UNIVERSITÉ DE MONTPELLIER

SOWIT

AMAP

UR HortSys

cirad
LA RECHERCHE AGRONOMIQUE POUR LE DÉVELOPPEMENT

Développement d'un réseau de neurones pour la détection de mangues sous environnement mobile

DEPARTEMENT D'ENSEIGNEMENT EEA
ELECTRONIQUE - ELECTROTECHNIQUE - AUTOMATIQUE

NACHITE Ayoub

Master 2 EEA – Systèmes Électroniques Intégrés

Année Scolaire :

2019/2020

Responsable Pédagogique :

VIRAZEL Arnaud

Encadré par :

BORIANNE Philippe (UMR Amap)
FAYE Émile (UPR HortSys)

Remerciements

Je tiens à remercier Borianne Phillippe pour son suivi, ses conseils préciseux et son encadrement durant toute la durée du stage que ça soit en présentiel ou distanciel, je remercie aussi Emile Faye , Julien Sarron et Hamza Bendahou pour leur suivi durant les différentes réunions de travail.

Merci aussi à ma collègue du bureau 18 Laetitia pour son soutien et son aide pratique, et à tous le membres de AMAP qui ont pris du temps pour m'intégrer et me conseiller durant toute la période où j'étais en présentiel.

Abstract

This work is part of a 6-month internship at CIRAD, consisting in developing a neural network for mangoes detection under mobile environment, the network used is the Tiny Yolo v2 dedicated to boarding via the TensorFlow framework.

Keywords : Neural Networks, Mangoes, Yolov2 , TinyYolov2, Python , TensorFlow

Sommaire

I-Introduction

Les Réseaux de neurones ont pu apporter en peu de temps une grande valeur ajoutée dans pleins de domaines d'application, ainsi que des solutions très intéressantes en surpassant même l'expertise humaine sur quelques taches.

L'agriculture fait partie des domaines qui pourraient bénéficier de l'apport considérable des réseaux de neurones afin d'améliorer les estimations de rendement des exploitations et des technologies usuelles utilisées durant les différentes activités liées à la culture et la production (smart farming [1]), et c'est dans ce cadre que le stage intitulé « développement d'un réseau de neurones pour la détection de mangues sous environnement mobile » est réalisé, afin d'apporter une solution aux problèmes d'estimation que rencontrent les cultures de fruits, et notamment les mangues en Afrique de l'ouest.

Les Réseaux de neurones, avec leur capacité de traiter des quantités importantes de données visuelles, sont les candidats parfaits pour offrir aux agriculteurs des moyens efficaces pour estimer le rendement de leur exploitation avec facilité, et remédier au manque de fiabilité des technologies utilisées auparavant pour l'estimation.

II-Présentation du stage.

1-Organismes encadrant le stage :

CIRAD / UR HORTSYS / UMR AMAP :

Le **CIRAD** (Le Centre de coopération internationale en recherche agronomique pour le développement) est un établissement public créé en 1984, à caractère industriel et commercial. Le siège social du CIRAD se situe à Paris, mais ses deux centres de recherche en France sont situés à Montpellier (Campus de Lavalette) et Montferrier-sur-Lez (Campus de Baillarguet) ; le CIRAD comprend 800 chercheurs qui travaillent en partenariat avec des acteurs du monde de la recherche en France et sur trois continents.

Le CIRAD participe au développement durable des pays tropicaux et méditerranéens, en mettant à leur service son expertise dans les sciences appliquées à l'agriculture, et en axant le travail sur des thématiques comme le changement climatique et la lutte contre les inégalités.

Le partage des connaissances et la contribution de leur diffusion pour les pays du sud sont à ce jour des missions primordiales du CIRAD, puisqu'ils contribuent à faciliter l'accès pour ses partenaires du Sud aux programmes de recherche scientifique à travers les réseaux ou il est engagé que ça soit à l'échelle Européen ou International.

HortSys (Fonctionnement agro-écologique et performances des systèmes de culture horticoles) est une unité propre de recherche du CIRAD qui travaille sur l'horticulture et l'équilibre alimentaire mondial.

L'unité a comme mission alors de proposer des systèmes horticoles innovants qui se basent sur le fonctionnement agro-écologique dans le but de préserver l'environnement et limiter les risques liés aux écosystèmes et la santé humaine.

Les activités de l'unité sont conduites à Montpellier, Martinique et Réunion, ainsi que dans quelques pays d'Afrique de l'Ouest comme le Sénégal, le Bénin, Kenya et Madagascar.

AMAP (Botanique et modélisation de l'architecture des plantes et des végétations) est une unité mixte de recherche sous tutelle du CIRAD et d'autres organismes (CNRS, UM, INRAE et IRD) ; elle travaille sur le développement et l'évaluation des modèles d'analyse pour le suivi des paramètres du développement architectural d'espèces végétales isolées ou en peuplement.

Le travail de recherche de AMAP est axé sur la systématique et la phylogénie de végétaux actuels et fossiles, et la modélisation du fonctionnement des plantes ; l'unité créée aussi des logiciels permettant de diffuser ses méthodes auprès les chercheurs et les étudiants.

L'Imagerie des plantes et des paysages est l'un des importants axes de recherche d'AMAP, notamment pour mettre à disposition de l'écologie de la plante des méthodes de traitement d'images, l'expertise des chercheurs qui se consacrent à ce thème permet de fournir aux botanistes et agronomes des résultats et des outils facilités pour conduire leurs travaux.

SOWIT

SOWIT est une start-up spécialisée dans l'agriculture de précision, elle met à disposition des agriculteurs des solutions industrielles « métier » basées sur des algorithmes d'intelligence artificielle et traitement de l'image, à travers la mobilisation de plusieurs dispositifs comme les drones et les capteurs, permettant ainsi le suivi par les agriculteurs de leurs cultures ainsi que la prise décision au quotidien.

SOWIT travaille pour le développement de l'agriculture et mène des travaux en partenariat avec des acteurs dans le monde entier et notamment en Afrique.

2-Synthèse de la bibliogrphie

La Technologie est devenue de plus en plus présente ces dernières années en Agriculture, permettant aux exploitants de mieux suivre leurs productions ; cette démarche connue souvent sous le nom du « SMART FARMING » [1] (cf. fig. 1) mobilise des technologies avancées et souvent couteuses (drones, capteurs, IOT) [2] ; bien que très efficace et apportant des résultats et des améliorations significatifs, son déploiement auprès des petits agriculteurs n'ayant ni les moyens ni l'expertise pour manipuler ce type de protocoles s'avère compliqué.

Afin de répondre à cette problématique que ce stage intitulé « Développement de réseau de neurones pour la détection de mangues sous environnement mobile » est mené ; son objectif est d'exploiter des flux de données visuelles via l'usage de l'Intelligence Artificielle et notamment les Réseaux Neurones Profonds qui sont présents de plus en plus en Agriculture [3] , afin de faire la détection de mangues sur des images couleur, le stage qui rentre dans le cadre du projet Pixfruit qui a comme objectif d'offrir aux agriculteurs en Afrique de l'ouest une meilleure estimation de production des vergers de Mangues que l'inspection limité d'arbres, méthode peu précise et chronophage [4].

Les Réseaux de Neurones Artificiels (RNA) [5] font partie des méthodes de classification par apprentissage, utilisant des « neurones », i.e. des fonctions mathématiques transformant des valeurs d'entrée en sortie selon des règles précises ; les réseaux reposent sur l'agencement complexe et pertinent des neurones pour résoudre des problèmes linéaires et non linéaires [6].

Les RNA sont présents dans plusieurs domaines d'application, mais on s'intéresse plus particulièrement aux réseaux destinés au traitement d'images, et notamment aux Réseaux de Neurones de Convolution (CNN) [7] appartenant à la famille des Réseaux de Neurones profonds [8].

Le Réseau le plus connu dans la famille des CNN est le R-CNN [9] ; il a progressivement bénéficié d'évolutions significatives en termes d'efficacité, notamment avec le Fast-R-CNN [10] le Faster-R-CNN [11.a], et le mask-R-CNN [11.b] ; tous ces réseaux sont focalisés sur l'accélération du processus de détection afin de pouvoir traiter des données visuelles de taille de plus en plus grandes. Une grande partie des travaux réalisés ces dernières années sur la détection de fruits (pommes, mangues, etc.) [12] [13] par apprentissage automatique utilisent
le réseau Faster-R-CNN et obtiennent de très bonnes prédictions allant souvent de 0,80 jusqu'à 0,95 [14].

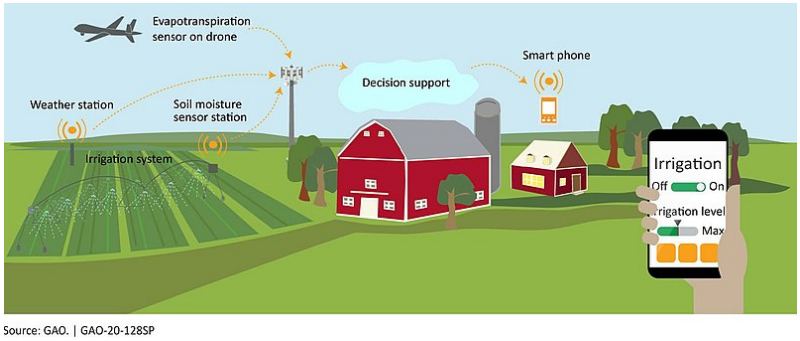


Figure 1 : Les différents composants du Smart Farming (Source : Wikimedia)

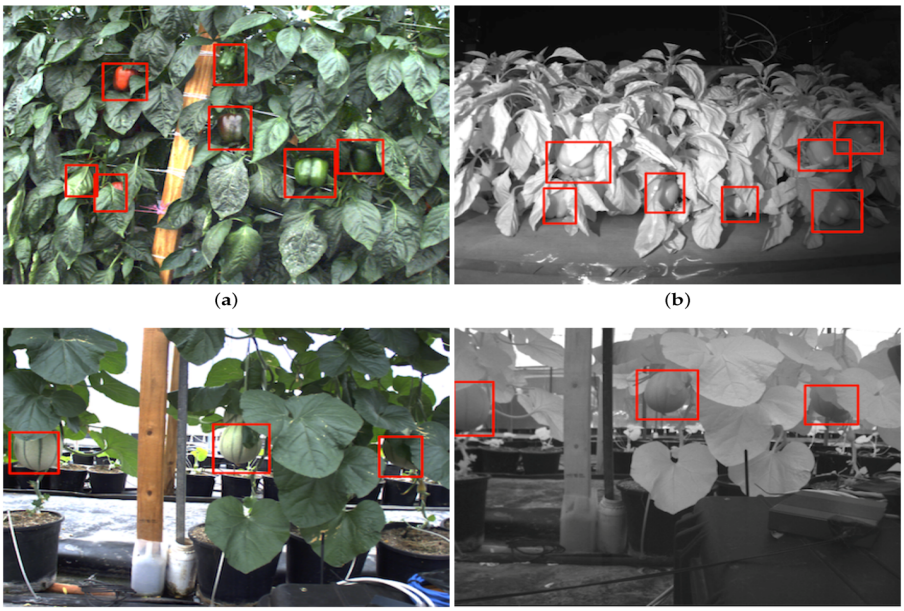


Figure 2 : Exemple de détection de poivrons et melons avec un Faster R-CNN [14]

Cependant, les R-CNN sont concurrencés par le Réseau YOLO (You Only Look Once) [15] qui, avec une structure lui permettant l'analyse des images en une seule étape (Single Shot) [16], lui procure plus de rapidité, et ainsi lui permet de faire de la détection de fruits en temps réel avec de bons résultats de prédiction [17] [18].

