République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de Sciences

Département Mathématique et Informatique (MI)

Mémoire de Master

Filière : Informatique

Spécialité: Ingénierie des systèmes et technologie web

Thème

Déploiement sur mobile d'un modèle à base du Transfer Learning pour la détection de maladies liées aux plantes

Proposé par :

- Mr. TOULAIT Sofiane.

Présenté par :

- Mr.Belagoun Ahmed BenYoucef.
- Mr. Khelil cherfi Abdesslam.

Année : 2022 – 2023

Remerciements :

Tout d'abord, louanges et remerciements à Dieu, le Tout-Puissant pour ses averses de bénédiction tout au long de notre travail de recherche pour compléter la recherche avec succès

Je tiens à exprimer ma sincère gratitude à « Toulait Sofiane », mon superviseur, pour ces conseils, cette patience et cet encouragement tout au long de ce projet. Son expertise et sa perspicacité ont été inestimables pour m'aider à relever les défis de la mise en œuvre et du déploiement d'un modèle d'apprentissage par transfert pour la détection des maladies des plantes.

J'aimerais également remercier les membres du Département De Mathématiques Et d'Informatique pour leur soutien et leur collaboration. Leurs contributions à ce projet ont été importantes, et je leur suis reconnaissant de leurs commentaires.

Enfin, je voudrais saluer le laboratoire de recherche scientifique de l'Université de Médée, et remercier tout particulièrement le professeur « Oumona Mustafa » pour son soutien, qui a rendu cette recherche possible.

Dédicace :

Dédicace 01 :

Dévotions spéciales à mon père « **Mahfoud** » et ma mère « **yamna** », ma sœur « **Amel** », mes frères « **Nadjib** »et « **Seddik** », ma belle-sœur « **Hiba** » qui ont toujours été ma source d'inspiration et de motivation. Leur amour et leur soutien inébranlable m'ont guidé à chaque étape de mon parcours académique. Notre petit ange « **Amena** ».

Cette thèse est dédiée à :

- ✓ À mon collègue **Khelil cherfi Abdesslam** Merci pour les efforts déployés.
- ✓ À mes grands-parents, mes oncles, mes tantes et mes cousins
- ✓ Les familles « **Belagoun** » et « **bourial** »et « **ben chikh** ».
- ✓ Mes amis

(**Noureddine**, **Abdelkader**, **Yazid**, **Yahia**, **Tayeb**)

- ✓ Tous ceux qui me connaissent.
- ✓ M'ont aidé dans cette thèse.

Ahmed ben youcef

Dédicace 02 :

Je dédie ce mémoire à mes chers parents ma mère et mon père lah yerhmo, Pour leur patience, leur soutien et leurs encouragements, à mes frères et à mes amis et mes camarades. Sans oublier tous les professeurs.

Abdessalam Khelil Chorfi

Résumé :

Ce projet se concentre sur l'application des techniques d'apprentissage par transfert dans le domaine de la détection et de la classification des maladies des plantes. L'objectif est d'implémenter plusieurs modèles pré-entrainés et évaluer leurs performances, et ensuite, procéder au déploiement de l'un des modèles sur téléphone mobile dans le cadre d'une application Android, qui permettra aux utilisateurs d'identifier et de diagnostiquer les maladies des plants de pomme et de raisin. Les chapitres de ce mémoire couvrent plusieurs aspects, allant de l'apprentissage profond, des méthodes de transfert d'apprentissage, des stratégies de déploiement et de l'évaluation des modèles ainsi que le développent de l'application mobile. Quatre modèles pré-entraînés populaires, à savoir DenseNet-201, VGG16, ResNet-50 et Inception-V3, sont utilisés pour l'extraction des caractéristiques et la classification. Les performances de ces modèles sont évaluées sur un PantVillage qui est un dataset contenant plus de 54000 images de plantes, en mesurant l'exactitude dans la phase d'entraînement, la perte dans la phase d'entraînement, l'exactitude dans la phase de test et la perte dans la phase test. Les résultats montrent que Densenet201obtient la plus haute exactitude à la fois sur le dataset d'entraînement et de tests, suivi de InceptionV3, VGG16 et ResNet50. Cependant, en raison des contraintes liées à l'environnement Android, VGG16 a été choisi comme modèle final pour le déploiement dans l'application mobile. L'application développée offre une interface utilisateur intuitive et démontre l'efficacité de l'apprentissage par transfert dans l'identification et le diagnostic des maladies des plantes.

Mot clés :

Apprentissage profond, apprentissage par transfert, VGG16, ResNet-50, Inception-V3, DenseNet-201, application Android, PlantVillage.

الملخص :

يركز هذا المشروع على تطبيق تقنيات التعلم التحويلي في مجال الكشف عن أمراض النبات وتصنيفها. الهدف هو تنفيذ العديد من النماذج المدربة مسبقًا وتقييم أدائها، ثم المضي قدمًا في نشر أحد النماذج على هاتف محمول كجزء من تطبيق Android، والذي سيسمح للمستخدمين بتحديد وتشخيص أمراض نباتات التفاح والعنب. تغطي فصول هذه الرسالة العديد من الجوانب، بدءًا من التعلم العميق وطرق نقل التعلم واستراتيجيات النشر وتقييم النموذج بالإضافة إلى تطوير تطبيقات الهاتف المحمول. يتم استخدام أربعة نماذج شائعة مدربة مسبقًا، وهي Densenet201 وVGG16 وResNet50 وInceptionV3، لاستخراج الميزات وتصنيفها. يتم تقييم أداء هذه النماذج على PantVillage وهي مجموعة بيانات تحتوي على أكثر من 54000 صورة نباتية، من خلال قياس الدقة في مرحلة التدريب، والخسارة في مرحلة التدريب، والدقة في الاختبار، والخسارة في مرحلة الاختبار. تظهر النتائج أن Densenet201 يحقق أعلى دقة في كل من مجموعة بيانات التدريب والاختبار، يليه InceptionV3 وVGG16 وResNet50. ومع ذلك، نظرًا لقيود بيئة Android، تم اختيار VGG16 كنموذج نهائي للنشر في تطبيق الهاتف المحمول. يوفر التطبيق المطور واجهة مستخدم سهلة الاستخدام ويوضح فعالية نقل التعلم في تحديد وتشخيص أمراض النبات.

كلمات المفتاحية:

التعلم العميق، نقل التعلم، VGG16، ResNet-50، Inception-V3، DenseNet-201، تطبيق Android، PlantVillage.

Abstract :

This project focuses on the application of transfer learning techniques in the field of plant disease detection and classification. The goal is to implement several transfer learning models and evaluate their performances, and then to develop an Android-based application that allows users to identify and diagnose diseases in apple and grape plants. In this project, we covered various aspects: deep learning, transfer learning methods, deployment strategies, evaluation, and development of the application. Four popular pre-trained models, namely Densenet201, VGG16, ResNet50, and InceptionV3, are used for feature extraction and classification. The performance of these models is evaluated on "PlantVillage" dataset, measuring accuracy, training loss, test accuracy, and test loss. The results show that Densenet201 achieves the highest accuracy on both training and test sets, followed by InceptionV3, VGG16, and ResNet50. However, due to the size constraints of the Android environment, VGG16 is chosen as the final model for deployment in the application. The developed application provides an intuitive user interface and demonstrates the effectiveness of transfer learning in accurate plant disease identification and diagnosis.

Key words:

Deep learning, transfer learning, VGG16, ResNet-50, Inception-V3,

DenseNet-201, Android Mobile Application, PlantVillage

Table des matières :

Remerciements : ..... **i**
Dédicace : ..... **ii**
Résumé : ..... **iv**
Table des matières : ..... **vii**
Abréviations : ..... **x**
Liste Des Figures : ..... **xi**
Liste Des Tableaux : ..... **xiii**
Introduction Générale : **1**
Chapitre 1: Deep Learning **3**
1.1. Introduction: 4
1.2. Apprentissage automatique : 4
1.3. Apprentissage profondeur : 5
1.4. Apprentissage en profond vs l'apprentissage automatique : 6
1.5. Apprentissage supervisé : 7
1.6. Apprentissage non supervisé 8
1.7. Apprentissage semi-supervisé : 9
1.8. Architectures de l'apprentissage profond : 9
1.81. Réseau de Neurones Artificiels : 9
1.8.2. Réseau Neuronal Convolutif (CNN) : 10
1.8.3. Réseaux de Neurones Récurrents : 11
1.9. Concepts d 'apprentissage en profond: 12
1.9.1. Propagation avant : 12
1.9.2. Propagation en arrière : 12
1.9.3. Descente de gradient : 13
1.9.4. Epoch : 14
1.9.5. Batch size : 14
1.9.6. Training : 15
1.9.7. Validation : 15
1.9.8. Testing : 15
1.9.9. Over fitting : 16
1.9.10. Under fitting : 16
1.9.11. Loss function: 17
1.9.12. Taux d'apprentissage : 17
1.10. Applications of Deep Learning : 18
1.11. Conclusion : 19

Chapitre2 : Réseau Neuronal Convolutif et apprentissage par transfert. 20
2.1. Introduction : 21
2.2. Image Net : 21
2.3. Important CNN layers : 22
2.3.1. Couche convolutive (Convolution Layer) : 22
2.3.2. Couche de « pooling » : 23
2.3.3. Dropout : 25
2.3.4. Couche Flatten (Flatten layer) : 26
2.3.5. Couches denses (dense layer) 26
2.4. L'apprentissage par transfert (Transfer Learning) : 27
2.5. Quelques concepts de L'apprentissage par transfert : 28
2.6. Certains modèles pour L'apprentissage par transfert : 28
2.6.1. VGG (Visual Geometry Group) 29
2.6.2. Residual Network (ResNet) : 30
2.6.3. Dense Net : 31
2.6.4. Inception : 32
2.7. Conclusion : 34
Chapitre 3 : Méthodes de déploiement des modèles. 35
3.1. Introduction : 36
3.2. Méthodes de déploiement : 36
3.2.1. Flask : 36
3.2.2. Django : 37
3.2.3. Streamlit : 38
3.2.4. Docker : 39
3.2.5. Gradio 40
3.2.6. Tensor Flow lite : 40
3.3. Conclusion : 41
Chapitre 4 : Expérimentation 42
4.1. Introduction : 43
4.2. Agriculture intelligente : 43
4.3. Quelques exemples concrets d'agriculture intelligente : 44
4.4. Description du Dataset : 44
4.5. Environnement et outils de développement : 47
4.5.1. Google Co-laboratory : 47
4.5.2. Modules Python utilisés : 48
4.6. Modèles pré-entraînées utiliser : 49

4.7. Les hyperparamètres des modèles pré-entraîné : 49
4.7.1. Détails du réglage VGG-16 : 51
4.7.2. Détails du réglage de Inception V3 : 51
4.7.3. Détails du réglage de ResNet-50 : 51
4.7.4. Détails du réglage de DenseNet-201 : 52
4.8. Modification apportées aux modèles utiliser : 52
4.9. Prétraitement et augmentation des données : 53
4.10. Résultats et Discussion : 53
4.10.1. Pour le model VGG16 : 53
4.10.2. Pour le model ResNet-50 : 55
4.10.3. Pour le model InceptionV3 : 56
4.10.4. Pour le model Dense Net-201 : 58
4.11. Comparez les résultats des modelés : 59
4.12. Conclusion : 60
Chapitre 5 : Le déploiement sur une application mobile 61
5.1. Introduction : 62
5.2. Pourquoi le VGG16 : 62
5.3. Environnement et outils de développement : 62
5.3.1. JavaScript : 62
5.3.2. Android Studio : 63
5.4. Notre application : 64
5.4.1. Fonctionnent d'application : 64
5.4.2. Sélectionnez votre plante : 65
5.4.3. Interface de prédiction : 66
5.4.4. La liste des maladies : 67
5.4.5. Accès aux blogs d'informations : 68
5.5. Conclusion : 69
Conclusion Générale : 70
Bibliographie : 73

Abréviations :

- **ANN :** Un réseau de neurones artificiels**.**
- **API :** Application programming interface.
- **CNN :** réseau neuronal convolutif.
- **CPU :** Central Processing Unit.
- **CSRF :** Cross-Site Request Forgery.
- **Dense Net :** Densely Connected Convolutional Networks.
- **GPU :** Graphics Processing Unit.
- **HTTP :** Hypertext Transfer Protocol.
- **IA :** L'intelligence artificielle.
- **IDE :** Integrated Development Environment.
- **LSVRC :** Image Net Large Scale Visual Recognition Challenge.
- **OCR :** Reconnaissance optique de caractères.
- **PNJ :** personnages non joueurs.
- **ResNet :** Residual Network.
- **RNN :** Récurrent neural networks.
- **SDK :** Software Development Kit.
- **SGD :** Descente de gradient stochastique.
- **TF Lite :** TensorFlow Lite.
- **TPU :** Tensor Processing Unit.
- **VGG :** Visual Geometry Group.
- **Xss :** Cross-Site Scripting.

