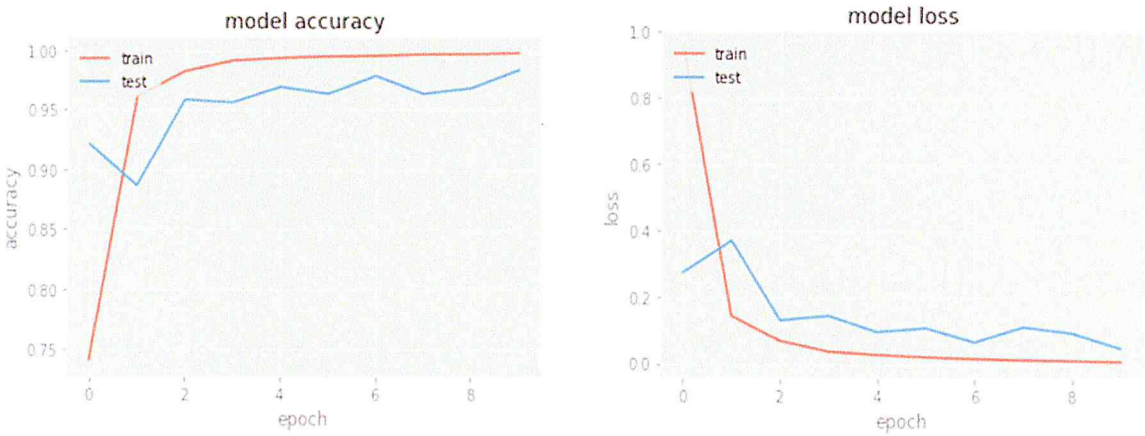


Figure 50: Précision et Erreur pour le Modèle 4

La Figure 49 montre que la précision de l'apprentissage (à gauche) augmente avec le nombre d'époque, de même avec le teste à partir de la deuxième époque après une diminution entre l'époque un et deux, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision de teste à diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter la taille de la base d'images ainsi que le nombre d'époque.

De même, l'erreur d'apprentissage (à droite) diminue avec le nombre d'époque, le teste marque une augmentation entre l'époque un et deux après on voit une diminution remarquable en fonction de nombres d'époque.

4.2. Tableau de comparaison des résultats

Le tableau 10 montre l'architecture utilisée dans chaque modèle ainsi que le batch size et le nombre d'époques. Les résultats obtenus sont exprimés en termes d'erreur te précision d'apprentissage et de teste et de temps d'exécution par époque.

	Architecture Utilisé			Batch Size	Nombre D'epoch	Erreur D'apprentissage	Précision d'apprentissage	Erreur De test	Précision de test	Temps/ epoch
	Couche de Convolution	Couche de Pooling	Fully Connected							
Modèle1	5	3	2	32	100	14.46%	94.49%	63.06%	82.19%	541s
Modèle 2	3	3	2	16	50	56.16%	79.23%	66.25%	79.31%	389s
Modèle 3	5	3	2	256	100	45.78%	78.22%	47.03%	77.19%	46s
Modèle 4	6	4	2	128	10	0.10%	99.98%	4.78%	98.40%	2182s

Tableau 5: Résultats des différents modèle CNN

On voit d'abord que le temps d'exécution est trop couteux. Ceci revient à la grande dimension des bases d'images. Le modèle 4 a présenté les meilleurs résultats avec une précision de 99.98 % et 0.10% d'erreur.les couches de convolution reflètent ces bons résultats, cependant le temps d'exécution était très couteux (à cause du volume de base d'image).

D'une manière générale, un réseau de neurone convolutionnel important et profond donne des bons résultats et la performance de notre réseau se dégrade si une couche convolutive est supprimée. Par exemple, d'après le tableau ajouter une couches de convolution intermédiaires la précision a augmenter de 5%. Donc, la profondeur est primordiale pour atteindre des bons

résultats. Les résultats obtenus se sont améliorés à mesure que nous avons approfondie notre réseau et augmenté le nombre d'époque. La base d'apprentissage est également un élément déterminant dans les réseaux de neurones convolutionnels, il faut avoir une base d'apprentissage de grande taille pour aboutir à des meilleurs résultats.

4.3. L'augmentation des données

L'augmentation de données est une solution efficace pour éviter le problème de sur apprentissage nous allons l'introduire dans notre réseau et voir comment il impacte les performances.

Pour obtenir plus de données, il suffit d'apporter des modifications mineures à notre ensemble de données existant, telles que les retournements ou les traductions ou les rotations. Notre réseau de neurones penserait que ce sont des images distinctes de toute façon.