Figure 3 : Exemple de détection de mangues avec YOLO [17]

L'étude bibliographique, ainsi que l'objectif du portage du Réseau de Neurones sur Smartphone, nous poussent à faire de YOLO le CNN choisi pour ce travail, le Faster R-CNN nécessitant des étapes de calcul intensif et de redimensionnement incompatible avec les calculateur embarqué [19]. Différentes Versions de YOLO existent (Fulls et Tiny YOLO) [20] ; la version Tiny YOLO[21] semble destinée aux systèmes embarqués et donc adaptée pour une utilisation sur Smartphone [22] [23]. Différentes adaptions du Réseau sous

environnement TensorFlow sont disponibles [24] afin de simplifier le portage sous les Smartphones qui seront identifiés selon le besoin de YOLO en termes de Spécifications [25], grâce à la version mobile de TensorFlow (Lite) [26].

3-Contexte applicatif du stage

Le stage rentre dans le cadre du transfert technologique intitulée ‘’PixFruit’’ entre le CIRAD et SOWIT ; ce transfert a pour objectif de développer des outils d’estimation de la production de mangues.

Le but du stage est de développer un réseau de neurones permettant la détection des mangues à partir des photos (cf. fig. 4 et 5), afin de pouvoir l’embarquer en sur dispositif mobile, et ainsi fournir aux agriculteurs une solution simplifiée pour l’estimation du rendement de leurs manguiers.

Figure 4

Figure 5

Le Travail consistera alors pendant ce stage à déployer en premier lieu le réseau avec les deux versions de YOLO v2 (Full et Tiny) sur machine, pour l'embarquer en second lieu sur des émulateurs Android et quelques Smartphones mis à notre disposition après l'identification du matériel capable d'embarquer le Réseau YOLO ; par la suite une comparaison des performances du réseau sera réalisée sous ces différents environnements, ainsi que par rapport aux résultats obtenus avec le Faster R-CNN et la version 4 de la famille YOLO.

Les différents résultats du stage serviront de base pour une application smartphone qui va être développée dans le cadre du projet PixFruit.

III-Rappels

1-Rappels sur la classification et réseaux de neurones

L'intelligence artificielle (IA) regroupe l'ensemble des technologies et sciences mises en œuvre pour étudier des méthodes ayant pour but de créer des programmes intelligents capables de résoudre les problèmes, et ainsi simuler l'intelligence humaine en quelque sorte.

Le Machine Learning (ML) est un sous ensemble de l'IA, permettant la classification de données par apprentissage souvent supervisé, l'apprentissage est l'étape clé du ML, il permet l'extrapolation d'une loi à partir de la succession d'itérations d'un ensemble de données.

L'apprentissage permet d'éviter la programmation des règles utilisées par d'autres techniques et algorithmes classiques de classification, puisqu'au fil des itérations il s'adapte à la complexité de la tâche de la classification, et aux données afin de fournir un modèle de prédiction stable et généralisable.

Le Deep Learning (DL) ou apprentissage profond est un sous ensemble de l'IA aussi, correspondant à un ensemble des méthodes mathématiques, et permettant la résolution de problèmes complexes et leur compréhension : il repose essentiellement sur des modèles qui s'inspirent du cerveau humain appelés ((Réseaux de neurones artificiels)).

Réseaux de neurones artificiels (RNA) : Ils représentent une simplification du fonctionnement du réseau de neurones biologique basé sur les neurones et les synapses ; les neurones dans les RNA sont représentés par des fonctions mathématiques et leur empilement permet de simuler n'importe quel descripteur.

Dans sa forme basique, un RNA contient un seul neurone, modélisé par un système à N entrées et une seule sortie (cf. fig. 6).

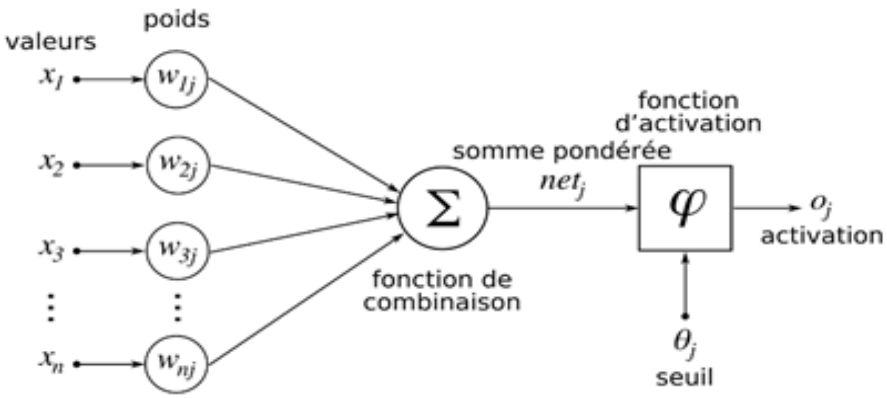


Figure 6 : Schéma du perceptron (source : Wikipédia)

Lors de l'apprentissage, un poids w est attribué à chacune des valeurs d'entrée, leurs produits sont sommés, et on se retrouve avec une valeur unique qui passera après par une fonction d'activation, on obtient alors un réseau de neurones constitué d'un seul neurone, appelé **Perceptron.**

Un perceptron simple ne peut jouer que le rôle d'un classifieur linéaire, deux classes uniquement peuvent être discriminés via une droite, d'où la nécessité du Perceptron multicouches.

Les **Perceptron multicouches** sont constitués de plusieurs couches (cf. fig. 7), où les informations sont traitées de façon unique de la couche d'entrée vers la couche de sortie.

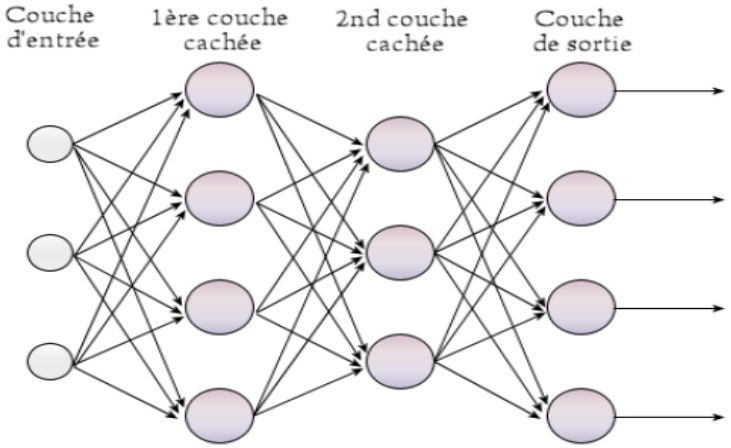


Figure 7 : Représentation d'un RNA multicouches d'une seule couche cachée (source : Wikipédia)

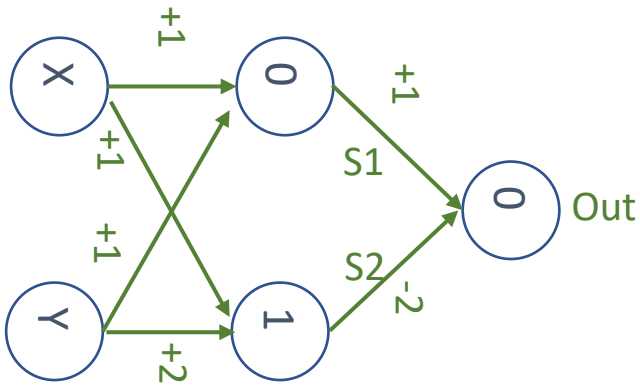


Figure 8 : Fonction XOR avec un RNA

Dans l'exemple de la fonction XOR (cf. fig. 8 et 9 et Table 1), on voit l'utilité de perceptron multicouches puisqu'avec un simple Perceptron le problème ne peut pas être résolu vu que les deux variables X et Y peuvent prendre deux valeurs 0 et 1, et comme le perceptron multicouches est doté de couches intermédiaires cachées, la contrainte de non séparabilité linéaire du perceptron monocouche n'est plus présente.

X	Y	S1	S2	Out
0	0	0	0	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	0

Table 1 : Table de vérité de XOR

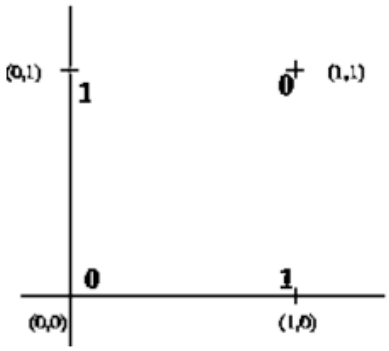


Figure 9 : Graphique de la fonction XOR (Source : Université de Lille)

Réseaux de neurones convolution (CNN) : Ils font partie de la famille des RNA profonds (plusieurs couches entre la couche d'entrée et la couche de sortie) et ils ont montré une grande efficacité pour le domaine du traitement d'images [27].