Liste Des Figures :

Figure 1 : Relation entre IA et l'apprentissage profond et l'apprentissage automatique. 4
Figure 2 : Example architecture de l'apprentissage en profond. 5
Figure 3 : Apprentissage en profond vs l'apprentissage automatique. 7
Figure 4 : Example apprentissage supervisé. 8
Figure 5 : Example apprentissage non supervisé. 9
Figure 6 : Réseau de Neurones Artificiels**.** 10
Figure 7: Réseau Neuronal Convolutif. 11
Figure 8 : Réseaux de Neurones Récurrents**.** 12
Figure 9: Over fitting 16
Figure 10: Under fitting. 17
Figure 11: Couche convolutive 23
Figure 12 : Max Pooling **.** 24
Figure 13 : Moyen pooling**.** 24
Figure 14 : Globale pooling. 25
Figure 15 : Un application le Dropout sur Neural net. 25
Figure 16 : Couche Flatten. 26
Figure 17 : Les couches denses 27
Figure 18 : Architecture de modèle VGG16. 29
Figure 19 : Architecture de modèle ResNet-50 31
Figure 20 : Architecture de modèle Dense Net-201. 32
Figure 21: Architecture de modèle L'Inception v3. 34
Figure 22: Le logo de Flask 37
Figure 23 : Logo de Django. 38
Figure 24: Logo de Streamlit 39
Figure 25: Logo de Docker 39
Figure 26: Logo de Gradio 40
Figure 27 : Logo de Tensor Flow lite 41
Figure 28: Fonction « Call Backs ». 51
Figure 29 : Extraction des couches ajoute aux modèles. 52
Figure 30 : Courbes de résultats de model VGG-16 affichage validation_ accuracy et accuracy. 54
Figure 31 : Courbes de résultats de model VGG-16 affichage loss et validation_loss. 54
Figure 32: Courbes de résultats de model ResNet-50 affichage validation_ accuracy et accuracy. 55
Figure 33: Courbes de résultats de model ResNet-50 affichage loss et validation_loss. 56
Figure 34 : Courbes de résultats de model Inception-v3 affichage validation_ accuracy et accuracy. 57
Figure 35 : Courbes de résultats de model Inception-v3 affichage loss et validation_loss. 57
Figure 36 : Courbes de résultats de model DenseNet-201 affichage validation_ accuracy et accuracy. 58
Figure 37: Courbes de résultats de model DenseNet-201 affichage loss et validation_loss. 59
Figure 38 : Interface de bienvenue. 64

Figure 39: Interfaces de conseils d'utilisation 65
Figure 40 : Interface « Sélectionnez votre plante ». 66
Figure 41: Interfaces « de prédiction ». 67
Figure 42: Interface « La liste des maladies ». 67
Figure 43: Exemple des interfaces « accès aux blogs d'informations » 68

Liste Des Tableaux :

Tableau 1 : partager de dataset “PlantVillage”(test,validation).....47
Tableau 2 : L’architecture des Modèles pré-entraînée.....49
Tableau 3 : Les hyperparameters de les modèles Pre-Trained.....50
Tableau 4 : Représenté les résultats des Modèles.....59

Introduction Générale :

Dans l’agriculture, les maladies des plantes sont un facteur majeur qui affecte la production. Il est donc important de les détecter et de les traiter rapidement pour éviter des pertes de récoltes importantes. La détection manuelle des maladies des plantes est un processus fastidieux et coûteux, qui peut être simplifié par l’utilisation de techniques de vision par ordinateur et de l'apprentissage profond. L’apprentissage par transfert (Transfer Learning) est couramment utilisé en reconnaissance d'image pour résoudre des problèmes similaires en utilisant des modèles de réseaux de neurones pré-entraînés sur de grandes quantités de données. Par exemple, le modèle VGG16 peut être utilisé pour la reconnaissance d'images de plantes malades. Un autre exemple est le modèle ResNet-50 qui a été pré-entraîné sur un ensemble de données ImageNet, et qui peut être utilisé pour la détection de maladies de la vigne. Le modèle Inception-V3 peut également être utilisé pour la classification de feuilles de thé en utilisant l’apprentissage par transfert. En utilisant ces modèles pré-entraînés, il est possible d'obtenir des résultats de haute qualité avec un temps d'entraînement considérablement réduit.

L'objectif de ce projet de fin d'études est :

- Acquérir des connaissances liées au domaine du l'apprentissage profond et en particulier comprendre les notions de réseaux de neurones convolutifs (CNN) Comprendre le concept d’apprentissage par transfert, et son intérêt.
- Implémenter un certain nombre de modèles d’apprentissage par transfert et évaluer leurs performances sur le dataset ‘plantvillage’ qui contient plus de 54000 images de feuilles de plantes réparties sur 38 classes différentes, et ensuite faire une comparaison de résultats obtenu.
- Déployer un modèle d’apprentissage par transfert sur téléphone mobile dans le cadre d’une application Android, ce qui permettra le passage du domaine théorique vers une implémentation d’une application rélle.
- Ce type de projets à une grande importance pratique, car il permettra aux agriculteurs de détecter rapidement les maladies des plantes et de prendre des mesures pour les traiter avant qu'ils ne se propagent et n'affectent la production.

Dans les chapitres suivants, nous allons décrire plus en détail les différents aspects de notre projet, y compris les techniques d’apprentissage par transfert, les architectures de réseau de neurones profonds, les méthodes de déploiement et les outils que nous allons utiliser.

Chapitre 1: Deep Learning

1.1. Introduction:

L'intelligence artificielle (IA) est un vaste domaine de l'informatique qui se concentre sur la création de machines intelligentes capables d'exécuter des tâches qui nécessiteraient généralement une intelligence au niveau humain. Les systèmes d'intelligence artificielle peuvent être formés pour reconnaître des modèles, faire des prédictions et résoudre des problèmes en utilisant diverses techniques, y compris l'apprentissage automatique et l'apprentissage profondeur.

En résumé, l'intelligence artificielle est un vaste domaine de l'informatique qui englobe de nombreux sous domaines différents, l'apprentissage automatique implique des machines de formation pour apprendre des données. Apprentissage profond a eu un impact profond sur de nombreux domaines de l'intelligence artificielle, et ses applications continuent de croître à mesure que la recherche progresse dans ce domaine.Figure 1 illustre la relation entre l'intelligence artificielle l'apprentissage profond et l'apprentissage automatique.

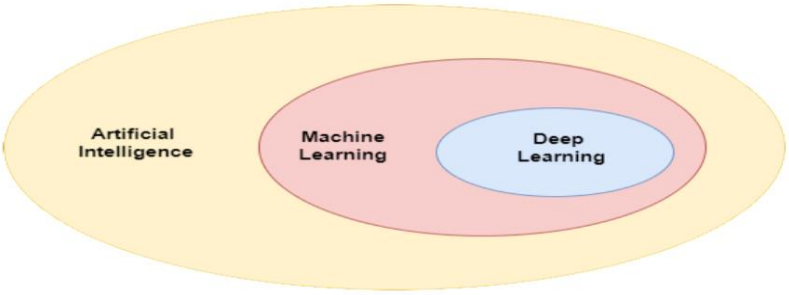


Figure 1 : Relation entre IA et l'apprentissage profond et l'apprentissage automatique.

1.2. Apprentissage automatique :

Apprentissage automatique (Machine Learning) : est un domaine de l'intelligence artificielle qui permet aux ordinateurs d'apprendre à partir de données sans nécessiter une programmation explicite. L'objectif de l'apprentissage automatique est de développer des algorithmes qui permettent aux ordinateurs de reconnaître des modèles dans les données et d'utiliser ces modèles pour faire des prédictions ou prendre des décisions.

À l'aide de techniques de modélisation statistique et mathématique, l'apprentissage automatique crée des modèles qui peuvent être formés à partir les données existantes et utilisés pour prédire les résultats sur de nouvelles données. Les applications d'apprentissage automatique sont diverses et incluent la reconnaissance vocale, la reconnaissance d'images, la prédiction des fraudes, les recommandations de produits, etc. [1]

1.3. Apprentissage profondeur :

L’apprentissage profond (Deep Learning) est une sous-discipline de l'apprentissage automatique qui utilise des réseaux de neurones artificiels profonds pour apprendre des représentations hiérarchiques des données.

Les réseaux de neurones artificiels sont constitués de couches de neurones interconnectés, ce qui leur permet d'apprendre des modèles de données de plus en plus complexes. Les algorithmes d'apprentissage en profondeur sont capables d'identifier des modèles dans les données qui sont difficiles pour les méthodes d'apprentissage automatique traditionnelles.

L'apprentissage en profondeur est utilisé pour des tâches telles que la reconnaissance d'images, la reconnaissance vocale, la traduction automatique et la prédiction du prochain mouvement d'un utilisateur sur un site Web. Il a connu un développement important ces dernières années en raison de l'augmentation de la puissance de calcul et de l'utilisation florissante des réseaux de neurones profonds dans de nombreuses applications dans différents domaines [2]. Figure 2 illustre Un exemple d’architecture d 'apprentissage en profond.

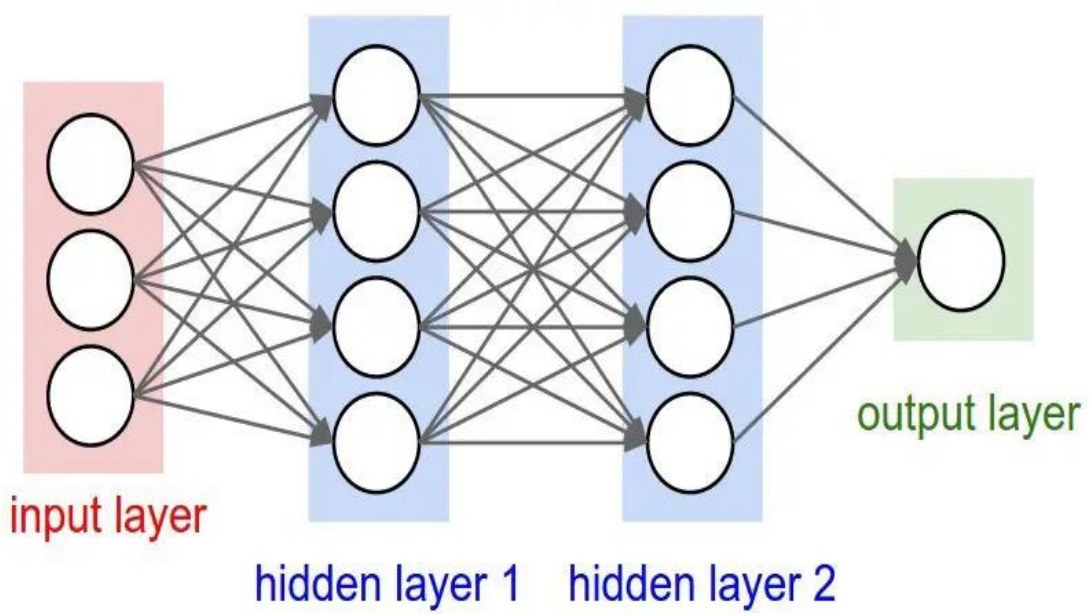


Figure 2 : Example architecture de l'apprentissage en profond.

1.4. Apprentissage en profond vs l'apprentissage automatique :

L'apprentissage en profond est une branche de l'apprentissage automatique qui utilise des réseaux de neurones artificiels profonds pour apprendre des représentations hiérarchiques des données, tandis que l'apprentissage automatique se concentre sur la création d'algorithmes permettant aux ordinateurs d'apprendre à partir de données sans être explicitement programmés.

La principale différence entre l'apprentissage en profondeur et l'apprentissage automatique traditionnel est la complexité du modèle d'apprentissage.

L'apprentissage en profondeur utilise des réseaux de neurones profonds qui peuvent apprendre des représentations hiérarchiques des données, leur permettant de détecter des modèles très complexes et de reconnaître des objets, des sons, du texte, etc. avec une grande précision.

En revanche, l'apprentissage automatique traditionnel utilise généralement des algorithmes plus simples qui ne peuvent pas modéliser la complexité des données aussi précisément que l'apprentissage en profondeur. Cependant, l'apprentissage automatique peut être plus facile à configurer et à utiliser pour des tâches plus simples.

Pour résumer, l'apprentissage en profondeur est une technique avancée d'apprentissage automatique qui peut obtenir des résultats très précis pour des tâches complexes comme la reconnaissance d'images ou de la parole, tandis que l'apprentissage automatique peut être utilisé pour des tâches plus simples et plus courantes comme la classification ou la prédiction [3]. Figure 13 illustre différence entre l'apprentissage en profond et l'apprentissage automatique.