La Figure 50 montre un échantillon d'images générées par l'augmentation d'image,

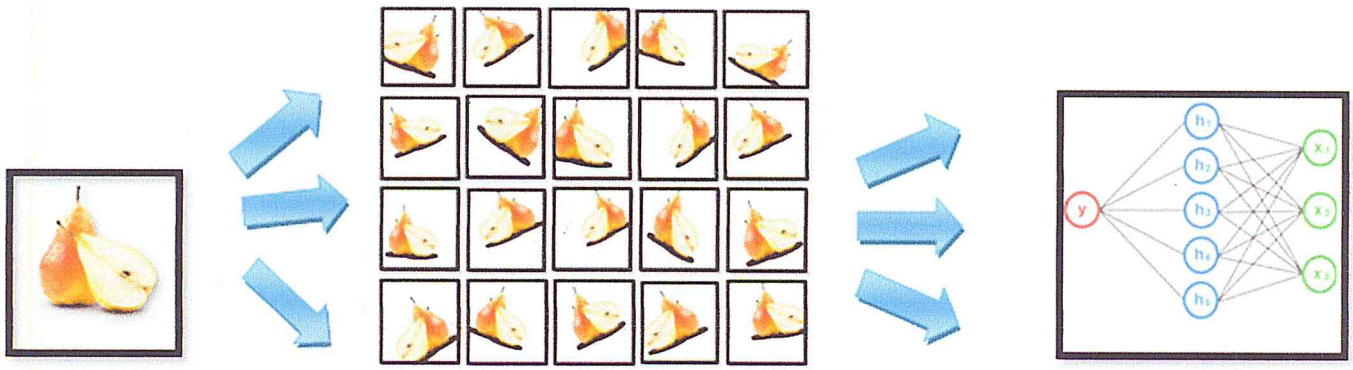


Figure 51: Augmentation des données en jeu

Dans le scénario du monde réel, nous pouvons avoir un ensemble de données d'images prises dans un ensemble limité de conditions. Mais, notre application cible peut exister dans une variété de conditions, telles que l'orientation, l'emplacement, l'échelle, la luminosité, etc. Nous prenons en compte ces situations en formant notre réseau neuronal avec des données modifiées de manière synthétique.

On applique l'augmentation de donné sur la base de donné Dataset1

La figure suivante représente les valeurs de la précision et d'erreurs obtenues après l'application d'augmentation des données.

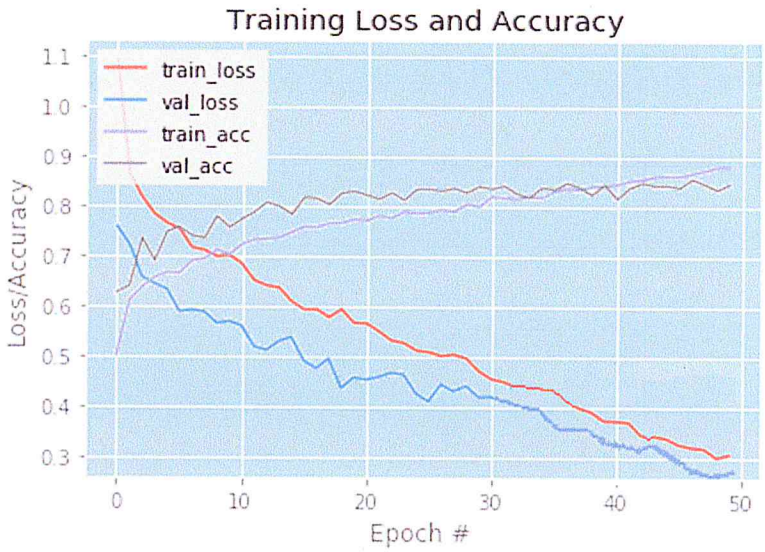


Figure 52: Précision et Erreur pour le Modèle1 avec l'augmentation des données

A partir des courbes de perte et de précision ci-dessus, nous pouvons observer que la perte de validation n'augmente pas.

La différence entre la précision de l'entrainement et de teste n'est pas très élevée. Ainsi, nous pouvons dire que le modèle a une meilleure capacité de généralisation, car les performances ne diminuent pas.

4.4. Résultats de classification obtenus avec le réseau de neurones simple

La figure suivante **(Figure 52)** montre les valeurs de la précision et d'erreur pour le modèle RNN.