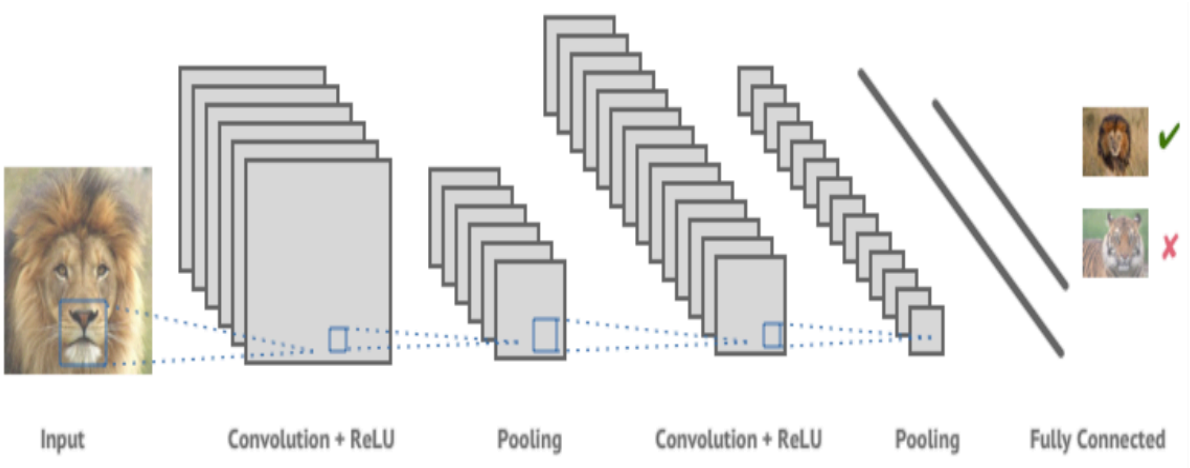


Figure 10 : Exemple d'un CNN [28]

L'architecture d'un CNN est formée par l'empilement de multiples couches permettant le traitement, en premier y'a la partie convolution qui permet d'extraire les caractéristiques des images en passant par un ensemble de filtres de convolution, des nouveaux images alors '' cartes de convolution '' sont créés sous forme d'un vecteur pour atteindre les couches entièrement connectées après une succession d'opérations de traitement (Pooling, correction et convolution), le rôle de ces couches nommés aussi FC (Fully Connected) est crucial puisqu'ils combinent tous les caractéristiques relevés par les cartes de convolution pour prendre une décision finale et classer l'image(cf. fig.10).

Pendant mes projets du Master 1 et Master 2 ainsi que mon stage du Master j'ai abordé l'aspect scientifique et technique des réseaux de neurones encore plus en détails, et ils pourront être visualisés sur le lien en référence [29].

IV- Données / Matériel / Outils de Validation

1-Présentation des Données

Images :

Les images des manguiers utilisés durant le stage sont des images couleur en provenance du Sénégal, elles ont été acquises en 2017 et 2018 ; il s'agit de 128 Images prises à 5 mètres de l'arbre.

Manguiers :

Les manguiers qui vont servir comme données pour le réseau de neurones contiennent trois variétés de mangues : BOUCODIEKHALE (BDH), KENT et KEITT.

Figure 11 : Mangue BDH

Figure 12 : Mangue KENT

Figure 13 : Mangue KEITT

Les trois variétés ont des caractéristiques différentes, et chacune entre elle a une forme et une couleur qui la distingue de l'autre ; les arbres qui les contiennent sont hétérogènes vu que leur taille ainsi que le positionnement des mangues et des feuilles changent d'une à l'autre.

Figure 14 : Manguier BDH

Figure 15 : Manguier KENT

Figure 16 : Manguier KEITT

Annotations expertes :

Les annotations expertes sont réalisées de façon manuelle pour chaque image, ils servent à identifier sur chaque image les mangues sous forme de boite englobante.

Les annotations se présentent sous forme de fichier Texte pour chaque image, et chaque mangue est représentée par des coordonnées relevées à travers sa dimension et son positionnement sur l'image (cf. fig. 17 et 18).

Figure 17 : Mangues annotées

	BX	BY	Width	Height
0	222	129	62	61
1	133	127	66	67
2	229	93	64	64
3	142	0	62	31

Figure 18 : annotations correspondantes aux mangues de la fig17

L'annotation experte est une phase importante pour garantir un bon fonctionnement du réseau, puisqu'une fausse annotation ou un oubli peuvent pénaliser le réseau durant l'entrainement et fausser les résultats de la détection.

2-Materiel

La machine utilisée durant le stage est une DELL Précision Tower 7910 munie d'un processeur Intel XEON E5-2620 v3 avec 12 cœurs, une mémoire de 64GB et un disque de capacité de 3 TO.

La carte graphique (GPU) est une NVIDIA Quadro M4000 dotée d'une mémoire GPU de 8GB, et sa puissance joue un rôle important puisqu'elle sollicitée durant plusieurs phases du traitement des réseaux de neurones spécialement l'entrainement.

3-Définition du Réseau YOLO/Tiny YOLO

YOLO (You only look once) est un réseau de neurones convolutif destiné à la détection et l'identification d'objets ; YOLO agit en single shot [16] et utilise un seul réseau de convolution pour simultanément détecter les objets et identifier leur classe respective, contrairement aux Region based-convolutional neural network (R CNN) qui déterminent des régions d'intérêt avant d'y détecter ou non la présence d'objets [9] [10] [11].

YOLO divise l'image en grilles et accorde pour chacune d'entre elle un nombre de boîtes englobantes ; les boites caractérisant les classes présentes dans les données d'entrainement, et qui dépassent un seuil prédéfini sont sélectionnées pour détecter les objets désirés (cf. fig. 19).

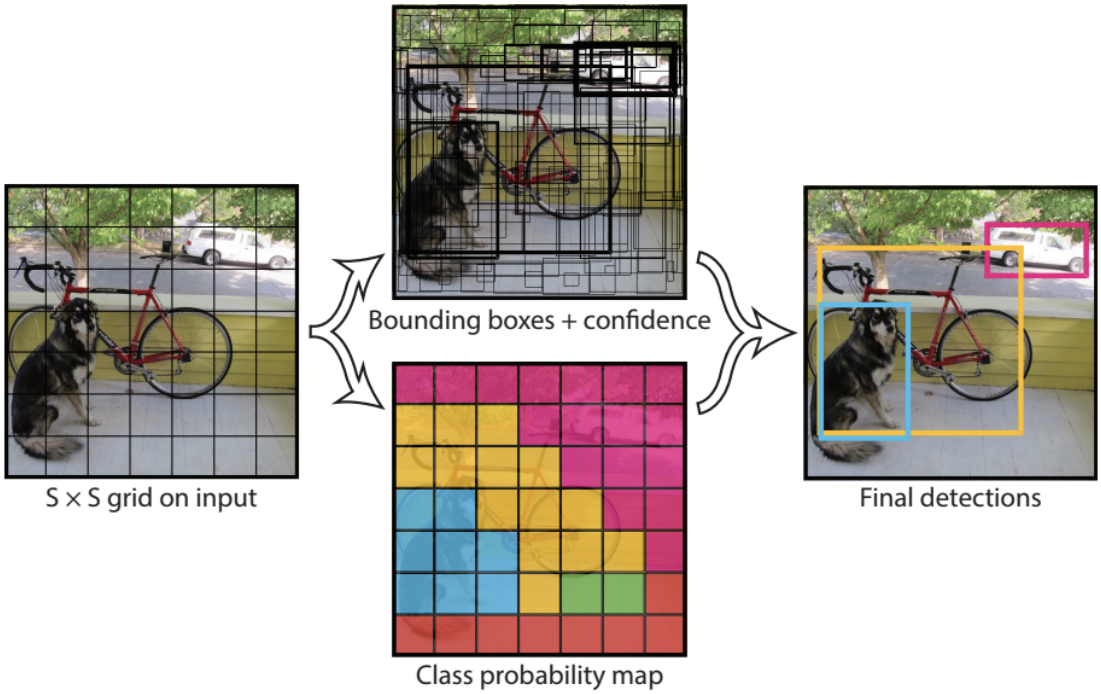


Figure 19 : Schéma de fonctionnement de YOLO [31]

YOLO v2 contient 24 couches de convolution suivies de deux couches entièrement connectées (cf. fig. 20) ; le **Tiny YOLO v2** qu'on utilise durant le stage pour la détection de mangues et qui est une variante de YOLO v2, a le même principe que la version full mais il est doté d'une architecture plus simplifiée (cf. fig. 21) et a été conçu pour tourner sur des systèmes embarqués.

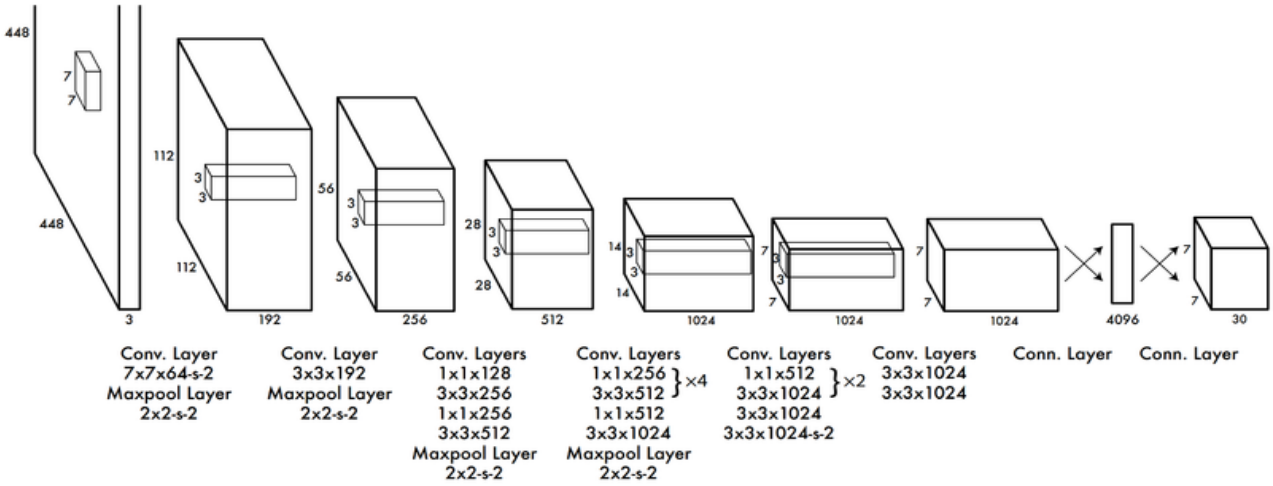


Figure 20 : Architecture de YOLO v2 [32]