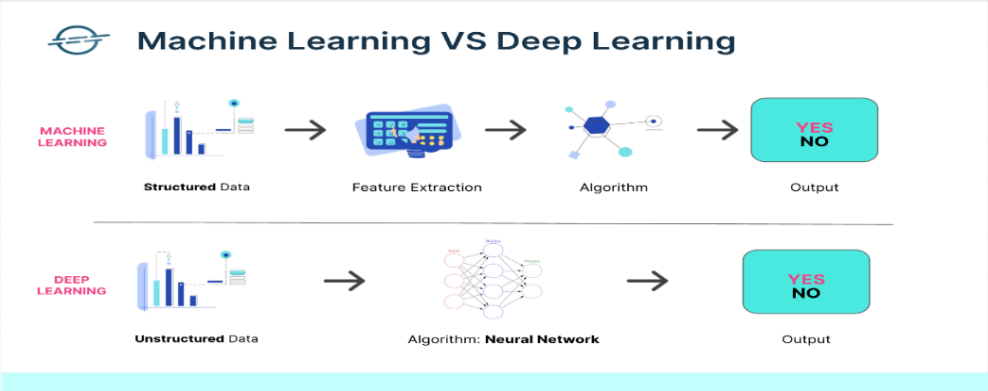


Figure 3 : Apprentissage en profond vs l'apprentissage automatique.

1.5. Apprentissage supervisé :

L'apprentissage supervisé est une méthode d'apprentissage automatique dans laquelle un algorithme est formé sur un ensemble de données étiquetées pour apprendre à prédire une sortie (ou une étiquette) en fonction de l'entrée.

Plus précisément, dans l'apprentissage supervisé, chaque échantillon de données dans l'ensemble d'apprentissage se compose d'une entrée (également appelée caractéristique ou variable explicative) et d'une sortie correspondante (également appelée variable cible ou étiquette). En apprenant à partir de ces exemples de données étiquetées, des algorithmes d'apprentissage supervisé sont formés pour prédire la sortie correcte pour chaque entrée.

L'objectif de l'apprentissage supervisé est de généraliser à partir des données de formation afin que des sorties précises puissent être prédites pour de nouvelles entrées qui n'ont pas été vues auparavant.

Les exemples courants d'apprentissage supervisé incluent la classification (le but est de prédire une variable cible discrète) et la régression (le but est de prédire une variable cible continue). L'apprentissage supervisé est largement utilisé dans de nombreuses applications, notamment la reconnaissance vocale, la détection de fraude, la recommandation de produits, [4]. Figure 4 est un exemple d'apprentissage supervisé.

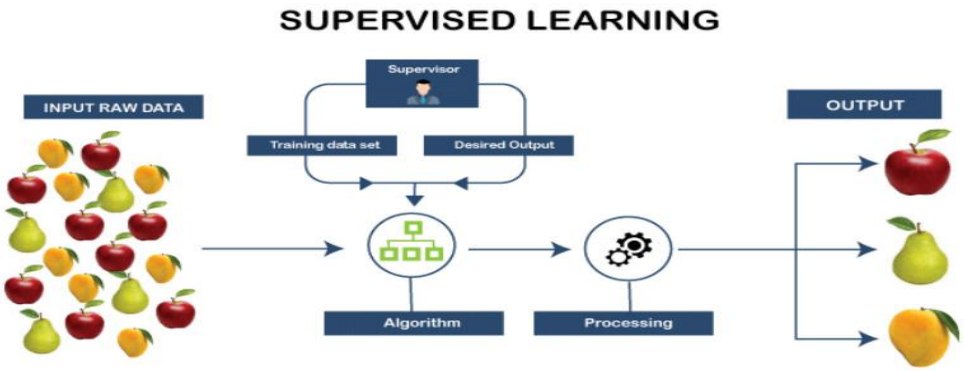


Figure 4 : Example apprentissage supervisé.

1.6. Apprentissage non supervisé

L'apprentissage non supervisé est une méthode d'apprentissage automatique dans laquelle un algorithme est formé sur un ensemble de données non étiquetées pour découvrir des structures ou des modèles dans les données.

Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé n'utilise pas de variables cibles étiquetées pour former un algorithme, mais cherche plutôt à identifier des structures ou des relations significatives dans les données sans direction préalable.

Le but de l'apprentissage non supervisé est de découvrir des regroupements naturels, des modèles ou des anomalies dans les données. Des algorithmes d'apprentissage non supervisés peuvent être utilisés pour regrouper des éléments similaires en ensembles (clusters), pour réduire la dimensionnalité des données ou pour trouver des modèles cachés dans des données non structurées telles que des images, du texte ou des signaux audios.

Les exemples courants d'apprentissage non supervisé incluent le clustering(regroupement), réduction de la dimensionnalité (comme analyse en composantes principales) et détection d'anomalies. Apprentissage non supervisé est utilisé dans de nombreux domaines, notamment l'analyse de données, reconnaissance de formes, vision par ordinateur [5]. Figure 5 est un exemple d'apprentissage non supervisé.

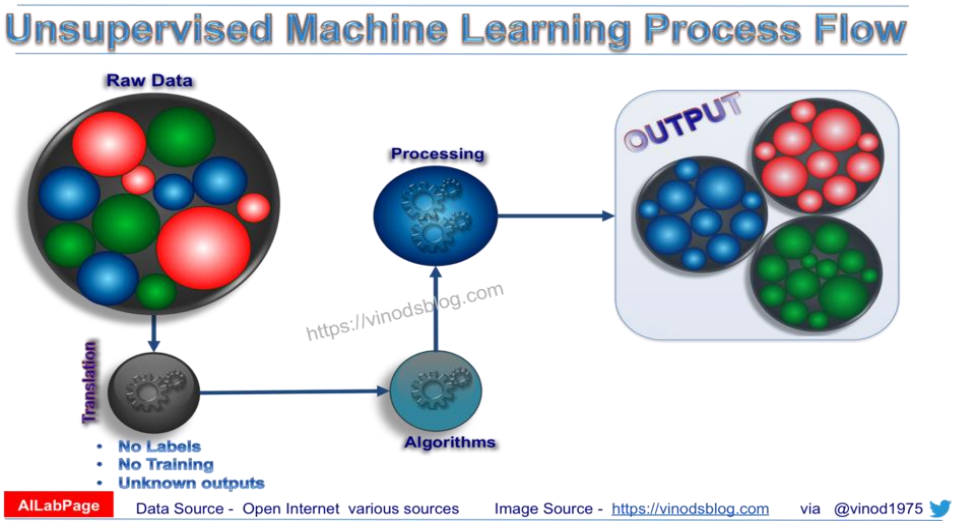


Figure 5 : Example apprentissage non supervisé.

1.7. Apprentissage semi-supervisé :

Il s'agit d'un type d'apprentissage automatique qui combine des éléments d'apprentissage supervisé et non supervisé. Le modèle est formé sur un ensemble de données partiellement étiqueté, avec certaines données d'entrée accompagnées de valeurs cibles et d'autres sans étiquettes. L'objectif est de tirer parti des données étiquetées pour améliorer l'exactitude des prévisions du modèle pour les données non étiquetées [6].

1.8. Architectures de l'apprentissage profond :

1.81. Réseau de Neurones Artificiels :

Un réseau de neurones artificiels est un modèle mathématique inspiré du fonctionnement du cerveau humain. Ils sont constitués d'un grand nombre de nœuds interconnectés appelés neurones artificiels. Chaque neurone est relié à plusieurs autres neurones par des connexions pondérées par des coefficients appelés poids.

Les réseaux de neurones sont capables d'apprendre à partir des données d'entrée en modifiant les poids des connexions entre les neurones. Ils sont particulièrement adaptés à des tâches telles que la classification, la prédiction ou la reconnaissance de formes en raison de leur capacité à révéler des structures et des relations complexes dans les données.

Les réseaux de neurones peuvent avoir différentes architectures, qui déterminent la façon dont les neurones sont organisés et connectés les uns aux autres. Les architectures courantes comprennent les réseaux de couches denses (également appelés réseaux multicouches), les réseaux de neurones convolutifs pour la vision par ordinateur et les réseaux de neurones récurrents pour le traitement de données séquentielles.

La formation d'un réseau de neurones peut être supervisée, où les données d'entrée sont étiquetées avec la sortie souhaitée pour former le réseau, ou non supervisée, où le réseau doit découvrir la structure des données sans étiquettes. Figure 1.6 illustre son architecture.

Les réseaux de neurones ont connu un grand succès dans de nombreux domaines, tels que la reconnaissance d'images, la reconnaissance vocale, la traduction automatique, le traitement du langage naturel, la prédiction de séries chronologiques, et bien d'autres [7].

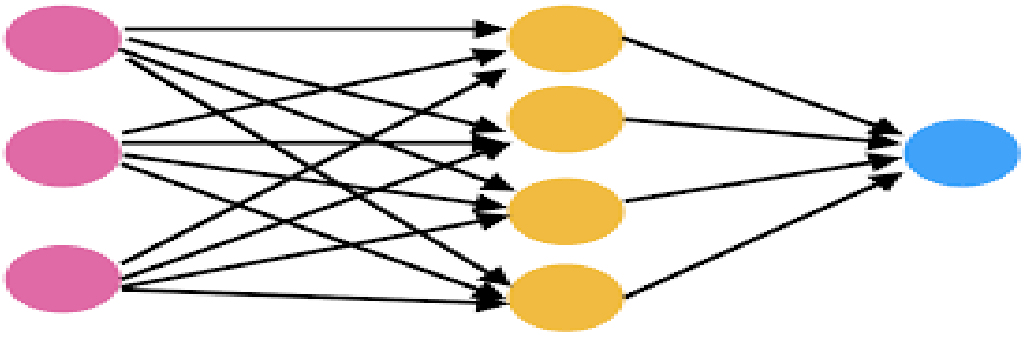


Figure 6 : Réseau de Neurones Artificiels*.*

1.8.2. Réseau Neuronal Convolutif (CNN) :

Le réseau neuronal convolutif est une classe de réseaux neuronaux spécialement conçus pour la reconnaissance de formes dans les images. Réseau Neuronal Convolutif extraient automatiquement les caractéristiques importantes des images à l'aide d'une technique appelée convolutions.

La complication est un processus mathématique qui permet de filtrer une image en la faisant passer à travers une matrice de poids appelée filtre ou noyau. Le filtre est appliqué à chaque région de l'image, en multipliant les pixels de l'image par les portions du filtre, puis en additionnant les résultats pour produire un seul nombre qui représente cette région de l'image. Le résultat est ensuite stocké dans une nouvelle image.

Réseau Neuronal Convolutif sont composés de plusieurs couches de neurones, chacune avec une fonctionnalité spécifique. La première couche est généralement une couche de convolution qui extrait des caractéristiques de base telles que des bords et des textures. Les couches suivantes utilisent

des complications plus complexes pour extraire des caractéristiques plus abstraites, telles que des formes et des objets.Figure 7 illustre son architecture.

Réseau Neuronal Convolutif peuvent également inclure des couches de pooling, qui réduisent la taille de l'image, par exemple : en prenant la valeur maximale ou moyenne dans chaque région. Les couches de pooling permettent de réduire la quantité de données que le réseau doit traiter, tout en préservant les caractéristiques les plus importantes

Réseau Neuronal Convolutif ont été utilisés avec succès dans de nombreux domaines de vision par ordinateur, tels que la surveillance d'objets, la détection de visages, la segmentation d'images, le support d'images médicales, la surveillance de caractères manuscrits, etc [8].

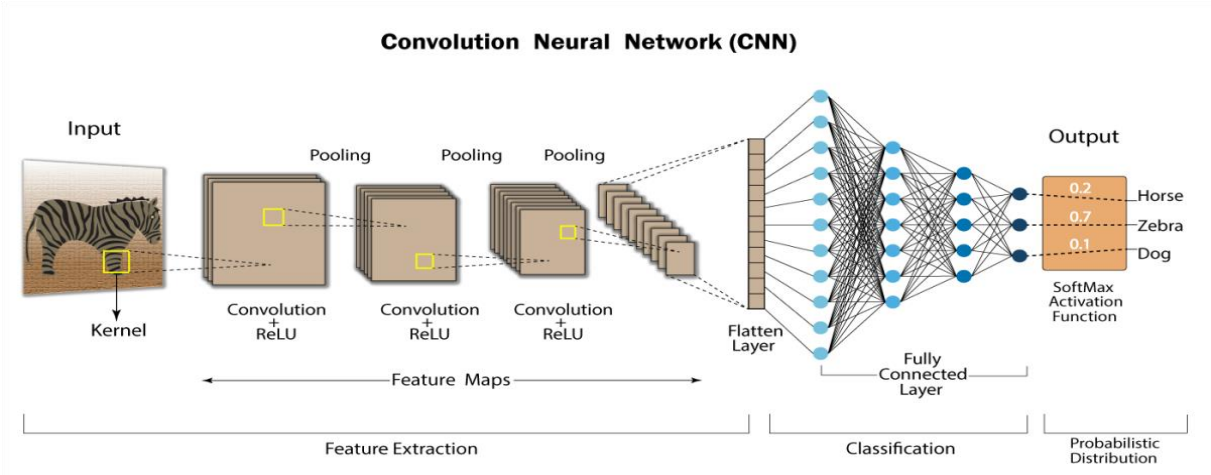


Figure 7: Réseau Neuronal Convolutif.