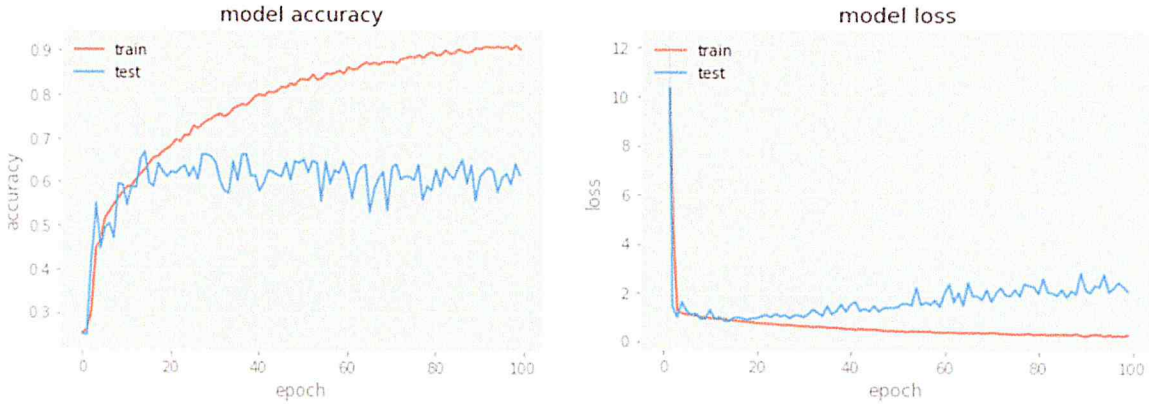


Figure 53: Précision et Erreur pour le Modèle RNN

Les résultats montrés dans la **Figure 52** indiquent que la précision obtenue est très bonne, mais on remarque que les courbes de perte de validation (teste) diminuent initialement, mais ensuite elle commence à augmenter progressivement. De plus, il existe une différence substantielle entre la précision de l'entraînement et celle du teste. Ceci est un signe évident d'*Overfitting*, ce qui signifie que le réseau a très bien mémorisé les données d'entraînement, mais qu'il n'est pas garanti de travailler sur des données non vues.

Nous allons introduire le *Dropout* dans notre réseau et voir comment il impacte les performances.

La figure suivante **(Figure 53)** représente les valeurs de la précision et d'errer pour le modèle ANN avec le *Dropout.*

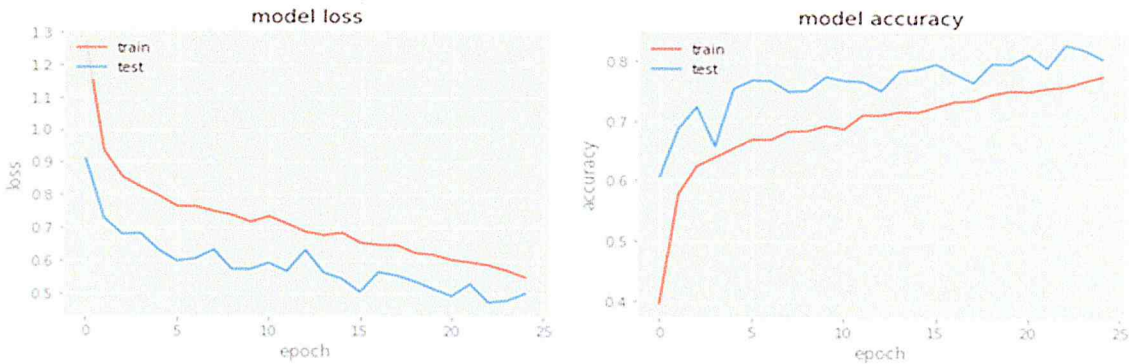


Figure 54: Précision et Erreur pour le Modèle ANN avec *Dropout*

A partir des figures 52 et 53 on voit tout de suite que le modèle avec *Dropout* réalise une performance largement meilleure que le modèle sans *Dropout*. De même l'erreur est tout aussi rapidement minimisée même avec des neurones en moins lors de l'apprentissage (l'effet du *Dropout*) à la seule différence que : avec *Dropout*, l'erreur continue d'être minimisée pour atteindre sa plus faible valeur après 25 époques pour ne plus diverger après. L'effet se voit tout aussi bien sur la précision où le pourcentage de bonne classification atteint et de 80% après 25 époques.

	Batch size	Nombre D'époques	Erreur D'apprentissage	Précision D'apprentissage	Erreur De teste	Précision De Teste	Temps /époque
Modèle 1	32	100	0.30	86.84%	2.13	61.80%	24s
Modèle 2	32	25	0.54	80.14%	0.55	82.16%	47s

Tableau 6: Résultats du Modéle 1(sans *Dropout*) et Modèle 2 (avec *Dropout*)

D'après le **Tableau 6** nous constatons que nous obtenons 86,84% de précision sur les données d'entraînement et 61,80% de précision sur les données de teste pour le modèle 1. L'effet se voit dans l'erreur de teste avec *Dropout* on marque une erreur de 0.55 dans le Modèle 2 alors que pour le Modèle 1 qui est entrainer sans *Dropout* on marque une erreur de 2.13

4.5. Résultats de classification obtenus avec l'algorithme K-NN

Nous avons appliqué l'algorithme K-NN sur la base d'image Dataset1, Dans cette base on a partitionné 60% de la totalité d'mages pour l'apprentissage et 40% pour le teste.
Le tableau suivant montre la matrice de confusion obtenue par l'algorithme K-NN.