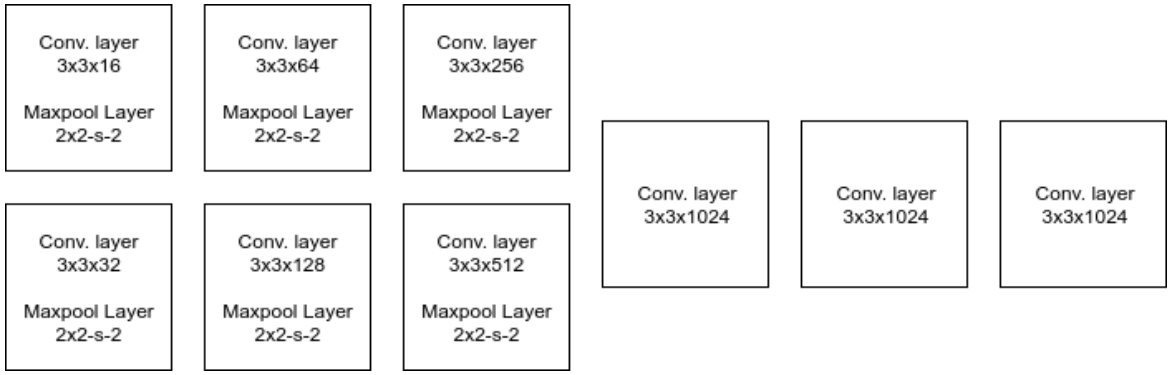


Figure 21 : Architecture du Tiny YOLO v2 [32]

4-Méthodes / Outils de validation

La validation consiste à mesurer la capacité du réseau à bien détecter les fruits de l'image ; pour cela on compare les annotations expertes et les prédictions du réseau.

L'indicateur qu'on a choisi d'utiliser comme méthode principal pour la validation est le F1 score, puisqu'il travaille sur les objets et nous permettra de connaitre la nature des erreurs ; il fait appel à des indicateurs comme la précision et le rappel qui s'appuient sur les effectifs des Vrais Positifs, faux Positifs et Faux Négatifs.

Sur l'ensemble des figures présentes sur cette section, les boites englobantes représentant les annotations sont en rouge et celles des prédictions du réseau sont en bleu.

Des scripts Python ont été développés pour exploiter tous les mesures et les indicateurs explicités ci-dessous afin d'évaluer les résultats de la détection fournis par le réseau de neurones.

Vrais positifs (VP) : c'est l'ensemble des prédictions du réseau de neurones est correcte, dans notre cas la mangue est bien détectée et correspond à l'annotation experte (cf. fig. 22 et 23).

Faux négatifs (FN) : c'est quand la prédiction du réseau de neurones est incorrecte, dans notre cas la mangue annotée par l'experte n'est pas détectée (cf. fig. 22 et 23).

Faux positifs (FP) : C'est l'ensemble des mauvaises prédictions du réseau, i.e. des fruits détectés par le réseau mais non annotés par l'expert. Mêmes les cas correspondant à un oubli de l'expert seront considérés comme une erreur du réseau (cf. fig. 22 et 23).

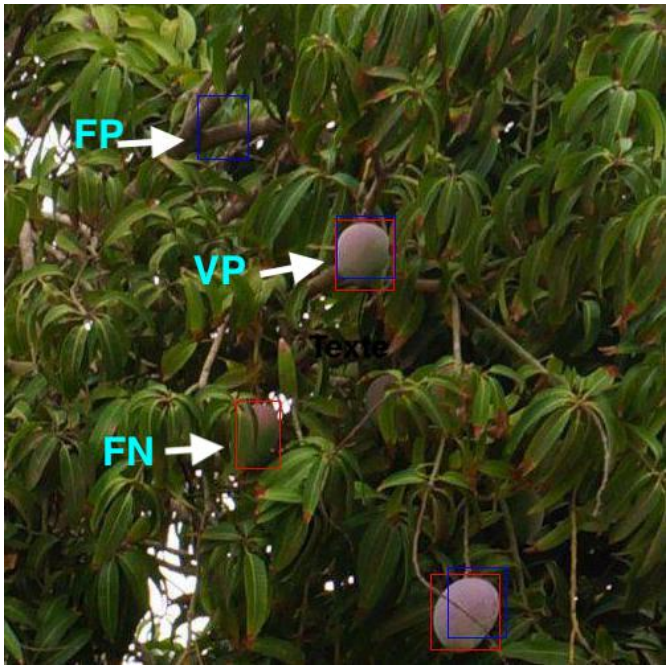


Figure 22 : exemple d'une photo du jeu de test avec des prédictions et annotations du réseau

Précision : elle permet de relever la proportion des prédictions correctes du réseau, par rapport à toutes les prédictions qu'il a fait '' justes et fausses'' (cf. fig. 23 et 24).

$$Precision = \frac{VP}{VP + FP}$$

Rappel (Recall) : il permet de relever la proportion des prédictions correctes du réseau mais seulement par rapport aux objets annotées (cf. fig. 23 et 24).

$$Recall = \frac{VP}{VP + FN}$$

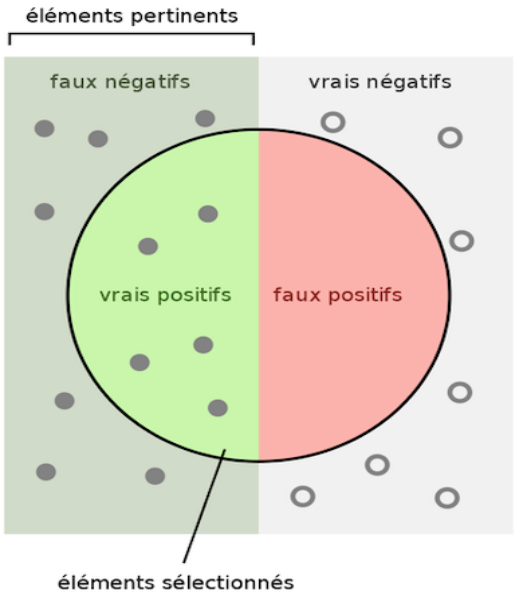


Figure 23 (source : Wikipédia)

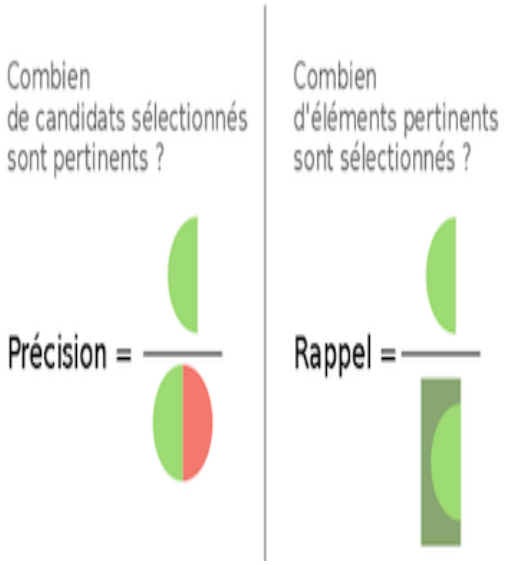


Figure 24 (Source : Wikipédia)

Justesse (Accuracy) : contrairement à la précision et au rappel, elle permet de désigner la proportion des prédictions justes mais par rapport au nombre total des prédictions et annotations.

$$Accuracy = \frac{VP}{VP + FP + FN}$$

IoU (Intesection over Union) : il permet de quantifier la qualité de la prédiction en évaluant chaque combinaison de boites englobantes de prédiction et celles de l'annotation, la prédiction est considérée comme bonne quand c'est supérieur à 0,5. (cf fig 25,26 et 27).

$$IoU = \frac{Zone\ d'intersection\ des\ 2\ boites}{Zone\ d'union\ des\ 2\ boites}$$

Figure 25 : Bonne prédiction (I (IoU > 0,5)

Figure 26 : Excellente prédiction (IoU proche de 1)

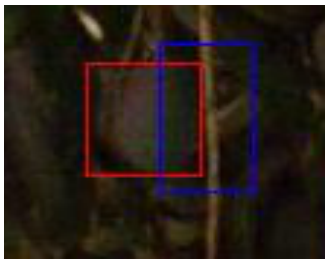


Figure 27 : Mauvaise prédiction (IoU <0,5)

F1 Score : C'est un indicateur global qui dépend de la précision et du rappel, donc il prend en considération à la fois les VP, FP et FN ; il est défini à partir de deux indicateurs (Rappel et Précision) ; l'un caractérisant les FP et l'autre les FN, ce qui facilite la recherche de la nature des erreurs commises par le réseau .

$$F1 = 2 * \frac{Precision * Recall}{Presicion + Recall}$$

Entrainement : Pour un réseau de Neurones de détection d'objets, l'entrainement est une étape primordiale pour arriver à la fin à bien détecter les objets souhaités, le but est d'utiliser des données étiquetées contenant les objets à détecter (jeu de données) pour apprendre au réseau à reconnaitre ces mêmes objets sur des données différentes de celles utilisées pour l'entrainement (jeu de test).

Perte (Loss) : C'est une fonction qui mesure la différence entre les prédictions et les annotations expertes durant l'apprentissage ; dans le cas des réseaux YOLO, elle est donnée par la somme de 3 types d'erreurs :

Erreur de localisation / Erreur de la confiance / Erreur de classification

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Figure 28 : Fonction Loss de YOLO[30]

L'erreur de localisation s'intéresse à la position et la taille (x,y,w et h) des boites englobantes de prédiction et d'annotation, afin de quantifier la différence entre les deux .

L'erreur de la confiance calcule la différence des scores de confiance entre la prédiction et l'annotation que ça soit dans le cas de la présence de l'objet ou celui de son absence.

L'erreur de classification représente la différence entre la probabilité de prédiction $p_i(c)$ et la réelle $\hat{p}i(c)$.