1.8.3. Réseaux de Neurones Récurrents :

Les réseaux de neurones récurrents sont une classe de réseaux de neurones qui sont conçus pour traiter des données séquentielles, telles que des séquences de mots dans du texte ou des séquences de valeurs temporelles dans des signaux temporels.

Les réseaux de neurones récurrents sont constitués de cellules qui contiennent des neurones, ainsi qu'un état interne appelé mémoire à court terme ou cellule de mémoire. L'état de la mémoire est mis à jour à chaque étape de temps en utilisant une fonction de rétroaction qui combine l'entrée actuelle avec l'état précédent de la mémoire. Cette fonction de rétroaction peut prendre forme d'une simple opération d'addition matricielle, ou être plus complexe en utilisant des portes pour contrôler l'écriture et la lecture de l'état de la mémoire. Figure 8 illustre son architecture.

Les réseaux de neurones récurrents sont capables de modéliser des séquences de longue durée en mémorisant des informations précédentes, ce

qui leur permet de capturer des dépendances à long terme dans les données. Les RNN peuvent également être empilés en couches pour former des réseaux de neurones récurrents profonds, appelés Deep intermittent Neural Networks (DRNN), qui sont capables de capturer des dépendances encore plus complexes.

Les réseaux de neurones récurrents ont été largement utilisés dans de nombreux domaines, tels que la surveillance de la parole, la traduction automatique, la génération de texte, la prédiction de séries temporelles et bien d'autres encore [9].

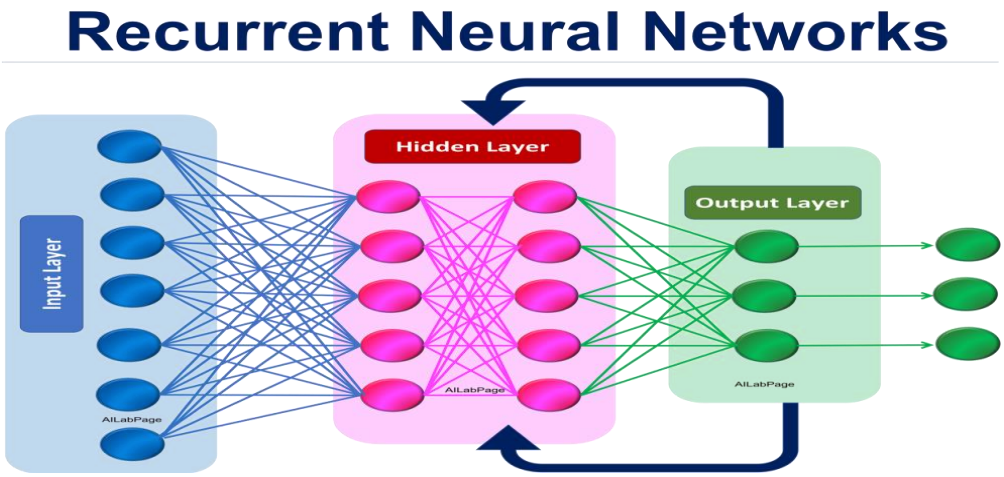


Figure 8 : Réseaux de Neurones Récurrents.

1.9. Concepts d 'apprentissage en profond:

1.9.1. Propagation avant :

La propagation avant (Forward propagation) est le processus de calcul de la sortie d'un réseau neuronal auquel une entrée est donnée. Pendant ce processus, l'entrée est passée par les couches du réseau neuronal, chaque couche effectuant un ensemble d'opérations mathématiques pour transformer l'entrée en une nouvelle représentation qui peut être utilisée par la couche suivante [10].

1.9.2. Propagation en arrière :

La propagation en arrière (Back-propagation) est un algorithme utilisé pour former les réseaux neuronaux. Il s'agit d'une technique d'apprentissage supervisé qui consiste à mettre à jour les poids et la mesure d'erreur d'un

réseau neuronal afin de minimiser la différence entre la sortie prévue et la sortie réelle [11].

1.9.3. Descente de gradient :

A. . Définition :

La descente de gradient est un algorithme d'optimisation utilisé pour minimiser les fonctions de perte dans les modèles d'apprentissage automatique. Le but de la descente de gradient est de trouver l'ensemble de poids (ou coefficients) qui minimise la fonction de perte. L'algorithme fonctionne en ajustant itérativement les poids dans le sens de la plus grande réduction de la fonction de perte [15].

B. Comment fonctionne La descente de gradient ?

La descente de gradient est de partir d'un ensemble initial de poids et de les mettre à jour vers le gradient négatif de la fonction de perte. Le gradient est un vecteur de dérivées partielles représentant le taux de variation de la fonction de perte par rapport aux poids. En mettant à jour les poids dans le sens des gradients négatifs, l'algorithme se déplace vers le minimum de la fonction de perte. Le taux d'apprentissage est un hyperparamètre qui détermine la taille de pas des mises à jour de poids. Un faible taux d'apprentissage conduit à une convergence lente, tandis qu'un taux d'apprentissage élevé conduit à dépasser et à osciller autour du minimum. Il est important de choisir un taux d'apprentissage approprié pour équilibrer la vitesse de convergence et optimiser la stabilité [12].

C. Les variantes de La descente de gradient :

1) Descente de gradient par lots :

Dans la descente de gradient par lots, le gradient de la fonction de perte est calculé à partir des poids de l'ensemble de données d'apprentissage complet, et les poids sont mis à jour après chaque itération. Cela fournit des estimations de gradient plus précises, mais peut être coûteux en calcul pour les grands ensembles de données [12].

2) Descente de gradient stochastique (SGD) :

Dans SGD, le gradient de la fonction de perte est calculé par rapport à un seul exemple d'apprentissage, et les pondérations sont mises à jour après

chaque exemple. SGD a un coût de calcul par itération inférieur à celui de la descente de gradient par lots, mais il peut être moins stable et ne pas converger vers la solution optimale [12].

3) Descente de gradient en mini-lot :

La descente de gradient en mini-lot est un compromis entre la descente de gradient en lot et SGD. Le gradient de la fonction de perte est calculé par rapport à un petit sous-ensemble sélectionné au hasard des exemples d'apprentissage (appelé mini-lot), et les poids sont mis à jour après chaque mini-lot. La descente de gradient de mini-lot fournit un équilibre entre la stabilité de la descente de gradient de lot et l'efficacité de calcul de SGD [12].

4) Momentum :

Momentum est une variante de la descente de gradient qui intègre les informations des mises à jour de poids précédentes pour aider l'algorithme à converger plus rapidement vers une solution optimale. Momentum ajoute un terme proportionnel à la moyenne mobile des gradients passés à la mise à jour du poids, permettant à l'algorithme de se déplacer plus rapidement vers la solution optimale [12].

1.9.4. Epoch :

Une Epoch (ou époque) correspond au moment où toutes les données d'apprentissage sont utilisées en même temps et est définie comme le nombre total d'itérations de toutes les données d'apprentissage dans un cycle pour l'apprentissage du modèle.

Le nombre des Epochs est considéré comme un hyperparamètre. Il définit le nombre de fois que l'ensemble de données complet doit être traité par L'algorithme d'apprentissage.

Chaque échantillon de l'ensemble de données d'apprentissage a eu la possibilité de mettre à jour les paramètres du modèle interne une fois au cours d'une Epoch [13].

1.9.5. Batch size :

Batch size est un hyperparamètre qui définit le nombre d'échantillons prélevés pour travailler sur un modèle d'apprentissage automatique particulier avant de mettre à jour ses paramètres de modèle interne. Batch peut être

considéré comme une boucle for itérant sur un ou plusieurs échantillons et effectuant des prédictions. Ces prédictions sont ensuite comparées aux variables de sortie attendues à la fin du batch. L'erreur est calculée en comparant les deux, puis utilisée pour améliorer le modèle.

Un ensemble de données d'entraînement peut être divisé en plusieurs Batch. S'il n'existe qu'un seul batch, que toutes les données d'apprentissage sont dans un seul Batch, alors l'algorithme d'apprentissage est appelé descente de gradient par Batch [14].

1.9.6. Training :

Un ensemble de données d'apprentissage est un ensemble de données d'exemples utilisé pendant le processus d'apprentissage est utilisé pour adapter les paramètres (les poids) d'un classifieur.

Pour les tâches de classification, un algorithme d'apprentissage supervisé examine un ensemble de données d'apprentissage pour déterminer ou apprendre la combinaison optimale de variables qui produira un bon modèle prédictif. L'objectif est de créer un modèle formé (adapté) qui se généralise bien aux nouvelles données inconnues. Le modèle ajusté est évalué à l'aide de "nouveaux" exemples tirés des ensembles de données retenus (ensembles de données de validation et de test) pour estimer la précision du modèle dans la classification des nouvelles données. Pour réduire le risque de problèmes tels que le surajustement (over fit). [15].

1.9.7. Validation :

L'ensemble de validation, fait partie de l'ensemble de training. Il est utilisé pour optimiser la sélection des hyperparamètres pour un algorithme d'apprentissage plus précisément, le modèle est formé avec divers hyperparamètres sur le training-set réduite la variante de modèle qui fonctionne le mieux sur l'ensemble de validation est sélectionnée et entraînée à nouveau en utilisant training set complet [16].

1.9.8. Testing :

Un ensemble de données de test est un ensemble de données indépendant de l'ensemble de données training, mais qui suit la même distribution de probabilité que l'ensemble de données training. Si le modèle qui correspond à l'ensemble de données d'apprentissage correspond également bien à l'ensemble de données de test, il y a un surajustement minimal.

Ainsi, l'ensemble de test est un ensemble d'exemples utilisés uniquement pour évaluer les performances (c'est-à-dire la généralisation) d'un classificateur

entièrement spécifié. Ces prédictions sont comparées aux classifications réelles des exemples pour évaluer la précision du modèle [17].

1.9.9. Over fitting :

On considère qu'un modèle est surajusté lorsqu'il donne d'excellents résultats sur les données d'apprentissage, mais qu'il n'est pas aussi performant sur les données de validation. Mais ne parvient pas à obtenir le même niveau de performance sur les données de validation. Un modèle surajusté ne parvient pas à généraliser correctement, car il apprend le bruit et les modèles des données d'apprentissage au point de nuire à la qualité des données. Données de training au point d'avoir un impact négatif sur les performances du modèle sur les nouvelles données. Si le modèle est surajusté, même un léger changement dans les données de sortie entraînera un changement significatif du modèle [18]. Figure 9 illustre son fonctionnement.

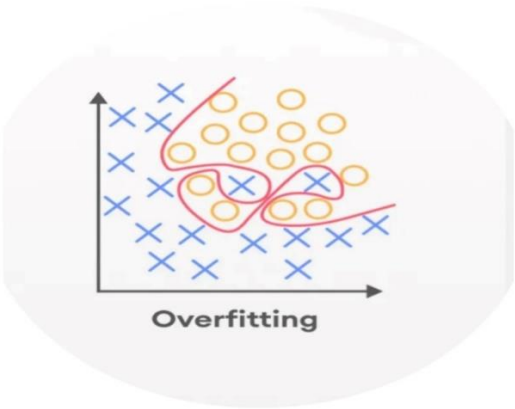


Figure 9: Over fitting.

1.9.10. Under fitting :

Il y a sous-adaptation lorsqu'un modèle n'est pas en mesure de faire des prédictions précises sur la base des données d'apprentissage et, par conséquent, n'a pas la capacité de bien généraliser sur de nouvelles données. Un autre cas de sous-adaptation est celui où un modèle n'est pas en mesure d'apprendre suffisamment à partir des données d'apprentissage, ce qui rend difficile la capture de la tendance dominante. Les modèles d'apprentissage automatique avec sous-adaptation ont tendance à avoir de mauvaises performances à la fois dans les ensembles de training et de test ensembles de test. Les modèles sous-adaptés présentent généralement une mesure d'erreur élevé et une faible variance [18]. Figure 10 illustre son fonctionnement.

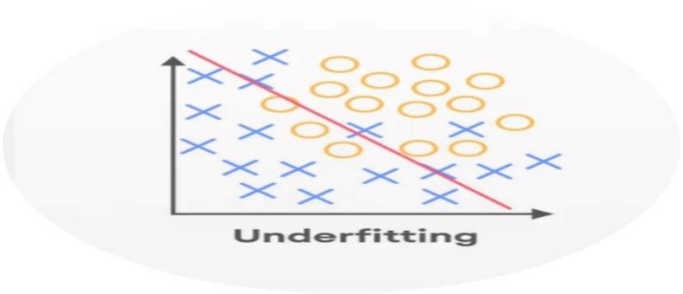


Figure 10: Under fitting.

1.9.11. Loss function:

La fonction de perte est une mesure de la performance d'un modèle. Elle est utilisée pour évaluer la performance d'un modèle et pour optimiser ses paramètres. La fonction de perte mesure la différence entre les valeurs prédites et les valeurs réelles. Elle est utilisée pour calculer le taux d'erreur d'un modèle et pour ajuster ses paramètres afin de minimiser le taux d'erreur [19].