	Banane	Fraise	Orange	Poire	
Banane	92	34	27	113	266
Fraise	8	170	29	63	270
Orange	14	47	194	45	300
Poire	64	45	23	218	350
	178	296	273	439	N=1186

Tableau 7 : Matrice de confusion pour L'algorithme K-NN

La matrice de confusion montré dans **Tableau 7** , indique que dans les 178 bananes réels, le classificateur K-NN a prédit que 8 étaient des fraises, 14 étaient des oranges et 64 étaient des poire, et dans les 296 fraises, il a prédit que 34 étaient des bananes, 170 étaient fraises,47 étaient des oranges et 45 étaient des poire, pour les 273 oranges, il a prédit que 27 étaient des bananes, 29 étaient des fraises,194 étaient des oranges et 23 étaient des poires, en dernier pour la totalité des 439 poires , il a prédit que113 étaient des bananes, 63 étaient des fraises,45 étaient des oranges et 218 étaient des poires.

$OvAc = (92 + 170 + 194 + 218) / 1186 \approx 0{,}658$

Nous pouvons voir à partir de la matrice que le classificateur en question a du mal à faire la distinction entre les bananes et les poire, mais peut faire la distinction entre les fraises et d'autres types de fruits plutôt bien, de même pour les oranges avec d'autres types.

La figure 54 représente la précision de teste en fonction de la valeur de K

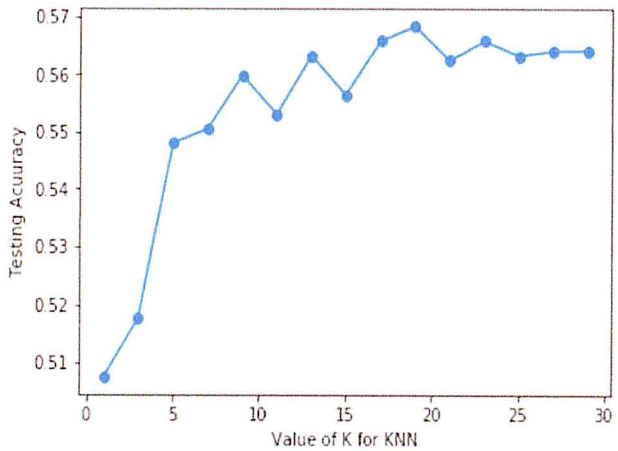


Figure 55: Précision de teste en fonction de la valeur de K

La **Figure 54** montre que la précision de l'ensemble de teste varie en fonction de la valeur de k. Quand la valeur de K = 19 l'algorithme atteint la plus haute précision qui est égale à 56,83% sur les données de teste.

La **Figure 55** exprime l'erreur de teste en fonction de la valeur de K.

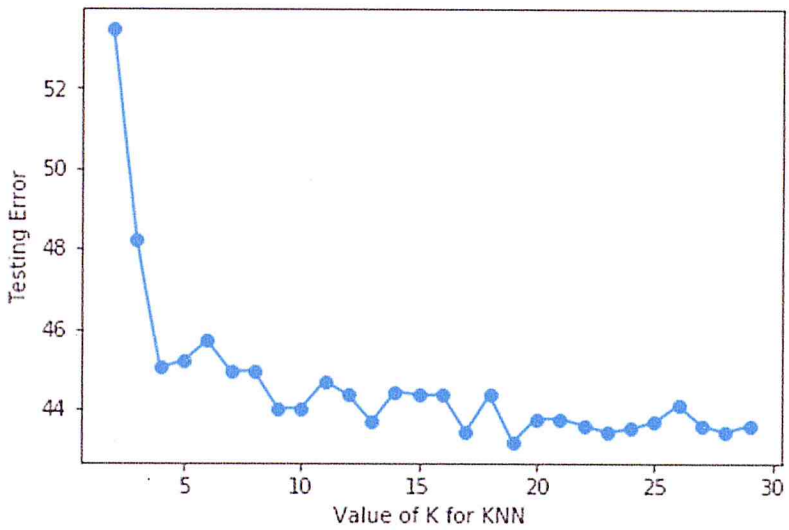


Figure 56: L'erreur de teste en fonction de valeur de K

Cette figure (**Figure 55**) montre que l'erreur de l'ensemble de testes varie en fonction de la valeur de k. Nous testeons le modèle pour tous k de 2 à 15.

Comme on peut le voir, le KNN le plus performant est celui pour lequel k = 19. On connaît donc notre classifieur final optimal : 19-nn. Ce qui veut dire que c'est celui qui classifie le mieux des données, et qui donc dans ce cas reconnaît au mieux les types des fruits.