V- Déploiement Réseau Tiny YOLO sur Machine

1-Déploiement du Réseau

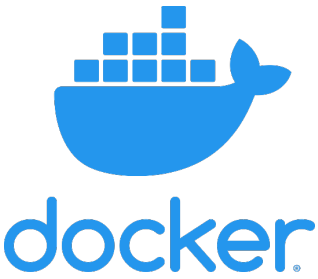
Darkflow

La version originale de YOLO est développée sous un open source Framework écrit en langage C et Cuda sous le nom de ‘’**Darknet’’** [20], toutefois une autre version nommée **Darkflow** basée sur la version ‘’Darknet’’ mais spécialement adaptée à **Tensorflow** et sa version **Lite** particulièrement dédiée au portage de réseaux sur smartphone semble mieux adapté à notre stage [26].

Tensorflow

TensorFlow est un outil d’apprentissage automatique open source développée par Google, son architecture permet de la flexibilité sur les calculs en GPU et CPU, il est écrit en C++ et python.

Conteneurisation



On utilise Docker pour la manipulation des réseaux durant notre stage ; Docker permet de lancer n’importe quelle application ainsi que toutes les dépendances dont elle a besoin pour tourner dans un conteneur isolé de la machine hôte, ce qui permet d’avoir une sorte de boite hermétique garantissant la sécurité de la machine hôte (cf. fig. 29),

Docker offre en même temps une possibilité de transportabilité entre différentes machines hôtes sans se soucier des dépendances qui peuvent changer d’une machine à l’autre.

Malgré l’utilité des conteneurs y’a des contraintes qui peuvent apparaitre souvent notamment celles en rapport avec la compatibilité du noyau linux de la machine hôte ainsi que quelques problèmes liés aux compatibilités entre différentes versions de Docker.

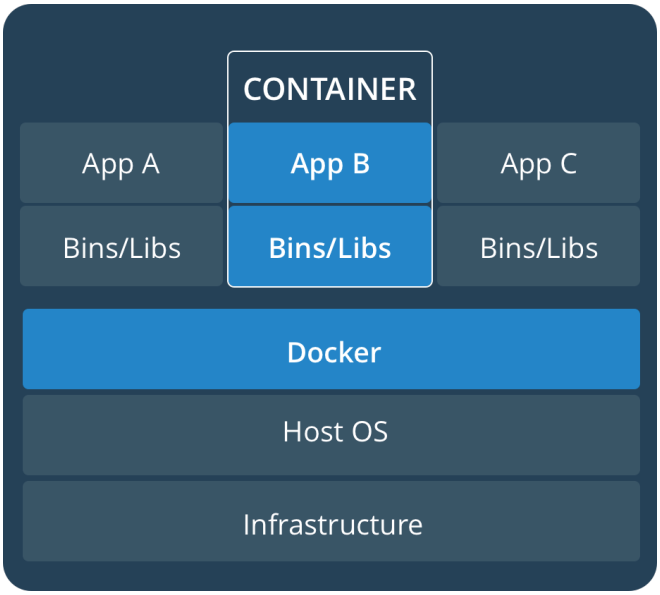


Figure 29 : Principe de fonctionnement d'un conteneur Docker [33]

2-Entrainement du Réseau.

Fine-Tunning

Le Fine-Tunning consiste à spécialiser avec une faible quantité de données des réseaux pré-entrainés sur des bases importantes de connaissances générales.

Pour la détection de mangues, on a opté pour un pré-entrainement réalisé sur la base de données Pascal VOC [34] : elle fait partie des bases de données les plus utilisées pour l'entrainement des réseaux spécialisés et est constituée de 9963 images de 500*300, contenant 24640 objets appartenant à 20 classes (humains, vélos, voitures ….).

Le recours au Pré-entrainement pour les réseaux spécialisés est dû au coût important du réentraînement intégral d'un réseau « from scratch » tant en termes de données, que de paramétrage ou temps machine.

Tuilage

Le pré-entrainement réalisé sur des images de 500*300 nous a poussé à découper nos images originales en tuiles de 500*500, les premiers essais réalisés sur les images de taille originale ainsi que des tuiles de taille 300*300 n'ayant pas donné de résultat de détection.

Entrainement

Pour l'entrainement sur Tiny Yolo v2 Darkflow , il a fallu convertir es annotations expertes initialement données en format ‘’txt ‘’ en format ‘’xml’’ de Pascal VOC (cf. fig. 30).

```
<filename>1_T1_L_MF_7.jpg</filename>
<size>
  <width>500</width>
  <height>500</height>
  <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
  <name>mango</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>165</xmin>
    <ymin>140</ymin>
    <xmax>197</xmax>
    <ymax>187</ymax>
  </bndbox>
</object>
<object>
  <name>mango</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>302</xmin>
    <ymin>14</ymin>
    <xmax>337</xmax>
    <ymax>54</ymax>
  </bndbox>
```

Figure 30 : exemple d'annotations xml darkflow

Jeu d'entrainement / Jeu de test

Jeu d'entrainement :

476 Tuiles 500x500 ➔ 1821 Annotations.

Jeu de test :

63 Tuiles 500x500 ➔ 442 Annotations.

L'entrainement a duré **6H30** pour un nombre **d'epoques** (nombres de cycles durant lesquels le réseau de neurones parcourt le jeu de données en entier ''epochs '') **de 1456** et avec **une loss moyenne de 0,9,** elle rentre dans la marge des loss des travaux sur la littérature sur le Tiny YOLO qui varient entre 0.1 et 2.

3-Résultats

Tiny YOLO v2	JEU D'ENTRAINEMENT	JEU DE TEST
F1 SCORE	**0,962**	**0,636**
ACCURACY	**0,928**	**0,466**
RECALL	**0,966**	**0,538**
PRECISION	**0,959**	**0,777**

Table 2 : Résultats des indicateurs de validations pour le Tiny Yolo v2

Les prédictions sur Tiny Yolo Darkflow sont données en format '' json'' (voir Annexe 1), et il a fallu les convertir en format ''txt '' et à la forme des annotations expertes niveau coordonnées afin de pouvoir effectuer la comparaison et avoir les différentes métriques ci-dessus grâce aux scripts python dédiés à la validation (voir Annexe 2)

Analyse :

Bien que le F1 score pour le jeu d'entrainement soit de 96%, on observe une chute significative de la valeur de l'indicateur sur les jeux de test : le réseau arrive à détecter correctement sur plusieurs images du jeu de test (cf. fig. 30), mais le 53% du rappel mettent en cause les Faux négatifs et on le voit bien sur quelques images où le réseau trouve des difficultés (cf. fig. 31).

Figure 30 : Exemple d'une image du jeu de test

Figure 31 : Exemple d'une image du jeu de test

On voit bien que le réseau a du mal à détecter les fruits coupés ou cachés ; il est perturbé aussi dans le cas où il y a un chevauchement de mangues, et considère les deux comme une seule mangue (cf. fig. 31) ; on voit bien sur la figure 32 (Annotation en rouge

Prédictions en bleu) produite par les scripts python développés pour la validation des résultats, qu'il y'a des oublis (FP) et des erreurs de détection (FN).

Figure 32 : Exemple d'une image du jeu de test avec prédictions et annotations

Afin de pointer la nature des difficultés du réseau et détecter si c'est un problème de jeu de données ou du mécanisme du réseau, on suit le même processus effectué sous Tiny YOLO v2 avec d'autres réseaux.

Comparaison avec Yolo v2 / Faster R-CNN / YOLO v4

On a effectué l'entrainement du Tiny Yolo v2 (2018), d'un YOLO v2 (2016), d'un Faster R-CNN (2015) et d'un YOLO v4 (Mai 2020) avec le même jeu de données annotées. Les différents réseaux ont été évalués d'une part à partir du jeu d'entrainement pour se mettre en situation a priori la plus favorable (cf tab 2) et d'autre part à partir d'un même jeu de test indépendant du jeu de données pour illustrer la tendance réelle (cf tab 3).

JEU D'ENTRAINEMENT	YOLO v2	Tiny YOLO v2	Faster R-CNN	YOLO v4
F1 SCORE	**0,935**	**0,962**	**0,911**	**0,983**
ACCURACY	**0,878**	**0,928**	**0,837**	**0,967**
RECALL	**0,88**	**0,966**	**0,948**	**0,993**
PRECISION	**0,998**	**0,928**	**0,877**	**0,973**

Table 3 : Comparatif des résultats des indicateurs de validation pour le jeu d'entrainement entre les différents réseaux

JEU DE TEST	YOLO v2	Tiny YOLO v2	Faster R-CNN	YOLO v4
F-MEASURE	**0,669**	**0,636**	**0,852**	**0,819**
ACCURACY	**0,503**	**0,466**	**0,743**	**0,693**
RECALL	**0,536**	**0,538**	**0,840**	**0,737**
PRECISION	**0,891**	**0,777**	**0,865**	**0,92**

Table 4 : Comparatif des résultats des indicateurs de validation pour le jeu de test entre les différents réseaux

Analyse :

YOLO v2 suit globalement le même comportement que le Tiny YOLOv2 (cf. fig. 33), des valeurs fortes pour le jeu d'entrainement et un effondrement très significatif lors du test du jeu de validation ; dans les deux cas, on constate un effondrementt du rappel, pointant donc une augmentation du nombre de faux positifs.

Figure 33 : Même exemple de la fig 31 avec YOLO v2

Le Faster R-CNN présente un F1 score largement meilleur que ceux des réseaux YOLO v2 sur le jeu de test bien qu'il soit inférieur quand il s'agit du jeu de données ; avec une meilleure aptitude à détecter les mangues partiellement cachées et coupées par des branches, il est moins pénalisé que les réseaux YOLO (cf. fig. 34).

Figure 34 : Même exemple de la fig 31 avec Faster R-CNN

YOLO v4 lui aussi présente un F1 Score supérieur à celui de ses prédécesseurs, bien qu'il soit un peu inférieur niveau détection aux performances du Faster R-CNN.

Il détecte correctement et sans beaucoup de faux négatifs, il gère des problèmes comme celui de la confusion durant le chevauchement des fruits ou dans les cas des fruits cachés que YOLO rencontre sur la version 2 (cf. fig. 35).