1.9.12. Taux d'apprentissage :

En apprentissage automatique, le taux d'apprentissage (Learning rate) est un hyperparamètre qui contrôle la taille des pas de l'algorithme d'optimisation de la descente de gradient pendant le training. L'algorithme de descente de gradient est utilisé pour minimiser la fonction de perte d'un modèle en ajustant les poids et les mesure d'erreur du modèle dans la direction de la descente la plus rapide du gradient.

Taux d'apprentissage détermine la taille des étapes prises dans l'algorithme de descente de gradient. Un taux d'apprentissage plus élevé signifie de plus grandes étapes, tandis qu'un taux d'apprentissage plus faible signifie de plus petites étapes. Si le taux d'apprentissage est trop élevé, l'algorithme peut dépasser le minimum de la fonction de perte et ne pas converger vers la solution optimale. D'autre part, si le taux d'apprentissage est trop faible, l'algorithme peut converger trop lentement et se coincer dans une solution sous-optimale.

Choisir un taux d'apprentissage approprié est une partie essentielle du training d'un modèle d'apprentissage automatique. Voici quelques stratégies courantes pour choisir un Learning rate :

- Taux d'apprentissage fixe : Utilisez un taux d'apprentissage fixe tout au long du training.
- Taux d'apprentissage Horaires : réduire progressivement le Learning rate au fil du temps, par exemple par un facteur de 2 après chaque n Epoch.
- Taux d'apprentissage adaptatif : Utilisez des algorithmes qui ajustent le taux d'apprentissage en fonction des progrès de la formation, comme Adam ou Adagrad.

Il n'existe pas universelle pour sélectionner le taux d'apprentissage, car le meilleur choix dépend du problème spécifique et de l'ensemble de données utilisé. L'expérimentation et l'ajustement sont souvent nécessaires pour trouver un taux d'apprentissage approprié qui permet au modèle de converger rapidement et efficacement vers la solution optimale [20].

1.10. Applications of Deep Learning :

Le Deep Learning est une technique de Machine Learning qui utilise des réseaux de neurones profonds pour apprendre à partir de données complexes. Cette technique est utilisée dans de nombreuses applications pour résoudre des problèmes difficiles dans des domaines tels que la vision par ordinateur, le traitement du langage naturel, la reconnaissance vocale et la reconnaissance de formes. Voici quelques-unes des applications de Deep Learning les plus courantes :

1. Vision par ordinateur : Les réseaux de neurones profonds sont largement utilisés dans la vision par ordinateur pour la détection d'objets, la reconnaissance faciale, la segmentation d'images et la reconnaissance optique de caractères (OCR).
2. Traitement du langage naturel : Le Deep Learning est utilisé dans le traitement du langage naturel pour la traduction automatique, la génération de texte, la reconnaissance de la parole et l'analyse de sentiments.
3. Reconnaissance vocale : Les réseaux de neurones profonds sont utilisés dans la reconnaissance vocale pour la transcription de la parole, la reconnaissance de la parole et la synthèse vocale.
4. Recommandation de produits : Les entreprises de commerce électronique utilisent le Deep Learning pour recommander des produits à leurs clients en fonction de leurs préférences et de leur historique d'achat.

5. Santé : Les réseaux de neurones profonds sont utilisés pour l'analyse d'images médicales, la détection de maladies, la segmentation de tissus et la prédiction de résultats.
6. Jeux vidéo : Le Deep Learning est utilisé pour l'optimisation des personnages non joueurs (PNJ) dans les jeux vidéo et pour la création de jeux vidéo autonomes.
7. Automobile : Les voitures autonomes utilisent des réseaux de neurones profonds pour la reconnaissance des panneaux de signalisation, la détection de piétons, la reconnaissance des feux de circulation et la navigation autonome.
8. Finances : Les banques et les institutions financières utilisent le Deep Learning pour la détection de la fraude, la prédiction de risques et la détection de tendances.

En somme, le Deep Learning est une technologie polyvalente qui peut être utilisée dans de nombreux domaines pour résoudre des problèmes difficiles et réaliser des tâches qui étaient auparavant impossibles [21].

1.11. Conclusion :

En conclusion, le Deep Learning est une fashion d'apprentissage automatique qui a révolutionné de nombreux domaines, tels que la surveillance d'image, la surveillance de la parole, le traitement du langage naturel, la détection d'anomalies, la prédiction, la modélisation et bien d'autres. Les réseaux de neurones profonds sont capables d'apprendre des représentations hiérarchiques des données, ce qui leur permet de capturer des caractéristiques complexes et de réaliser des tâches complexes avec une précision élevée. Les modèles de Deep Learning sont capables de s’adapter à des ensembles de données de grande taille et sont donc très utiles pour les opérations nécessitant un traitement de données massif en temps réel. Grâce aux progrès de l'informatique et de la technologie, le Deep Learning a un potentiel énorme pour résoudre des problèmes complexes dans de nombreux domaines, tels que la santé, l'agriculture, l'industrie, les services financiers, les transports, etc.

Chapitre2 : Réseau Neuronal Convolutif et apprentissage par transfert.

2.1. Introduction :

L'apprentissage profond a révolutionné le domaine de la vision par ordinateur, et ImageNet, un ensemble de données d'images à grande échelle, a joué un rôle central dans cette révolution. ImageNet contient des millions d'images étiquetées et a utilisé comme un ensemble de données de référence pour la formation et l'évaluation des modèles de reconnaissance d'image.

Les réseaux de neurones convolutifs sont les modèles d'apprentissage profond les plus couramment utilisés pour les tâches de reconnaissance d'images. Les réseaux de neurones convolutifs comprennent plusieurs couches, y compris les couches convolutives, les couches de mise en commun et les couches entièrement connectées. Chacune de ces couches remplit une fonction spécifique dans l'architecture globale.

L'apprentissage par transfert est une technique populaire utilisée dans l'apprentissage profond pour transférer les connaissances d'une tâche à une autre. Dans l'apprentissage par transfert, des modèles Pré-entraîné sont utilisés comme point de départ pour une nouvelle tâche, et le modèle est affiné sur la nouvelle tâche pour améliorer ses performances.

Les modèles pré-entraînés sont des modèles d'apprentissage profond qui ont été pré-entraînés sur de grands ensembles de données, tels qu'ImageNet. Ces modèles sont déjà optimisés et formés pour des tâches spécifiques et peuvent être affinés sur de nouveaux ensembles de données pour atteindre une grande précision avec moins de temps de formation et de données.

2.2. Image Net :

ImageNet est une base de données d'images à grande échelle utilisée dans la recherche en vision par ordinateur. ImageNet a créé par des chercheurs de l'Université de Princeton et de l'Université de Stanford, ImageNet contient plus de 14 millions d'images étiquetées organisées en plus de 21.000 catégories. Les images dans ImageNet sont haute résolution et couvrent un large éventail de catégories d'objets, y compris les animaux, les personnes, les véhicules et plus encore.

ImageNet est largement utilisée comme un ensemble de données de référence pour évaluer la performance des algorithmes de vision par ordinateur, en particulier pour la reconnaissance des objets et les tâches de classification. Challenge le plus connu d'ImageNet est le Défi de reconnaissance visuelle à grande échelle Image Net (ILSVRC), qui a eu lieu chaque année de 2010 à 2017. Dans ce challenge, les participants ont formé leurs algorithmes sur l'ensemble de données Image Net et ont concouru pour obtenir la plus grande précision sur un grand ensemble d'images de test.

La disponibilité de l'ensemble de données d'Image Net et le succès du challenge de l'ILSVRC ont mené à des progrès importants dans la recherche en vision par ordinateur, y compris l'élaboration de modèles d'apprentissage profond comme les réseaux de neurones convolutifs (CNN) qui ont atteint l'état de performance de pointe sur l'ensemble de données Image Net.

La base de données Image Net est d'une importance fondamentale pour les tâches de classification liées à la vision de tous les principaux modèles préétablis proposé dans la littérature son initialement formés sur elle. Image Net se compose d'environ15 millions d'images marquées haute résolution séparées en environ 22 000 catégories. Les images dans la base de données sont récupérées par les moteurs de recherche et là-peut être considérée comme des données multimédias communes. Sous cadre Image Net, un concours annuel de classification est notamment la « reconnaissance visuelle à grande échelle ImageNet Défi », où les participants utilisent un sous-ensemble de ces données pour la formation des algorithmes de classification de leur choix. La taille réelle de l'ensemble de données de formation utilisé dans le concours consiste d'environ 1,2 million d'images, et les participants doivent ensemble de données d'essai en 1000 classes distinctes [22].

2.3. Important CNN layers :

Dans les réseaux de neurones convolutifs (CNN), plusieurs couches importantes sont couramment utilisées pour les tâches de reconnaissance d'images [26]. Voici de brèves définitions de certaines de ces couches :

2.3.1. Couche convolutive (Convolution Layer) :

Les couches convolutives sont les éléments de base de CNN. Il transporte une grande partie de la charge de calcul du réseau. Cette couche effectue un produit scalaire entre deux matrices, où une matrice est un ensemble de paramètres apprenables, également appelé noyau, et l'autre matrice est une partie restreinte du champ récepteur. Le noyau est spatialement plus petit mais plus profond que l'image. Cela signifie que si l'image se compose de canaux, la hauteur et la largeur du noyau seront spatialement petites, mais la profondeur sera étendue aux trois canaux [23]. Figure 11 représenté la Couche convolutive.

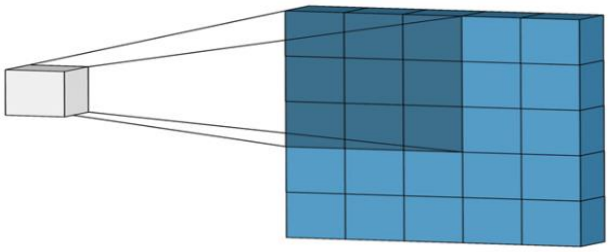


Figure 11: Couche convolutive .

2.3.2. Couche de « pooling » :

L'opération de pooling consiste à glisser un filtre bidimensionnel sur chaque canal de la carte des caractéristiques et à résumer les caractéristiques se trouvant dans la région couverte par le filtre [24].

a) Pourquoi utiliser couches de pooling?

Les couches de pooling sont utilisées pour réduire les dimensions des cartes de caractéristiques. Ainsi, il réduit le nombre de paramètres à apprendre et la quantité de calcul effectué dans le réseau.

La couche de pooling résume les caractéristiques présentes dans une région de la carte de caractéristiques générée par une couche de convolution. Ainsi, d'autres opérations sont effectuées sur les caractéristiques résumées au lieu aux caractéristiques précisément positionnées générées par la couche de convolution. Cela rend le modèle plus robuste aux variations de la position des caractéristiques dans l'image d'entrée [24].

b) Types de couches de pooling:

-Le Max Pooling : est une opération de mutualisation qui sélectionne l'élément maximum à partir de la région de la carte de fonctionnalités couverte par le filtre. Ainsi, la sortie après la couche max-pooling serait une carte de fonctionnalités contenant les caractéristiques les plus importantes de la précédente carte de fonctionnalités. Figure 12 représenté Max Pooling

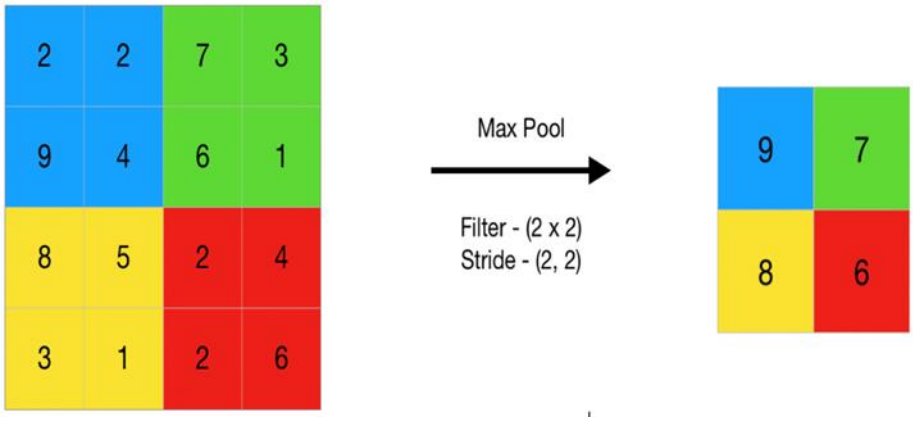


Figure 12 : Max Pooling .

-Le moyen pooling (Average Pooling) : calcule la moyenne des éléments présents dans la région de la carte des caractéristiques couverte par le filtre. Ainsi, alors que max pooling donne la fonctionnalité la plus importante dans un patch particulier de la carte des fonctionnalités, le pooling moyen donne la moyenne des fonctionnalités présentes dans un patch [24]. Figure 13 représenté Le moyen pooling.