Évaluation sur les données de teste

Le **Tableau 8** représente l'évaluation sur les données du teste.

	précision	Rappel	f1-score	Support
Banane	0.54	0.33	0.41	266
Fraise	0.56	0.61	0.58	270
Orange	0.69	0.64	0.67	300
poire	0.50	0.64	0.56	350
avg / total	0.57	0.56	0.56	1186

Tableau 8: Évaluation sur les données du teste

Le tableau 8 montre une **précision** égale à **57%,** nous pouvons voir que le fuit orange obtient la meilleur précision de 69% et le fruit poire obtient la précision de classification la plus faible avec un pourcentage de précision égale à 50%.

4.6. Résultats de classification obtenus avec l'algorithme SVM

Le tableau suivant représente les résultats de la précision et du rappel obtenus d'après l'algorithme SVM.

	Précision	rappel	f1-score	support
Banane	0.56	0.60	0.58	430
Fraise	0.79	0.83	0.81	430
Orange	0.77	0.66	0.71	430
poire	0.59	0.60	0.60	430
avg / total	0.68	0.67	0.67	1720

Tableau 9: Précision et rappel

Le **Tableau 9** montre une **précision de 68%,** nous pouvons voir que le fuit fraise obtient la meilleure précision de 79% et le fruit poire obtient la précision de classification la plus faible avec 59%.

Le Tableau 10, montre la matrice de confusion retourné par le classificateur SVM.

	Banane	Fraise	Orange	Poire	
Banane	259	14	49	108	430
Fraise	30	357	14	29	430
Orange	47	58	282	43	430
Poire	124	24	23	259	430
	460	453	368	439	N=1720

Tableau 10: Matrice de confusion

On voit que dans les 460 bananes réels, le classificateur SVM a prédit que 259 étaient des banane, 30 étaient des fraises, 47 étaient des oranges et 124 étaient des poire, et dans les 453 fraises, il a prédit que 14 étaient des bananes, 357 étaient fraises,58 étaient des oranges et 24 étaient des poire, pour les 368 oranges, il a prédit que 49 étaient des bananes, 14 étaient des fraises, 282 étaient des oranges et 23 étaient des poires, en dernier pour la totalité des 439 poires , il a prédit que 108 étaient des bananes, 29 étaient des fraises, 43 étaient des oranges et 259 étaient des poires.

Nous pouvons voir à partir de la matrice que le classificateur en question a du mal à faire la distinction entre les bananes et les poire, mais peut faire la distinction entre les fraises et d'autres types de fruits plutôt bien, de même pour les oranges avec d'autres types.

$$\text{OvAc} = (259 + 357 + 282 + 259) / 1720 \approx 0{,}672$$

4.7. Résultats obtenus par la détection

YOLO a été implémenté à l'origine en utilisant *Darflow*, un framework de réseau de neurones open-source écrit en C et CUDA. Pour notre implémentation de YOLO, nous avons utilisé *Tensorflow*, une librairie open-source pour développée par Google et *Keras*, une API de réseaux neuronaux de haut niveau, écrite en *Python* est capable de fonctionner sur *TensorFlow* ou *Theano*.

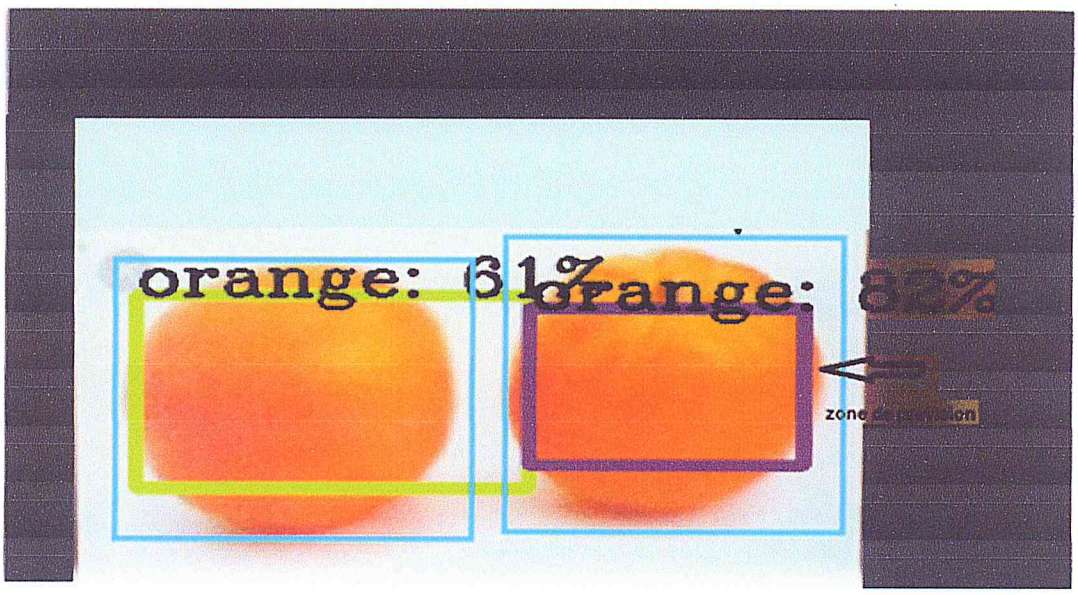


Figure 57: Faible IoU

Notre premier exemple d'image a un faible score d'intersection sur union ,indiquant qu'il y a un chevauchement (*Overlap*) significatif entre les deux boîtes englobantes, de plus le cadre de sélection de la vérité au sol (bleu) est plus large que le cadre de sélection prévu (vert et marron).