Figure 35 : Même exemple de la fig 31 avec YOLO v4

VI- Embarquement du Réseau Tiny YOLO sur Smartphone

1-Définition de l'environnement utilisé

TensorFlowLite : une version de Tensorflow destinée aux systèmes embarqués et smartphones, qui permet de convertir le fichier de poids final obtenu après l'entrainement (paramétrage du réseau pour la détection) en un fichier lite qui sera embarqué après sur smartphone via une application.

Android Studio : environnement de développement destiné pour Android, il propose des outils pour le développement d'applications mobiles ainsi que des émulateurs virtuels de smartphones Android sur un ordinateur.

Flutter : un Framework de développement mobile open source conçu par Google, il utilise comme langage de programmation le langage Dart crée par Google et permet de développer des applications natives pour Android et IOS.

2- Matériel / Méthodes

Matériel

Les tests réalisés ont été faits sur un Smartphone Huawei Mate 20 Lite et les émulateurs Nexus One et Nexus 6 :

Huawei Mate 20 Lite :

Mémoire : 64 GO
RAM : 4GO

GPU : ARM Mali G51 MP4
Chipset : HiSilicon Kirin 71
CPU : Octa core (2.2 GHz, Quad core, Cortex A73 + 1.7 GHz, Quadcore, Cortex A53).

Nexus One :

Mémoire : 512 MB
RAM : 512 MB
GPU : Adreno 200
Chipset : Qualcomm
CPU : Scorpion (1GHZ)

Nexus 6 :

Mémoire : 32 GB
RAM : 3 GB
GPU : Adreno 420
Chipset : Qualcomm Snapdragon 805 APQ8084
CPU : Quad core, 2.7 GHz, Krait

Méthodes

On commence par convertir notre fichier de poids final après la fin de l'entrainement du Tiny Yolo v2 en fichier tflite via Tensorflow, et on utilise le package tflite de Flutter pour accéder à une API de Tensorflow lite [35] et on adapte après les différents fichiers et détails du code de l'application à notre fichier tflite spécialisé (cf. fig. 36,37 et 38 prises du Huawei)

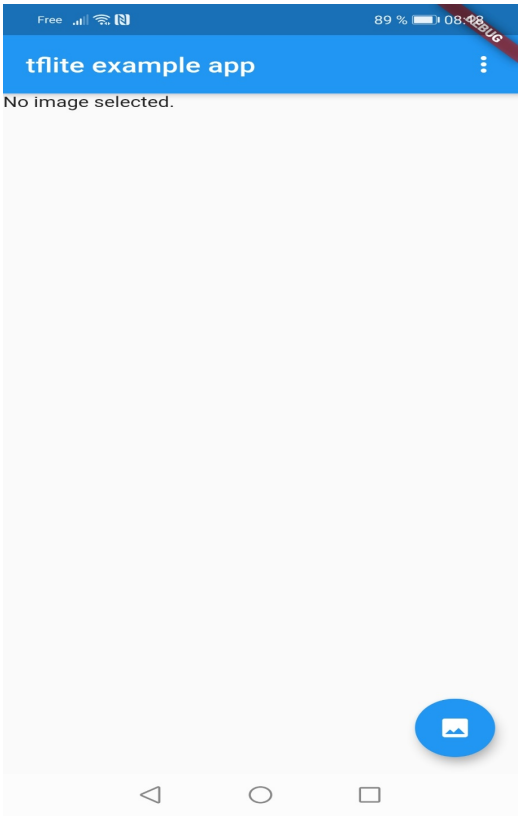


Figure 36 : première page de l'application

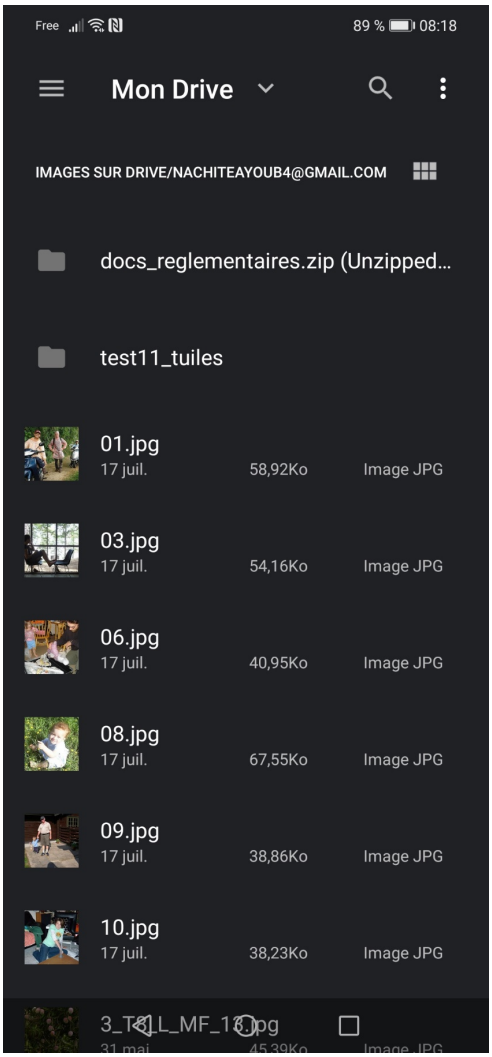


Figure 37 : choix de l'image de l'application

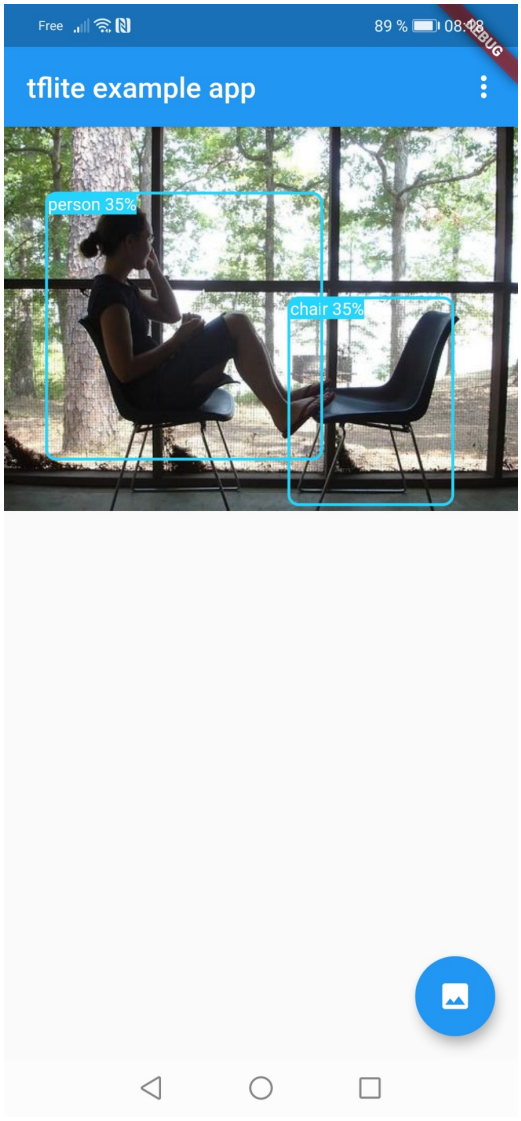


Figure 38 : phase de la détection sur l'application

Le choix de l'image à détecter sur l'application se fait soit à partir de la galerie du smartphone, soit à partir du Google drive de l'utilisateur.

3- Résultats

Les résultats sur Smartphone Huawei Mate 20 Lite du réseau Tiny Yolo embarqué sur le jeu de test sont moins précis, on a largement moins de détections sur les mêmes images traités sur machine, les trois exemples ci-dessous le démontrent bien (cf. fig. 39,40,41 et 42).

Si les détections obtenues sur les des deux émulateurs et sur le matériel physique sont exactement les mêmes, les temps d'inférence (temps d'exécution avant la détection) sont significativement différents.

Temps d'inférence	Huawei Mate 20 Lite	Nexus One (Virtuel)	Nexus 6 (Virtuel)
Exemple 1	798 ms	858 ms	706 ms
Exemple 2	708 ms	824 ms	710 ms

Table 5 : Tableau comparatif du temps d'inférences sur les 3 smartphones

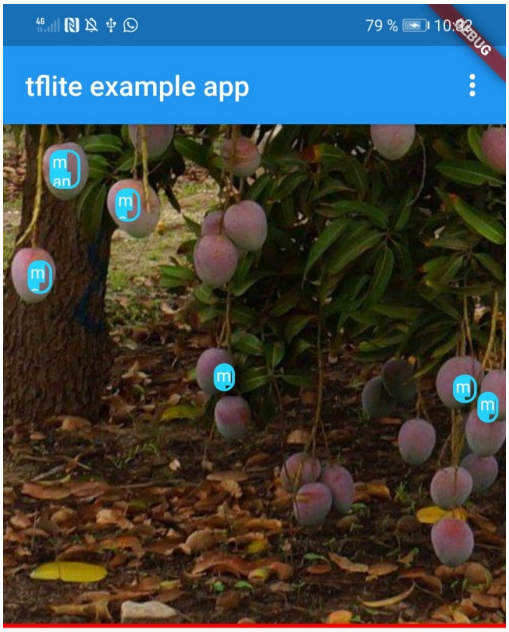


Figure 39 : Exemple 1 jeu de test sur Huawei

Figure 40 : Exemple 1 jeu de test sur Machine

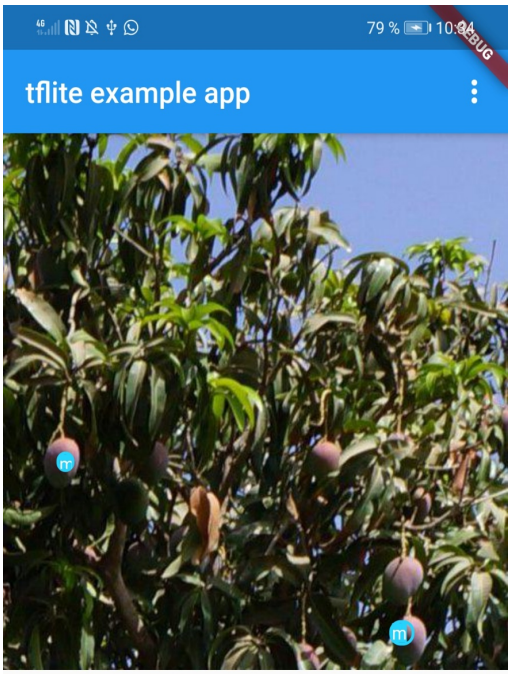


Figure 41 : Exemple 2 jeu de test sur Huawei

Figure 42 : Exemple 2 jeu de test sur Machine

Ressources énergétiques

Afin de quantifier l'effet de l'embarquement sur le smartphone, la figure ci-dessous (cf. fig. 43) suit l'évolution des différentes ressources utilisées par le smartphone Huawei mate 20 Lite depuis le lancement de l'application, puis le choix de l'image et le moment de l'exécution de l'image avant la détection.