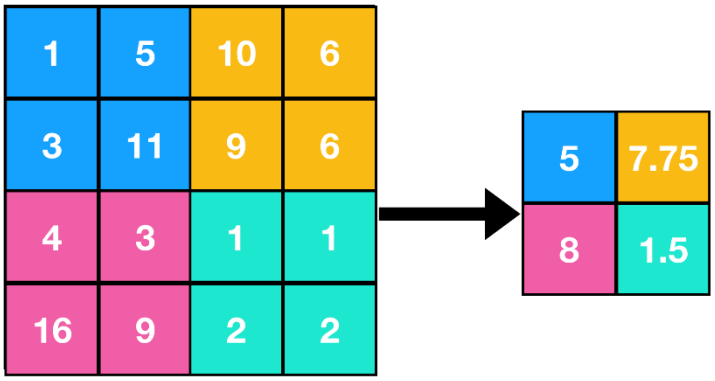


Figure 13 : Moyen pooling.

- globale pooling (Global pooling) : est une opération utilisée dans les réseaux de neurones convolutifs (CNN) pour réduire la dimensionnalité des caractéristiques extraites par les couches convolutives précédentes. Contrairement à la convolution ou à l'échantillonnage, qui réduisent la taille spatiale des caractéristiques, le global pooling réduit la dimensionnalité en agrégeant les valeurs sur l'ensemble de la carte des caractéristiques. Il peut être de différents types, tels que le "global max pooling" qui sélectionne la valeur maximale ou le "global average pooling" qui calcule la valeur moyenne. Cette opération permet de réduire le nombre de paramètres du modèle, tout en préservant les informations essentielles pour la tâche de classification ou de prédiction [25]. Figure 14 représenté globale pooling.

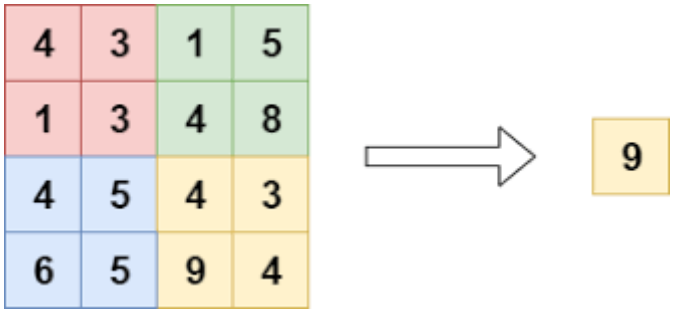


Figure 14 : Globale pooling.

2.3.3. Dropout :

Dropout est une technique permettant de réduire le surajustement lors de l'entraînement d'un modèle.

Le terme " Dropout " fait référence à la suppression de neurones dans une couche d'un modèle d'apprentissage en profondeur.

En effet, on désactive temporairement certains neurones du réseau, et toutes ses connexions entrantes et sortantes :

La sélection des neurones à désactiver est aléatoire. Une probabilité p est attribuée à tous les neurones qui déterminent leur activation.

Lorsque p = 0,1, chaque neurone a une chance sur 10 d'être désactivé.

A chaque époque, nous appliquons cette panne aléatoire. C'est-à-dire qu'à chaque passage (forward propagation) le modèle apprendra une configuration de différents neurones, qui s'activent et se désactivent de manière aléatoire [26]. Figure 15 représenté un application le Dropout sur Neural net.

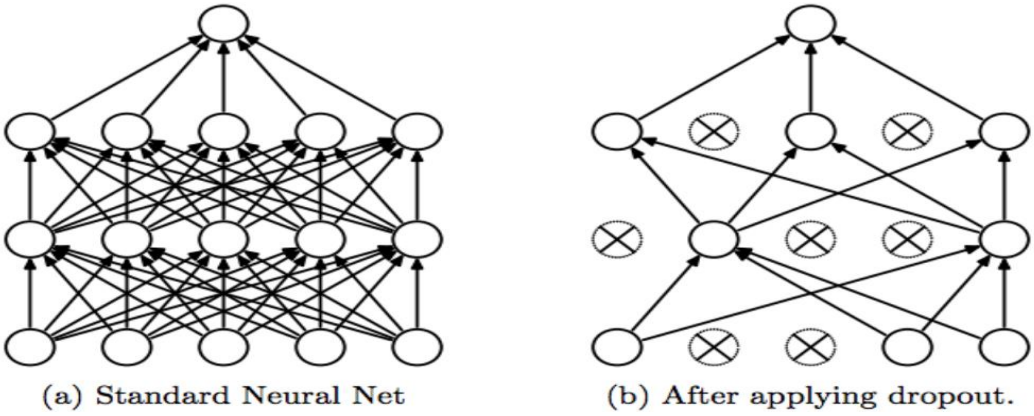


Figure 15 : Un application le Dropout sur Neural net.

2.3.4. Couche Flatten (Flatten layer) :

Une couche Flatten est une couche dans une communauté neuronale qui est utilisée pour transformer les enregistrements d'entrée multidimensionnels en vecteur 1 dimension. La cause de la couche aplatie est de prendre la sortie de la couche précédente, qui sera à l'intérieur de la forme d'une image 2D, d'un volume 3D, ou d'un tenseur de meilleures dimensions, et de la remodeler directement en un vecteur non marié qui peut être introduit directement dans une couche entièrement connectée (Fully Connected Layers).

La couche Flatten n'a plus de paramètres à apprendre, et sa caractéristique est honnêtement de remodeler le tenseur d'entrée directement en vecteur. Cependant, il occupe une position cruciale dans beaucoup d'architectures d'étude profonde, en particulier celles concernant les CNN, car il permet la version de système complexe, multidimensionnelle entrer des informations l'utilisation de couches entièrement connectées (Fully Connected Layers) [27]. Figure 16 représenté La couche Flatten.

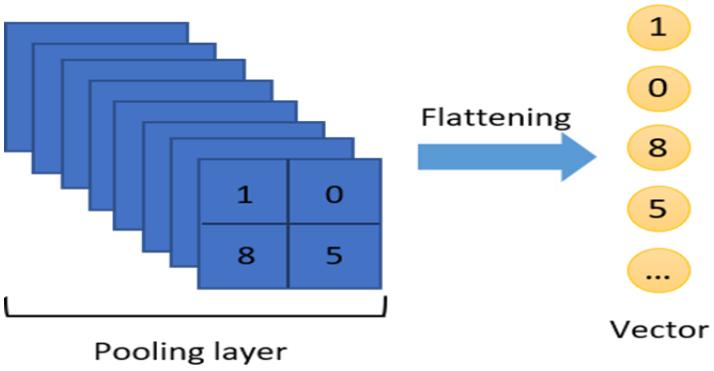


Figure 16 : Couche Flatten.

2.3.5. Couches denses (dense layer)

Les couches denses comprennent un ensemble de neurones qui pourraient être absolument liés à tous les neurones à l'intérieur de la couche précédente. Chaque neurone dans une couche dense entre de tous les neurones à l'intérieur de la couche précédente et produit une sortie basée p sur ses propres poids et biais.

La cause des couches denses est d'analyser des relations non linéaires compliquées entre les fonctions extraites par voie de moyens de couches à l'avance. En combinant ces fonctions, les couches denses peuvent produire des représentations de haut niveau qui pourraient être bénéfiques pour les tâches de catégorie ou de régression.

Dans l'ensemble, les couches denses jouent un rôle crucial dans les NCN par le biais de moyens de présentation d'une manière d'analyser les relations compliquées entre les fonctions extraites au moyen de moyens de couches à l'avance [27]. Figure 17 représenté La couche denses.

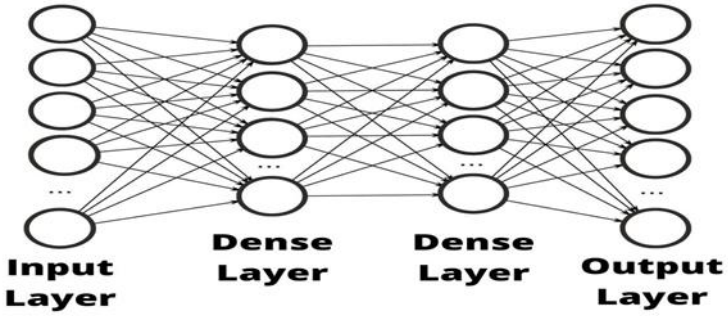


Figure 17 : Les couches denses .

2.4. L'apprentissage par transfert (Transfer Learning) :

L'apprentissage par transfert est une technique d'apprentissage automatique où un modèle pré-entraîné est utilisé comme point de départ pour une nouvelle tâche, plutôt que de former un modèle à partir de zéro.

L'idée derrière l'apprentissage par transfert est qu'un modèle pré-entraîné a déjà appris à reconnaître de nombreuses caractéristiques et tendances dans un grand ensemble de données, comme Image Net, et peut être utilisé comme point de départ pour une nouvelle tâche. En utilisant un modèle pré-entraîné, le processus d'apprentissage peut être accéléré et le modèle peut potentiellement atteindre une plus grande précision sur la nouvelle tâche avec moins de données et de temps de formation.

Il existe plusieurs façons d'effectuer l'apprentissage par transfert, y compris la mise au point, l'extraction de fonctionnalités et l'adaptation de domaines. La mise au point consiste à prendre un modèle pré-entraîné et à le former sur un nouvel ensemble de données, tout en maintenant certains des poids originaux fixes. L'extraction de fonctionnalités consiste à utiliser le modèle pré-entraîné comme extracteur de fonctionnalités fixes. L'adaptation de domaine consiste à adapter le modèle pré-entraîné à un nouveau domaine, tel qu'un ensemble différent de styles d'image ou de langages.

L'apprentissage par transfert est particulièrement utile dans l'apprentissage profond, en particulier dans la vision par ordinateur, le traitement du langage naturel et la reconnaissance vocale, où de grands modèles pré-entraîné sont disponibles. Il peut réduire considérablement le temps et les ressources nécessaires pour former un nouveau modèle à partir de zéro et améliorer la performance du modèle sur la nouvelle tâche [28].

2.5. Quelques concepts de L’apprentissage par transfert :

Il existe plusieurs architectures de transfert d'apprentissage selon le type de modèle pré-entraîné utilisé et la tâche cible à effectuer. Voici quelques exemples :

1) Fine-tuning : Cette méthode consiste à prendre un modèle pré-entraîné sur une tâche similaire et à le recycler sur la tâche cible en ajustant les poids de la dernière couche du réseau. Cela permet de bénéficier de la connaissance du modèle pré-entraîné tout en adaptant le modèle aux spécificités de la tâche cible.

2) Extraction de caractéristiques : cette approche consiste à utiliser des modèles pré-entraîné pour extraire des caractéristiques d'images, de texte ou de signaux audio, par exemple, et les utiliser pour former de nouveaux modèles sur la tâche cible.

3) Adaptation de domaine : cette approche consiste à utiliser un modèle pré-entraîné sur un domaine source pour améliorer les performances du modèle sur un domaine cible différent mais similaire. Cela peut être utile lorsque les données d'entraînement sont limitées dans le domaine cible.

4) Multi-Task Learning : cette approche implique la formation d'un modèle sur plusieurs tâches simultanément en partageant les représentations apprises sur différentes tâches. Cela améliore les performances du modèle sur toutes les tâches.

Il existe de nombreuses autres architectures de transfert d'apprentissage en fonction des spécificités de la tâche et du modèle pré-entraîné utilisé [28].

2.6. Certains modèles pour L’apprentissage par transfert :

Il existe plusieurs modèles pré-entraînés populaires pour le transfert d'apprentissage dans différents domaines de l'apprentissage automatique, qu’on représente ces quatre modèles :

2.6.1. VGG (Visual Geometry Group) :

Une architecture de réseau neural de convolution (CNN) avec 16 ou 19 couches qui a atteint une grande précision dans le concours Image Net. Il se compose seulement de 3x3 couches convolutives et 2x2 couches de regroupement [29]. Le modèle VVG16 sera choisi au sein de la famille VVG.

A) Modèle VGG16 :

VGG16 est un modèle de réseau neuronal convolutif (CNN) pré-entraîné pour la classification d'images, développé par le Visual Geometry Group de l'Université d'Oxford. Il a été publié en 2014 et est devenu un modèle populaire de transfert d'apprentissage.

Le modèle VGG16 se compose de 13 couches convolutives et de 3 couches entièrement connectées, avec un total de 138 millions de paramètres. Il prend en entrée une image de taille 224x224x3 et peut prédire les catégories d'objets à partir d'un ensemble de 1000 catégories dans l'ensemble de données Image Net.

Le nom "VGG16" fait référence à la structure du modèle, qui se compose de 16 couches (13 couches convolutives et 3 couches entièrement connectées), utilisant de petits filtres convolutifs (3x3) avec une foulée de 1 pixel et une fonction d'activation. Le modèle utilise également une couche max-clustering après chaque couche convolutive pour réduire la dimensionnalité spatiale de l'image. Le modèle VGG16 est souvent utilisé comme extracteur de caractéristiques pour l'apprentissage par transfert dans des tâches de reconnaissance d'images similaires, remplaçant les couches entièrement connectées par de nouvelles couches adaptées à des tâches spécifiques [30]. Figure 18 représenté Architecture de modèle VGG16.