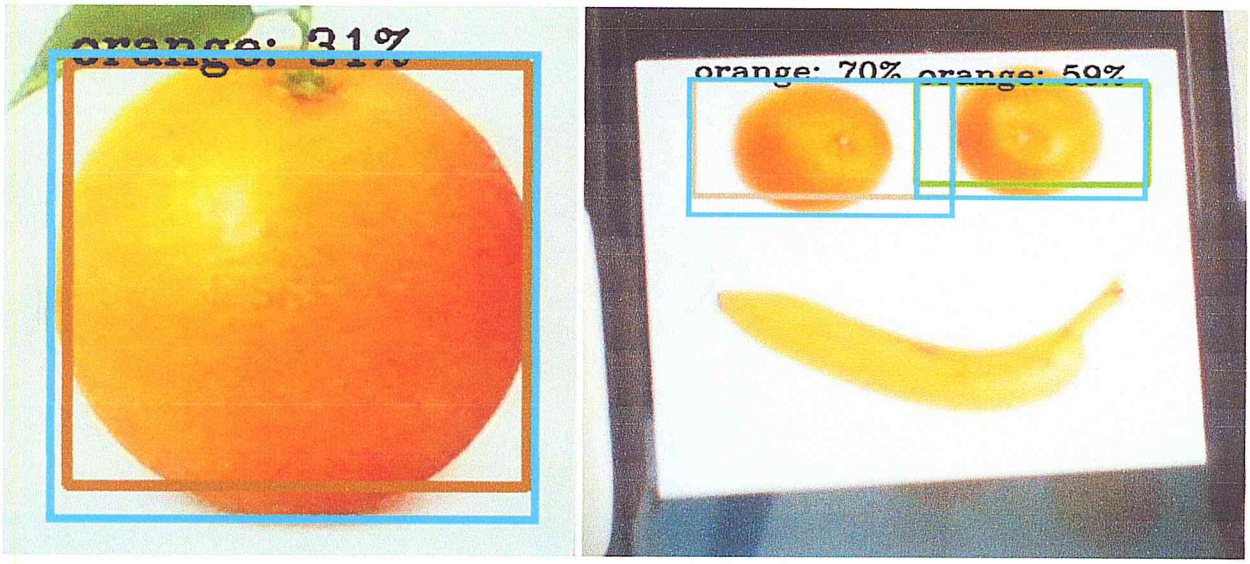


Figure 58: Bonne IoU

La Figure ci-dessus montre un exemple de bonne détection, nous remarquons un rapprochement entre la boite englobante et celle de la boite de vérité de sol.

Figure 59: Excellent IoU

Notre troisième exemple montré dans la figure ci-dessus indique une excellente détection, notons que la boîte englobante prédite se superpose presque parfaitement avec la boîte englobante de la vérité au sol.

Conclusion

La détection des fruits permettra par la suite le comptage des fruits pour prédire le rendement .

Nous avons présenté dans ce chapitre de différentes approches de classification : l'approche CNN avec quatre modèles et de différentes architectures, l'approche SVM comme aussi l'approche K-NN, et finalement la méthode YOLO et on a montré les différents résultats obtenus en termes de précision et d'erreur.

En comparant les résultats obtenus on trouve que la taille de la base d'image, le choix des paramètres (époque pour le CNN l'ANN et YOLO, K pour le K-NN) et la profondeur du réseau sont des facteurs importants pour l'obtention de meilleurs résultats.

V. CONCLUSION ET FUTURE PERSPECTIVES

Conclusion générale

Nous avons introduit dans ce mémoire les notions de base qui servent de fondement à la compréhension de différentes techniques de classification des fruits. Plusieurs méthodes ont été proposées dans la littérature, nous avons présenté quelques unes qui nous semble les plus courantes dans le processus de la classification et la détection des fruits.

Les paramètres du réseau sont difficiles à définir a priori. C'est pour cette raison que nous avons défini différents modèles avec différentes architectures afin d'obtenir des meilleurs résultats en terme de précision et d'erreur.

La mise en œuvre des trois algorithmes de classification tel que K-NN, SVM et réseau de neurones simple peut être considérée comme satisfaisante, mais en terme de précision le CNN a marqué des meilleurs résultats malgré que nous avons utilisé une taille minimale de la base d'apprentissage, mais, il est couteaux par rapport en temps d'exécution.