Les pics au début niveau energie reviennent au moment où l'utilisateur parcourt les images, puis une chute est remarquée quand le choix est fait, pour remonter après sur tous les niveaux (Mémoire, CPU et Energie) quand l'operation de l'execution de l'image est en cours avant la detection finale.

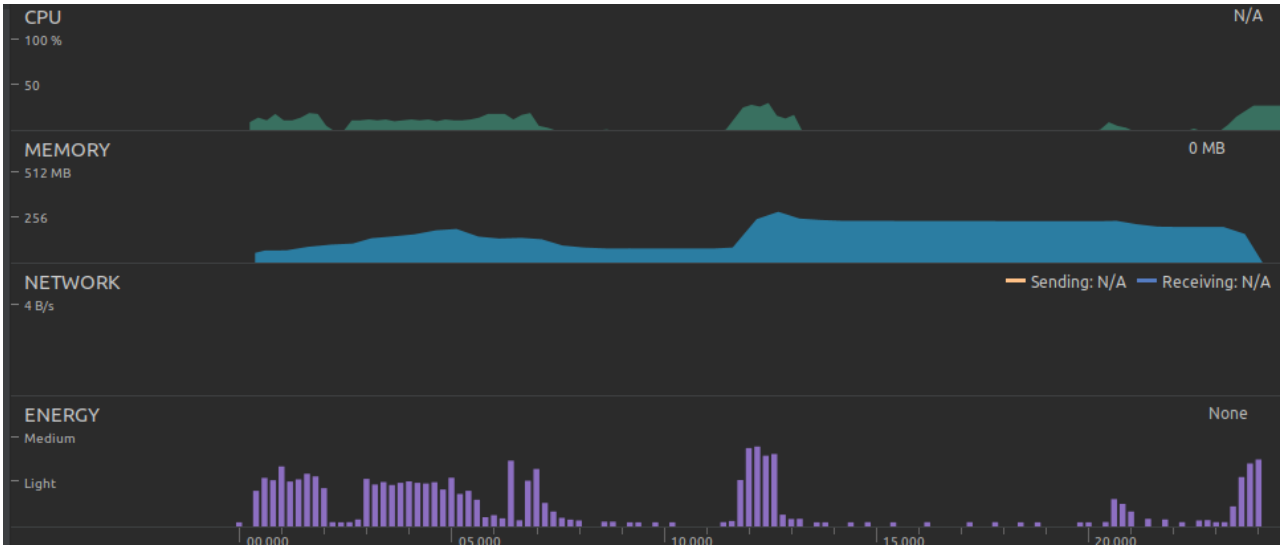


Figure 43 : Ressources énergétiques consommées par Huawei Mate 20 Lite

Analyse

On constate que le réseau Tiny Yolo v2 embarqué a du mal niveau détection puisqu'il perd beaucoup en performances par rapport à ses résultats sur machine, est ce que la conversion du fichiers poids en tflite pénalise le réseau ?

Les essais menés sur le réseau original (VOC) à travers lequel notre réseau a été pré-entrainé pointent le même problème, puisqu'il y'a une différence entre les résultats obtenus sur machine et sur Smartphone Huawei Mate 20 Lite (cf.fig.44 et 45).

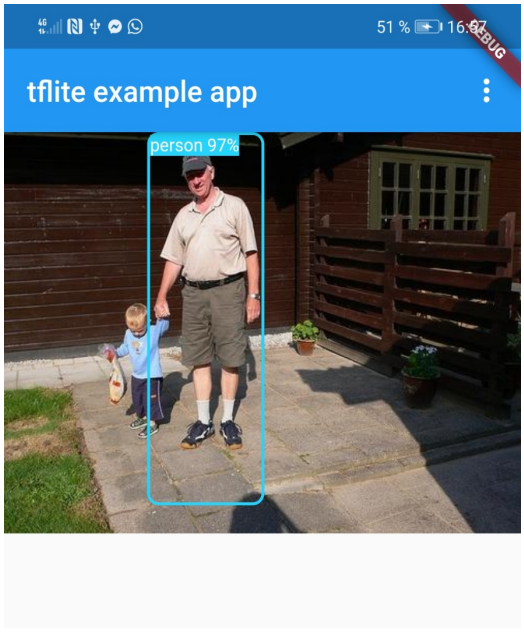


Figure 44 : Exemple VOC sur Smartphone

Figure 45 : Exemple VOC sur Machine

Comme c'est le cas sur notre réseau spécialisé de mangues, le réseau original entrainé sur la base de données VOC loupe aussi des objets qui ont été bel et bien détectés par le même réseau en version machine.

VII- Discussion / Conclusion

Au vu des différents résultats, la conclusion est sans appel : le Réseau Tiny Yolo v2 a beaucoup de difficultés dans les tâches de détection de fruits, tant par rapport au Yolo v4 qu'au Faster R-CNN, l'objectif de l'embarquement nous condamnait en quelque sorte à l'utiliser (Tableau ci-dessous).

	Version FULL	Version TINY
Faster R-CNN	**Juin 2015**	**……**
YOLO v2	**Fin 2016**	**Mars 2018 ➔ Embarquable**
YOLO v3	**Avril 2018**	**Aout 2018 ➔ Non Embarquable**
YOLO v4	**Mai 2020**	**Fin Juin 2020 (Version non officielle , un projet soutenu par darknet) ➔ Rien encore sur l'embarquement**

Table 6: Années d'apparition des différents principaux CNN de détection et possibilité d'embarquement

Le domaine de la détection d'objets par CNN évolue rapidement, et des réseaux plus performants apparaissent chaque année, on l'a vu avec les résultats du YOLO v4 et qui d'ailleurs n'a vu le jour que deux mois après le début de mon stage, le Faster R-CNN a montré aussi de bonnes performances mais son architecture fait qu'il ne peut pas être embarqué au contraire du YOLO v2 à travers sa variante le Tiny YOLO, malgré le fait qu'ils soient sensibles à beaucoup de contraintes comme le rapport ratio/image (cf.fig. 46 et 47) ou la difficulté avec les petits objets, et ce qui rend leur détection moins performante.

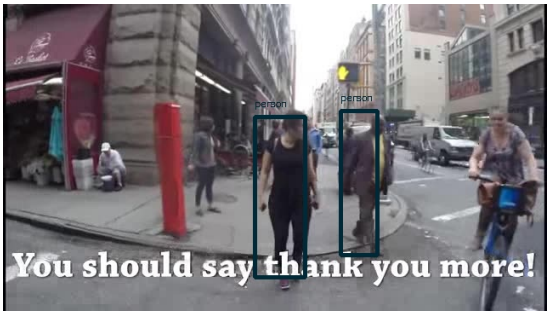


Figure 46 : Image test VOC 640*360

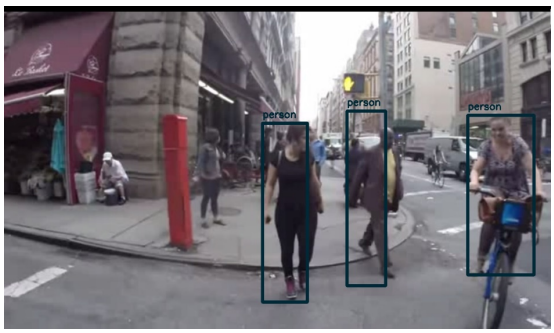


Figure 47 : Image test VOC 1173*690

YOLO v4 semble un réseau prometteur si une version officielle Tiny destiné à l'embarquement voit le jour d'ici quelques mois ; d'autres réseaux comme Ssd Mobilenet

peuvent être intéressants à explorer en vue d'un entrainement et test sur machine et smartphone pour les mangues.

Certes, disposer en fin de stage d'un réseau embarqué et précis n'a pas été possible. On a toutefois pu explorer les différentes failles et contraintes qui joueront surement un rôle important dans un futur portage pour permettre le comptage correct de fruits, notamment les opérations de pré et post traitement sur les tuiles pour se ramener à la taille de la photo prise par le smartphone, ainsi que la gestion des prédictions afin d'avoir une sortie permettant de compter les mangues.

Sur le plan personnel, ce stage m'a permis de voir le domaine des réseaux de neurones d'un côté purement applicatif après avoir fait différents projets et stage portant sur les réseaux de neurones avec une vision plus théorique et hardware, ce qui vient donc compléter mes acquis des années passées pour voir leur impact de près sur des domaines qui touchent nos vies de tous les jours.