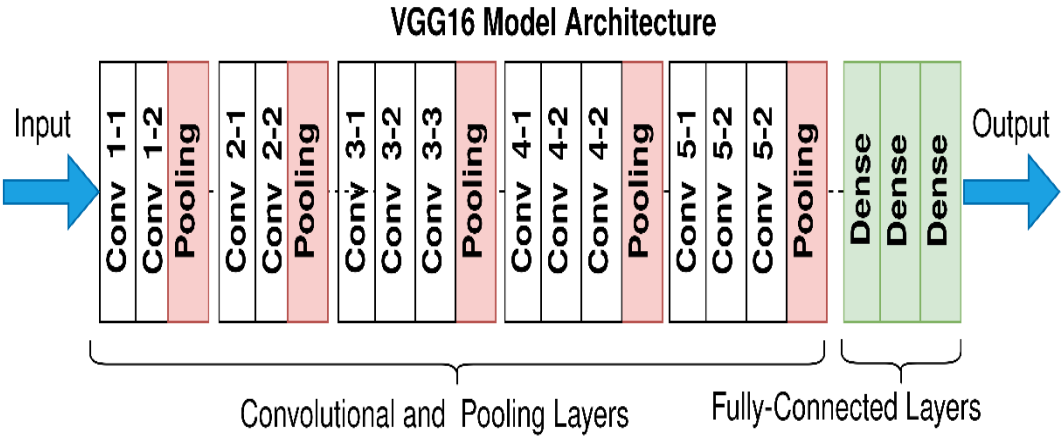


Figure 18 : Architecture de modèle VGG16.

2.6.2. Residual Network (ResNet) :

ResNet est une architecture de réseau neuronal à convolution profonde développée par Microsoft Research en 2015. Il est conçu pour résoudre le problème de la disparition des gradients dans les réseaux neuronaux très profonds.

ResNet inclut des connexions résiduelles, qui sont des raccourcis qui permettent au gradient de contourner certaines couches. Ces raccourcis permettent au réseau d'apprendre les mappages d'identité, ce qui peut améliorer les performances des réseaux très profonds.

ResNet est généralement formé à de grands ensembles de données, comme Image Net, et affiné sur des tâches spécifiques avec des ensembles de données plus petits. Il existe plusieurs variantes de ResNet, y compris ResNet-18, ResNet-34, ResNet-50, ResNet-101 et ResNet-152, qui diffèrent par leur profondeur et le nombre de couches.

ResNet a réalisé des performances de pointe sur plusieurs tâches de vision par ordinateur, notamment la classification des images, la détection des objets et la segmentation sémantique. Il a également été largement utilisé comme modèle de formation préalable pour le transfert d'apprentissage dans diverses applications [31]. Le modèle ResNet-50 sera choisi au sein de la famille ResNet.

A) Modèle ResNet-50 :

ResNet-50 est une architecture de réseau neuronal à convolution profonde qui a été introduite en 2015 par Kaiming He et al. Dans leur article "Deep Residual Learning for Image Recognition." Il se compose de 50 couches, et est basé sur le cadre d'apprentissage résiduel, qui aborde le problème de la disparition des gradients dans les réseaux neuronaux très profonds en introduisant des connexions résiduelles. Figure 19 représenté Architecture de modèle ResNet-50.

ResNet-50 a réalisé des performances de pointe sur plusieurs repères de classification d'images, y compris l'ensemble de données Image Net, qui comprend plus d'un million d'images réparties dans 1000 catégories. L'architecture se compose de plusieurs blocs de construction, dont chacun contient plusieurs couches convolutives et des connexions résiduelles. Les couches finales du réseau consistent en une couche globale moyenne de mise en commun, une couche entièrement connectée et une couche de sortie soft max.

ResNet-50 est souvent utilisé comme point de départ pour le transfert de l'apprentissage dans les tâches de vision par ordinateur, en raison de sa forte performance et de la disponibilité de poids pré-entraîné [31].

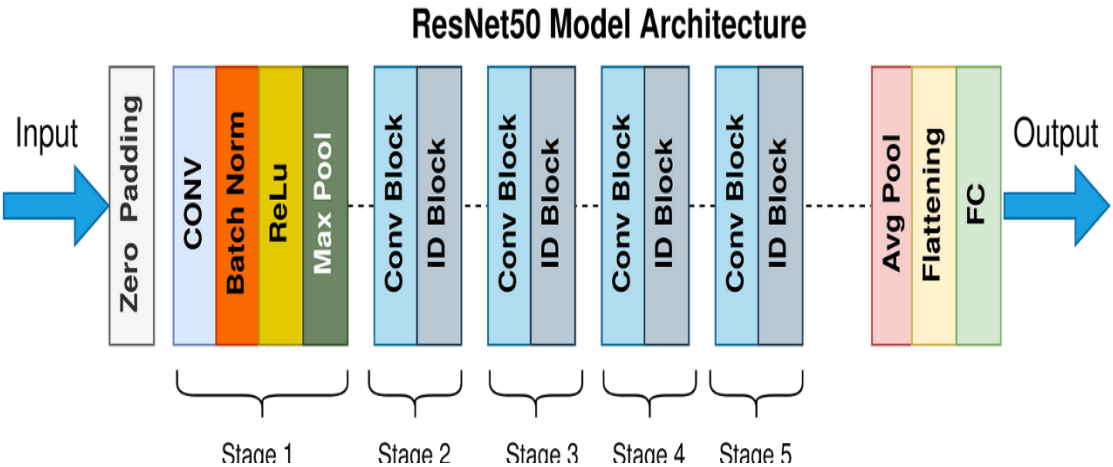


Figure 19 : Architecture de modèle ResNet-50 .

2.6.3. Dense Net :

Dense Net, abréviation de "Densely Connected Convolutional Networks", est un modèle de réseau de neurones convolutif (CNN) qui a été introduit en 2016 par Huang et al. Dense Net se distingue des autres architectures de CNN par sa structure dense de connexions entre les couches. Dans un réseau Dense Net, chaque couche est connectée directement à toutes les couches précédentes, créant ainsi une architecture dense. Cette connectivité dense permet de maximiser le flux d'informations et de favoriser le partage des caractéristiques à travers les couches du réseau.

La structure dense de Dense Net présente plusieurs avantages, notamment une meilleure utilisation des paramètres du modèle, une réduction de la redondance des caractéristiques, une meilleure propagation du gradient lors de l'apprentissage.Dense Net a démontré de bonnes performances dans diverses tâches de vision par ordinateur, telles que la classification d'images, la détection d'objets et la segmentation sémantique. [32]. Le modèle DenseNet-201sera choisi au sein de la famille Dense Net.

2.6.3.1. Dense Net-201 :

DenseNet-201 est une variante spécifique du modèle DenseNet, qui fait référence à une version spécifique avec 201 couches. DenseNet-201 a été proposé par Huang et al. En 2016 et est largement utilisé dans le domaine de la vision par ordinateur.

La principale caractéristique de DenseNet-201 est sa structure dense de connexions entre les couches. Dans ce modèle, chaque couche est connectée directement à toutes les couches précédentes, ce qui favorise un flux

d'informations dense et riche en fonctionnalités à travers le réseau. Cette connectivité dense permet au modèle de mieux exploiter les caractéristiques extraites à différentes échelles et niveaux d'abstraction, ce qui peut conduire à une représentation plus puissante des données.

DenseNet-201 se distingue également par son architecture profonde, avec un total de 201 couches. Cette profondeur accrue permet au modèle de capturer des motifs complexes et d'apprendre des représentations hautement discriminantes pour des tâches telles que la classification d'images, la détection d'objets et la segmentation sémantique. Figure 20 représenté architecture de modèle Dense Net.

Grâce à sa structure dense de connexions et à sa profondeur, DenseNet-201 a démontré de très bonnes performances dans des compétitions et des benchmarks de vision par ordinateur. Il est souvent utilisé comme une référence solide pour des tâches de reconnaissance d'images exigeantes, où des niveaux élevés de précision et de représentation sont nécessaires.[33]

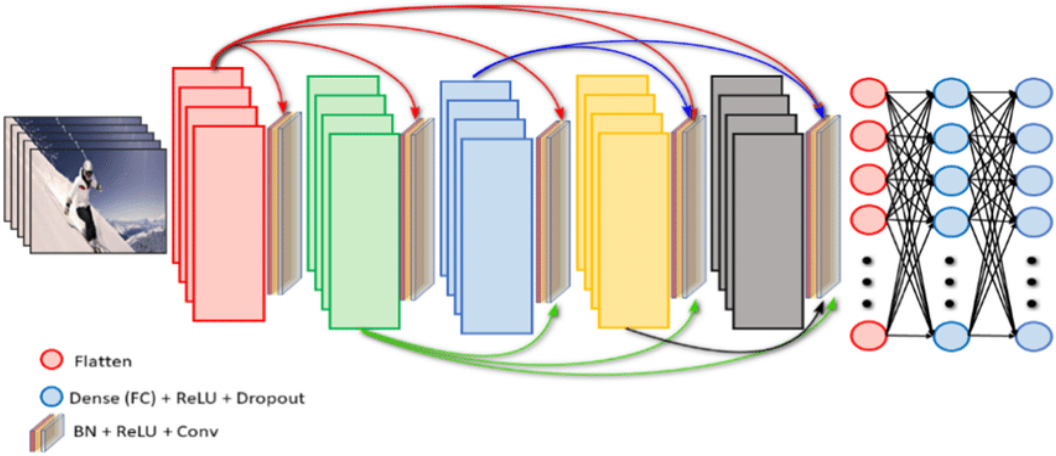


Figure 20 : Architecture de modèle Dense Net-201.

2.6.4. Inception :

Le terme "modèle Inception" est généralement utilisé pour faire référence à l'architecture Inception, qui est une architecture de réseau neuronal profond conçue pour des tâches de classification d'images.

L'architecture Inception est connue pour son utilisation novatrice des modules Inception, qui sont conçus pour capturer des caractéristiques à différentes échelles de manière efficace. Ces modules sont composés de plusieurs couches de convolution avec différentes tailles de filtre (1x1, 3x3, 5x5) et

d'opérations de pooling. En combinant ces différentes opérations en parallèle, le modèle peut capturer efficacement les informations locales et globales des images en entrée.

Le modèle Inception a connu plusieurs itérations, avec l'Inception v1 original suivi de l'Inception v2, l'Inception v3, et ainsi de suite. Chaque itération a introduit des améliorations et des optimisations à l'architecture, telles que les convolutions factorisées, la normalisation par lots et les classifieurs auxiliaires. Les modèles Inception ont atteint des performances de pointe sur divers benchmarks de classification d'images, notamment le défi de reconnaissance visuelle à grande échelle Image Net (ILSVRC). Ils ont également été largement adoptés et utilisés comme modèles pré-entraînés pour le transfert d'apprentissage dans diverses tâches de vision par ordinateur [34].

2.6.4.1. Inception v3 :

L'Inception v3 est une itération spécifique de l'architecture Inception, qui constitue une amélioration par rapport à ses versions précédentes.

L'Inception v3 présente plusieurs améliorations par rapport à l'Inception v2. Il utilise des convolutions de dimension réduite (3x3) avec des facteurs de réduction 1x1 avant les convolutions plus larges (5x5), ce qui permet de réduire le coût de calcul tout en préservant les performances du modèle. De plus, il intègre des blocs d'attention appelés "factorized reduction", qui permettent de réduire la dimensionnalité de l'espace des caractéristiques de manière efficace. Figure 21 représenté architecture de modèle Inception v3.

L'Inception v3 a été entraîné sur le jeu de données ImageNet et a atteint des performances remarquables dans la tâche de classification d'images. Il a également été utilisé comme modèle pré-entraîné dans des tâches de transfert d'apprentissage, où il a démontré sa capacité à extraire des caractéristiques pertinentes pour diverses tâches de vision par ordinateur, telles que la détection d'objets et la segmentation d'images.

L'Inception v3 est largement utilisé dans la communauté de l'apprentissage automatique en raison de sa combinaison de performances élevées et d'une architecture relativement légère. Il constitue une référence pour les architectures d'apprentissage profond et a ouvert la voie à d'autres variantes de l'architecture Inception [35].