Nous avons rencontré quelques problèmes dans la phase d'implémentation, l'utilisation d'un CPU a fait que le temps d'exécution était trop couteux spécialement pour la méthode YOLO qui a entrainé une base d'image et des paramètres minimales dans une durée d'une semaine voir plus. Afin de régler ce problème et avoir de meilleurs résultats il est recommandé d'utiliser des machines obtenant des GPU puissants au lieu d'un CPU, et pour généraliser notre travaille on doit appliquer l'entrainement YOLO sur des bases plus importantes en utilisant plusieurs classes d'images.

Pour améliorer ce travail on peut par la suite réaliser un comptage des cultures agricoles dans des images des arbres prises par le satellite ou par l'être humain pour donner plus d'information au agriculteurs afin de faire une estimation de rendement et par la suite effectuer une gestion spécifique au site basée sur la cartographie des rendements réduit les coûts de main-d'œuvre pour la culture de gestion et la récolte, et optimise la quantité de matériaux requis tels que les engrais et les produits chimiques agricoles.

VI. Bibliographie

[1] E. Arnaud, E. B. Universit, J. Fourier, and I. Rh, "Analyse d ' images – introduction," p. 30.

[2] S. Bargoti and J. P. Underwood, "Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards arXiv : 1610 . 08120v1 [cs . RO] 25 Oct 2016," 2006.

[3] Materiel-informatique.be, "Codage RVB - RGB.," 2015. .

[4] futurascience, "Définition | Pixel - Picture element | Futura Tech." .

[5] stemmer imaging, "Grey level / Grey value | STEMMER IMAGING." .

[6] C. Gamma, "5.1.1 Histogrammes : définition et utilité."

[7] U. Biskra, "Chapitre 03 Généralités sur le traitement d'images Introduction," *Caracteristique d'image*, 2014. .

[8] C. Rouge, J. Vert, and B. Magenta, "La couleur dans le système HSV 1.," pp. 1–12.

[9] H. Fethallah, M. Mohammed, B. Akkacha, and S. M. Ismail, "Remerciements," 2017. .

[10] Commentcamarche, "Comment ça marche : le bruit d'image.", 2017 .

[11] supinfo, "Traitement d'image, détection des contours par filtre | SUPINFO, École Supérieure d'Informatique," 2015. .

[12] blogs.msdn.microsoft, "Evaluer un modèle en apprentissage automatique – Big Data France," 2014. .

[13] D. M. W. POWERS, "Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[14] eyraud remy, "Classification, Apprentissage, Décision Rémi Eyraud Troisième cours."

[15] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote Sens. Environ.*, vol. 62, no. 1, pp. 77–89, Oct. 1997.

[16] "Classification accuracy assessment. Confusion matrix method." [Online].

Available: http://www.50northspatial.org/classification-accuracy-assessment-confusion-matrix-method/. [Accessed: 06-Oct-2018].

[17] A. Machine, S. Exploratoire, A. Statistique, I. Artifi-, and A. Machine, "Apprentissage Machine / Statistique," 2001. .

[18] math.univ-angers, "Classification supervisée."

[19] "Realism and Instrumentalism: Classical Statistics and VC Theory (1960–1980)," in *Estimation of Dependences Based on Empirical Data*, Springer New York, pp. 411–424.

[20] J. Mercer and J., "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 209, no. 441–458, pp. 415–446, Jan. 1909.

[21] "In machine learning, how can we determine whether a problem is linear/nonlinear? - Quora." [Online]. Available: https://www.quora.com/In-machine-learning-how-can-we-determine-whether-a-problem-is-linear-nonlinear. [Accessed: 06-Oct-2018].

[22] J.-L. Verley, *Dictionnaire des mathématiques; algèbre, analyse, géométrie*. Albin Michel, 1997.

[23] "Pseudocode for KNN classification. | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/Pseudocode-for-KNN-classification_fig7_260397165. [Accessed: 06-Oct-2018].

[24] mathunictoulouse, "mathunivtoulouse," 2012.

[25] "Architecture d'un réseau multicouches. | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/Architecture-dun-reseau-multicouches_fig1_290394554. [Accessed: 06-Oct-2018].

[26] Openclassrooms, "Qu'est ce qu'un réseau de neurones convolutif (ou CNN) ? - Classez et segmentez des données visuelles - OpenClassrooms." .

[27] blogocto, "Classification d'images : les réseaux de neurones convolutifs en toute simplicité | OCTO Talks !" .

[28] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation." pp. 2643–2651, 2013.

[29] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning."

[30] medium, "Comment les Réseaux de neurones à convolution fonctionnent."

[31] Github, "'Convolutional neural networks (CNN) tutorial,'" 2017. .