VIII- Références / Annexes

Références

[1] : Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017). **Big data in smart farming–a review. Agricultural Systems, 153, 69-80.**
[2] : Jayaraman, P. P., Yavari, A., Georgakopoulos, D., Morshed, A., & Zaslavsky, A. (2016). **Internet of things platform for smart farming: Experiences and lessons learnt. Sensors, 16(11), 1884.**
[3]: Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). **Deep learning in agriculture: A survey. Computers and electronics in agriculture, 147, 70-90..**
[4]: Anderson, N. T., Underwood, J. P., Rahman, M. M., Robson, A., & Walsh, K. B. (2019). **Estimation of fruit load in mango orchards: tree sampling considerations and use of machine vision and satellite imagery. Precision Agriculture, 20(4), 823-839.**
[5]: Yann Le Cun (2016).**L'apprentissage Profond: Une Révolution en Intelligence Artificielle Leçon Inaugurale au Collège de France Chaire Annuelle 2015-2016 Informatique et Sciences Numériques.**
[6]: Claude TOUZET (1992). **Les Réseaux de Neurones Artificiels :Introduction au connexionnisme cours,exercices et travaux pratiques**
[7]: Chua, L. O. (1998). **CNN: A paradigm for complexity (Vol. 31). World Scientific.**
[8]: https://www.sciencedirect.com/topics/computer-science/deep-neural-network.
[9]: Girshick, R., Donahue, J., Darrell, T.,Malik, J.(2014). **Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 580–587.**
[10]: Ren, S., He, K., Girshick, R., Sun, J.(2015). **Faster r-cnn: Towards real-time object detection with region proposal networks.Advances in neural information processing systems, 91–99.**
[11.a]: Chen, X., Gupta, A.(2017). **An implementation of faster RCNN with study for region sampling.arXiv:1702.02138.**
[11.b]: Kaiming He., Georgia Gkioxari., Piotr Dollár., Ross Girshick.(2018). **Mask R-CNN sampling.arXiv:1703.06870v3.**
[12]: Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., and McCool, C.(2016). **Deepfruits: A fruit detection system using deep neural networks. Sensors, 16(8), 1222.**
[13]: Borianne, P., Sarron, J., Borne, F., Faye, E.(2019). **Deep Mangoes: from fruit detection to cultivar identification in colour images of mango trees. DISP '19, Oxford, United Kingdom. ISBN: 978-1-912532-09-4.**
[14]: Koirala, T., Walsh , K.B., Wang, Z., McCarthy, C.(2019). **Deep learning – Method overview and review of use for fruit detection and yield estimation ,162,219–234.**
[15] : Joseph Redmon ; Santosh Divvala ; Ross Girshick ; Ali Farhadi .**You Only Look Once: Unified, Real-Time Object Detection**
[16] :Bresilla, K. Perulli, G.D., Boini, A., Morandi, B., Corelli, L., Manfrini, G.L.(2019). **Single- Shot Convolution Neural Networks for Real-Time Fruit Detection Within the Tree**
[17]: Koirala A, Walsh KB, Wang Z, McCarthy C (2019). **Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of MangoYOLO.**
[18]: Yunong Tian, Guodong Yang, Zhe Wang,Hao Wang, En Li, Zize Liang (2019). **Apple detection during different growth stages in orchards using the improved YOLOV3 model.**
[19]: https://github.com/tensorflow/models/issues/4848.
[20]: https://pjreddie.com/publications/.

[21] : http://ramok.tech/tag/tiny-yolo/
[22] : Alsing O (2018). **Mobile Object Detection using TensorFlow Lite and Transfer Learning. Master in Computer Science, KTH Royal Institute of Technology**
[23] : Muthu N (2019). **Real-Time Object Detection with Flutter, TensorFlow Lite and Yolo**
[24] : https://github.com/thtrieu/darkflow
[25]: http://ai-benchmark.com/
[26] : https://www.tensorflow.org/lite
[27] : https://medium.com/@ODSC/using-the-cnn-architecture-in-image-processing-65b9eb032bdc
[28] : https://blog.goodaudience.com/convolutional-neural-netin-tensorflow-e15e43129d7d
[29] : https://filesender.renater.fr/?s=download&token=a6df0035-d9cb-47d7-ae0d-384896202a93
[30] : https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
[31] : https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
[32] :https://developer.ridgerun.com/wiki/index.php?title=GstInference/Supported_architectures/TinyYoloV2
[33] : https://docs.docker.com/get-started/
[34] : http://host.robots.ox.ac.uk/pascal/VOC/
[35] : https://pub.dev/packages/tflite

Annexes

Annexe 1 : Exemple d'un fichier prediction json fournie par Darkflow

[{"label": "mango", "topleft": {"x": 64, "y": 0}, "bottomright": {"x": 122, "y": 16}, "confidence": 0.5}, {"label": "mango", "topleft": {"x": 103, "y": 0}, "bottomright": {"x": 156, "y": 29}, "confidence": 0.99}, {"label": "mango", "topleft": {"x": 362, "y": 0}, "bottomright": {"x": 423, "y": 23}, "confidence": 0.86}, {"label": "mango", "topleft": {"x": 462, "y": 7}, "bottomright": {"x": 499, "y": 57}, "confidence": 1.0}, {"label": "mango", "topleft": {"x": 34, "y": 14}, "bottomright": {"x": 85, "y": 73}, "confidence": 1.0}, {"label": "mango", "topleft": {"x": 188, "y": 27}, "bottomright": {"x": 235, "y": 85}, "confidence": 0.83}, {"label": "mango", "topleft": {"x": 102, "y": 60}, "bottomright": {"x": 154, "y": 129}, "confidence": 0.9}, {"label": "mango", "topleft": {"x": 211, "y": 77}, "bottomright": {"x": 264, "y": 135}, "confidence": 0.61}, {"label": "mango", "topleft": {"x": 14, "y": 120}, "bottomright": {"x": 48, "y": 179}, "confidence": 0.76}, {"label": "mango", "topleft": {"x": 185, "y": 96}, "bottomright": {"x": 247, "y": 162}, "confidence": 1.0}, {"label": "mango", "topleft": {"x": 188, "y": 223}, "bottomright": {"x": 233, "y": 276}, "confidence": 0.94}, {"label": "mango", "topleft": {"x": 437, "y": 227}, "bottomright": {"x": 480, "y": 287}, "confidence": 0.92}, {"label": "mango", "topleft": {"x": 198, "y": 267}, "bottomright": {"x": 248, "y": 313}, "confidence": 0.92}, {"label": "mango", "topleft": {"x": 456, "y": 263}, "bottomright": {"x": 497, "y": 319}, "confidence": 0.96}, {"label": "mango", "topleft": {"x": 397, "y": 293}, "bottomright": {"x": 438, "y": 331}, "confidence": 0.79}, {"label": "mango", "topleft": {"x": 451, "y": 295}, "bottomright": {"x": 484, "y": 345}, "confidence": 0.97}, {"label": "mango", "topleft": {"x": 292, "y": 345}, "bottomright": {"x": 338, "y": 401}, "confidence": 1.0}, {"label": "mango", "topleft": {"x": 422, "y": 388}, "bottomright": {"x": 470, "y": 446}, "confidence": 1.0}]

Annexe 2 : fichier csv sortant après les différents scripts python de validation (exemple :jeu de test tiny yolo v2)

	NOM-FICHIER	VP-(mango)	FP-(mango)	FN-(mango)
2	11_T8_S_MF_25.txt	7	2	4
3	11_T6_L_MF_10.txt	2	1	3
4	11_T6_L_MF_11.txt	1	2	1
5	11_T6_L_MF_2.txt	5	0	4
6	11_T8_S_MF_21.txt	4	1	4
7	11_T6_L_MF_16.txt	2	0	2
8	11_T8_S_MF_3.txt	0	1	4
9	11_T8_S_MF_18.txt	5	0	6
10	11_T6_L_MF_13.txt	2	1	5
11	3_T8_L_MF_16.txt	6	0	5
12	11_T6_L_MF_9.txt	2	1	2
13	11_T8_S_MF_22.txt	1	0	5
14	11_T8_S_MF_28.txt	0	1	1
15	3_T8_L_MF_6.txt	3	3	5
16	11_T8_S_MF_26.txt	2	2	2
17	11_T8_S_MF_7.txt	2	1	3
18	3_T8_L_MF_18.txt	5	0	6
19	3_T8_L_MF_23.txt	9	1	12
20	3_T8_L_MF_13.txt	16	0	4
21	11_T6_L_MF_17.txt	3	1	3
22	11_T6_L_MF_12.txt	2	1	3
23	11_T8_S_MF_24.txt	2	0	6
24	3_T8_L_MF_21.txt	1	2	1
25	3_T8_L_MF_12.txt	4	0	5
26	3_T8_L_MF_8.txt	2	1	1
27	11_T6_L_MF_3.txt	1	0	3
28	3_T8_L_MF_1.txt	5	1	3
29	11_T6_L_MF_21.txt	2	2	5
30	3_T8_L_MF_22.txt	15	3	9
31	3_T8_L_MF_19.txt	3	0	1
32	11_T6_L_MF_6.txt	7	0	1
33	11_T8_S_MF_13.txt	1	0	0
34	11_T8_S_MF_16.txt	2	1	1
35	11_T8_S_MF_14.txt	4	0	2
36	3_T8_L_MF_9.txt	3	1	3
37	11_T8_S_MF_17.txt	0	2	2
38	3_T8_L_MF_15.txt	8	1	8
39	3_T8_L_MF_17.txt	6	0	3
40	11_T8_S_MF_15.txt	6	4	2
41	11_T8_S_MF_19.txt	6	5	3
42	11_T6_L_MF_15.txt	5	2	4
43	11_T6_L_MF_7.txt	3	1	3
44	11_T8_S_MF_27.txt	5	3	5

45	3_T8_L_MF_7.txt	2	0	4			
46	11_T8_S_MF_10.txt	2	3	1			
47	3_T8_L_MF_14.txt	8	1	4			
48	11_T8_S_MF_2.txt	1	4	3			
49	3_T8_L_MF_2.txt	2	1	1			
50	11_T8_S_MF_12.txt	3	0	4			
51	11_T6_L_MF_8.txt	2	0	2			
52	11_T8_S_MF_1.txt	1	0	0			
53	11_T8_S_MF_32.txt	1	0	0			
54	3_T8_L_MF_24.txt	10	2	4			
55	11_T6_L_MF_18.txt	4	2	1			
56	3_T8_L_MF_5.txt	2	0	3			
57	11_T8_S_MF_8.txt	4	1	2			
58	11_T6_L_MF_22.txt	4	0	2			
59	11_T8_S_MF_20.txt	3	0	3			
60	3_T8_L_MF_3.txt	1	0	1			
61	11_T6_L_MF_1.txt	2	0	5			
62	11_T8_S_MF_9.txt	6	5	3			
63	11_T6_L_MF_14.txt	5	0	0			
64	3_T8_L_MF_11.txt	5	1	6			
65							
66		mango	background				
67	mango	238	68				
68	background	204	0				
69							
70	CLASS	ACCURACY	ERROR	PRECISION	RECALL	F-MEASURE	
71	mango	0.4666666666666667	0.5333333333333333	0.7777777777777778	0.5384615384615384	0.6363636363636364	
72	mean	0.4666666666666667	0.5333333333333333	0.7777777777777778	0.5384615384615384	0.6363636363636364	