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[33] "(PDF) Combinaison de descripteurs globaux et locaux pour la recherche par le contenu d'informations visuelles." [Online]. Available: https://www.researchgate.net/publication/309772777_Combinaison_de_descripteurs_globaux_et_locaux_pour_la_recherche_par_le_contenu_d%27informations_visuelles. [Accessed: 15-Sep-2018].

[34] L. Bottou, Y. Bengio, and Y. Le Cun, "Global Training of Document Processing Systems using Graph Transformer Networks."

[35] "Les Enjeux de la Recherche en Intelligence Artificielle Yann LeCun Directeur , Facebook AI Research Chaire Informatique et Sciences Numériques Qu ' est-ce que l ' Intelligence Artificielle L ' apprentissage machine," pp. 1–7, 2016.

[36] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2014.

[37] W. Marx, "Tracking historical papers and their citations," *Eur. Sci. Ed.*, vol. 38, no. 2, pp. 35–37, 2012.

[38] J. P. Wachs, H. I. Stern, T. Burks, and V. Alchanatis, "Low and high-level visual feature-based apple detection from multi-modal images," *Precis. Agric.*, vol. 11, no. 6, pp. 717–735, 2010.

[39] P.-Y. Le Guilloux, "L'éclaircissage du pommier : la métamitrone, un outil d'avenir ?"

[40] L. Macaire, N. Vandenbroucke, and J.-G. Postaire, "Segmentation d'images par classification spatio-colorimétrique des pixels Image segmentation by spatio-colorimetric classification."

[41] A. Syal, D. Garg, S. Sharma, and A. Professor, "A Survey of Computer Vision Methods for Counting Fruits and Yield Prediction," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 06, pp. 2319–7323, 2013.

[42] S. Bargoti and J. P. Underwood, "Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards," *J. F. Robot.*, 2017.

[43] S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh,

"Automated Visual Yield Estimation in Vineyards," *J. F. Robot.*, vol. 31, no. 5, pp. 837–860, Sep. 2014.

[44] K. Yamamoto, W. Guo, Y. Yoshioka, and S. Ninomiya, "On Plant Detection of Intact Tomato Fruits Using Image Analysis and Machine Learning Methods," pp. 12191–12206, 2014.

[45] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Ben Shahar, "Computer vision for fruit harvesting robots – state of the art and challenges ahead," *Int. J. Comput. Vis. Robot.*, vol. 3, no. 1/2, p. 4, 2012.

[46] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated Crop Yield Estimation for Apple Orchards," 2012.

[47] C. W. Bac, J. Hemming, and E. J. van Henten, "Robust pixel-based classification of obstacles for robotic harvesting of sweet-pepper," *Comput. Electron. Agric.*, vol. 96, pp. 148–162, Aug. 2013.

[48] A. R. Jiménez, R. Ceres, and J. L. Pons, "A Survey of Computer Vision Methods for Locating Fruit on Trees," 1911.

[49] D. Font, M. Tresanchez, D. Martínez, J. Moreno, E. Clotet, and J. Palacín, "Vineyard yield estimation based on the analysis of high resolution images obtained with artificial illumination at night.," *Sensors (Basel).*, vol. 15, no. 4, pp. 8284–301, Apr. 2015.

[50] M. Chhabra, A. Gupta, P. Mehrotra, and S. Reel, "Automated Detection of Fully and Partially Riped Mango by Machine Vision," Springer, India, 2012, pp. 153–164.

[51] R. Linker and E. Kelman, "Detection of apples in nighttime orchard images using the geometry of light patches."

[52] P. Li, S. Lee, and H.-Y. Hsu, "Study on citrus fruit image data separability by segmentation methods," *Procedia Eng.*, vol. 23, pp. 408–416, Jan. 2011.

[53] P. Wijethunga, S. Samarasinghe, D. Kulasiri, and I. Woodhead, "Towards a generalized colour image segmentation for kiwifruit detection," 2009.

[54] F. Kurtulmus, W. S. Lee, and A. Vardar, "Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and neural network," *Precis. Agric.*, vol. 15, no. 1, pp. 57–79, Feb. 2014.

[55] A. Payne, K. Walsh, P. Subedi, and D. Jarvis, "Estimating mango crop yield using image analysis using fruit at 'stone hardening' stage and night

time imaging," *Comput. Electron. Agric.*, vol. 100, pp. 160–167, Jan. 2014.

[56] "Modelling Apple Fruit Yield Using Image Analysis for Fruit Colour, Shape and Texture | Request PDF." [Online]. Available: https://www.researchgate.net/publication/282407024_Modelling_Apple_Fruit_Yield_Using_Image_Analysis_for_Fruit_Colour_Shape_and_Texture. [Accessed: 09-Sep-2018].

[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks."

[58] "Review of Deep Learning Algorithms for Object Detection." [Online]. Available: https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852. [Accessed: 16-Sep-2018].