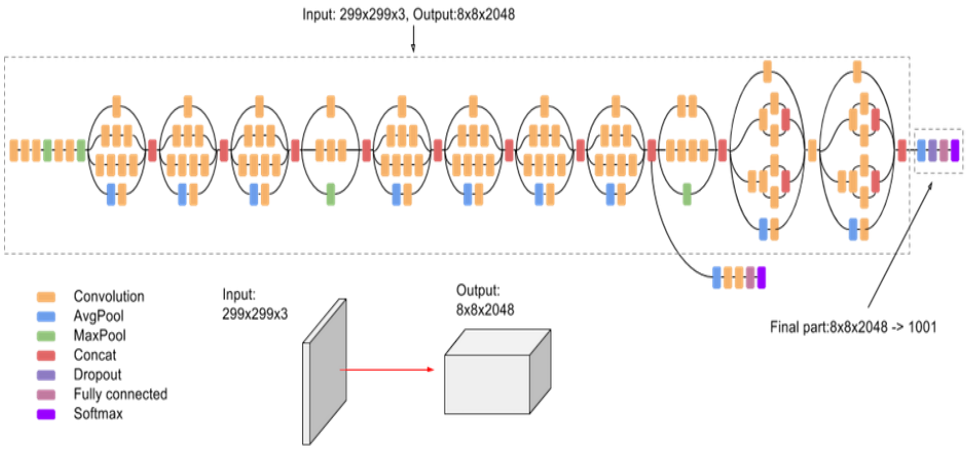


Figure 21: Architecture de modèle L'Inception v3.

2.7. Conclusion :

En conclusion, ImageNet, les couches CNN importantes, l'apprentissage par transfert et les modèles pré-formés sont tous des éléments essentiels d'apprentissage profond, en particulier dans le domaine de la vision par ordinateur. En utilisant ces techniques et ressources, il est possible de développer des modèles très précis et efficaces pour un large éventail de tâches de reconnaissance d'images.

Chapitre 3 : Méthodes de déploiement des modèles.

3.1. Introduction :

Le déploiement des applications d'apprentissage profond consiste à rendre les modèles entraînés accessibles et utilisables par les utilisateurs finaux ou d'autres applications. Ce processus est essentiel pour tirer parti des avantages de l'apprentissage profond dans des applications pratiques telles que la reconnaissance d'images et de la voix, le traitement du langage naturel et les systèmes autonomes.

Il existe plusieurs méthodes pour déployer des applications d'apprentissage profond, telles que les approches traditionnelles basées sur des serveurs, la conteneurisation et les solutions basées sur le cloud. Chaque approche présente des avantages et des inconvénients, et peut être plus adaptée à certains cas d'utilisation en fonction de facteurs tels que l'évolutivité, la sécurité et les ressources requises.

Le déploiement efficace des modèles d'apprentissage profond est essentiel pour la réussite de la mise en œuvre des applications d'IA dans divers domaines, offrant de nouvelles opportunités d'automatisation, de prédiction et de prise de décision.

3.2. Méthodes de déploiement :

Le déploiement de l'apprentissage en profond consiste à mettre en production un modèle d'apprentissage automatique préalablement entraîné. Cela si implique de rendre ce modèle accessible aux utilisateurs finaux afin qu'ils puissent effectuer des prédictions en temps réel sur de nouveaux ensembles de données. Le déploiement peut se faire sur des serveurs locaux, des serveurs cloud ou même des appareils mobiles, en fonction de la demande et des ressources disponibles. Le déploiement de modèles d'apprentissage en profond est une étape essentielle pour rendre les applications d'intelligence artificielle fonctionnelles et utiles dans le monde réel [36]. Voici quelques modelés de déploiement :

3.2.1. Flask :

Flask est un framework web léger en Python utilisé couramment pour le déploiement de modèles d'apprentissage en profond en production. Il simplifie le déploiement de modèles d'apprentissage en profond en offrant une architecture flexible et modulaire pour la création d'APIs RESTful. Figure 22 représente le logo de Flask. Pour déployer un modèle d'apprentissage en profond avec Flask,

Il est nécessaire de créer une API qui expose les fonctionnalités du modèle. Flask fournit des outils permettant de gérer les requêtes HTTP entrantes et sortantes, ainsi que d'assurer la communication avec le modèle.



Figure 22: Le logo de Flask .

Voici les étapes principales pour déployer un modèle de Deep Learning avec Flask :

1. **Charger le modèle** : importer le modèle pré-entraîné dans l'application Flask en utilisant une bibliothèque telle que TensorFlow ou PyTorch.
2. **Prétraitement** : effectuer le prétraitement des données d'entrée pour les adapter au format attendu par le modèle.
3. **API** : créer une API RESTful pour exposer les fonctionnalités du modèle en utilisant Flask.
4. **Déploiement** : déployer l'application Flask sur un serveur web pour qu'elle soit accessible en ligne. Une fois le modèle déployé avec Flask, il peut être utilisé pour effectuer des prédictions en temps réel en recevant des requêtes HTTP et en renvoyant des réponses.

Cela permet aux utilisateurs d'interagir avec le modèle via une interface web intuitive et conviviale.[45].

3.2.2. Django :

Django est un Framework Web Python populaire pour le déploiement de modèles d'apprentissage en profondeur. Il fournit des outils pour développer des applications Web robustes et évolutives, ce qui le rend idéal pour déployer des modèles d'apprentissage en profond dans des applications en ligne [38]. Figure 23 représenté Le logo de Django. Voici comment Django facilite le déploiement de modèles d'apprentissage en profond :

1) **Gestion de l'interface utilisateur** : Django fournit des outils pour créer des pages Web et des formulaires qui facilitent l'interaction entre les utilisateurs et les modèles.
2) **Facilité d'intégration** : Django peut être facilement intégré à différents services Web pour gérer les demandes des clients. Les modèles

peuvent être invoqués à partir de ces services Web en réponse aux demandes des utilisateurs.

3) **Gestion des ressources** : Django permet une gestion simple des fichiers statiques, des images, des fichiers CSS, du JavaScript et d'autres ressources, ce qui permet de présenter plus facilement les résultats du modèle aux utilisateurs.
4) **Sécurité** : Django fournit des fonctionnalités de sécurité intégrées, telles que la prévention des attaques CSRF (Cross-Site Request Forgery) et XSS (Cross-Site Scripting), ce qui aide à protéger l'application contre les attaques malveillantes.
5) **Scalabilité** : Django est conçu pour être facilement évolutif, ce qui permet aux développeurs d'ajouter de nouvelles fonctionnalités et de gérer une grande quantité de trafic Web sans compromettre la qualité de l'application.



Figure 23 : Logo de Django.

3.2.3. Streamlit :

Streamlit est un Framework open-source conçu pour faciliter la création et le déploiement rapide d'applications de données. Il permet également de déployer facilement des modèles d'apprentissage en profond en utilisant des scripts Python simples et intuitifs.Figure 24 représenté Le logo de Streamlit.

Avec Streamlit, les développeurs peuvent créer rapidement une interface utilisateur pour leur modèle de d'apprentissage en profond en quelques lignes de code. Streamlit permet également d'afficher des graphiques et des visualisations en temps réel et de les mettre à jour automatiquement lorsque les utilisateurs interagissent avec l'application.

En utilisant Streamlit, il est facile de déployer des modèles d'apprentissage en profond sur des serveurs cloud tels que Google Cloud Platform, Amazon Web Services, etc. En outre, Streamlit prend également en charge le déploiement de modèles d'apprentissage en profond sous forme d'applications

Web autonomes, ce qui permet aux utilisateurs d'accéder facilement à leur modèle depuis n'importe quel navigateur web .



Figure 24: Logo de Streamlit .

3.2.4. Docker :

Docker facilite le déploiement d'un modèle de Deep Learning en permettant d'isoler l'environnement de développement et de production. Il s'agit d'une plateforme logicielle permettant de créer, déployer et exécuter des applications dans des conteneurs légers et portables.Figure 25 représenté Le logo de Docker.

En utilisant Docker, il est possible de créer un conteneur contenant l'ensemble des dépendances requises pour exécuter un modèle d'apprentissage en profond, comme les bibliothèques Python, les Framework d'apprentissage en profond et les pilotes GPU. Ce conteneur peut être ensuite déployé sur n'importe quelle plateforme compatible avec Docker, qu'il s'agisse d'un serveur distant, d'un cluster Kubernetes ou d'un environnement de cloud computing.

Docker permet ainsi de simplifier le processus de déploiement, en garantissant que l'environnement d'exécution est cohérent avec celui de développement, et en assurant la portabilité et la reproductibilité du modèle d'apprentissage en profond [39].

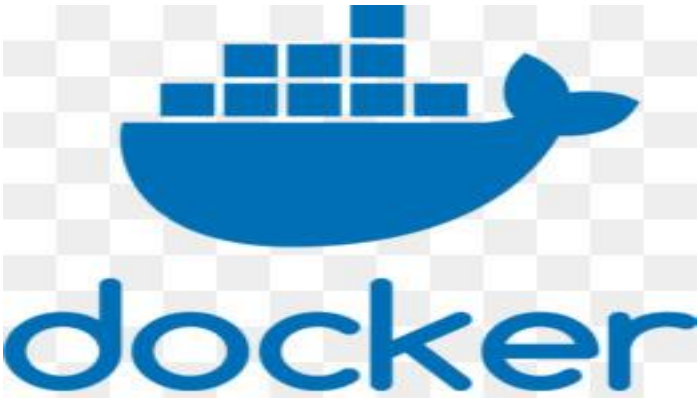


Figure 25: Logo de Docker .

3.2.5. Gradio :

Gradio est une bibliothèque open-source qui permet de créer rapidement et facilement des interfaces utilisateur pour les modèles d'apprentissage en profond. Elle permet de déployer des modèles de Deep Learning à l'aide d'une interface utilisateur simple et intuitive sans avoir besoin de connaissances en programmation web.Figure 26 représenté Le logo de Gradio

Grâce à Gradio, les utilisateurs peuvent facilement interagir avec les modèles en entrant des données dans des champs de saisie, en téléchargeant des fichiers ou en utilisant des caméras pour des applications de traitement d'images et de vidéos en temps réel.

En termes de déploiement, Gradio facilite grandement la tâche en fournissant des outils pour convertir des modèles d'apprentissage en profond en une interface utilisateur interactive et héberger le tout sur le cloud, sans avoir besoin de connaissances en programmation web ou en déploiement de modèles d'apprentissage en profond. Cela permet aux développeurs de se concentrer sur la création de modèles de haute qualité, tout en simplifiant considérablement le processus de déploiement pour les utilisateurs finaux [40].



Figure 26: Logo de Gradio .

3.2.6. Tensor Flow lite :

Tensor Flow Lite est une version allégée de la bibliothèque Tensor Flow, spécialement conçue pour le déploiement de modèles d'apprentissage en profond sur des appareils mobiles et embarqués tels que les smartphones, les tablettes, les montres intelligentes, les appareils photo, les robots, etc. Figure 25 représenté Le logo de Tensor Flow Lite.

Tensor Flow Lite offre plusieurs avantages pour le déploiement de modèles d'apprentissage en profond, notamment une faible consommation d'énergie, une utilisation efficace des ressources de l'appareil, une latence réduite et une compatibilité avec divers processeurs mobiles.

Pour faciliter le déploiement de modèles d'apprentissage en profond avec Tensor Flow Lite, il existe plusieurs outils et techniques, tels que :

- Tensor Flow Lite Converter : un outil qui permet de convertir un modèle Tensor Flow en modèle Tensor Flow Lite, optimisé pour une utilisation sur des appareils mobiles et embarqués.

- Tensor Flow Lite Interpréteur : une bibliothèque qui permet d'exécuter des modèles Tensor Flow Lite sur des appareils mobiles et embarqués.
- Tensor Flow Lite Model Maker : une bibliothèque qui permet de créer des modèles d'apprentissage en profond personnalisés pour des tâches spécifiques, optimisés pour une utilisation sur des appareils mobiles et embarqués [41].



Figure 27 : Logo de Tensor Flow lite .

3.3. Conclusion :

En conclusion, le choix de la méthode de déploiement dépend du cas d'utilisation spécifique et des exigences de l'application. En utilisant ces diverses méthodes de déploiement, les développeurs peuvent créer et déployer des modèles d'apprentissage profond sur une variété de plateformes et d'appareils, rendant les avantages de l'IA accessibles à un plus large éventail d'utilisateurs .

Chapitre 4 : Expérimentation

4.1. Introduction :

Dans ce chapitre, nous nous intéressons à la classification des maladies des plantes en utilisant l'apprentissage par transfert. Notre étude se focalise spécifiquement sur l'évaluation des performances de différents modèles d'apprentissage par transfert en utilisant dataset «PlantVillage ».

Nous avons choisi d'utiliser les modèles VGG16, ResNet-50, DenseNet-201 et Inception-v3 pour mener nos expériences.

Dans les sections suivantes de ce chapitre, nous présenterons les résultats de nos expériences en évaluant les performances de chaque modèle sur dataset "PlantVillage". Nous analyserons les performances en termes de Accrecy, de test et d'autres métriques pertinentes pour évaluer l'efficacité de chaque modèle dans la classification précise des maladies des plantes.

4.2. Agriculture intelligente :

L'agriculture intelligente, également connue sous le nom d'agriculture de précision, est un domaine en pleine expansion qui vise à améliorer l'efficacité et la productivité agricole en utilisant des technologies de pointe telles que l'intelligence artificielle. Cette relation entre l'agriculture et l'intelligence artificielle permet d'apporter des solutions innovantes aux défis auxquels l'agriculture est confrontée, tant à l'échelle nationale qu'internationale.

Dans de nombreux pays, y compris en dehors de l'Algérie, l'agriculture intelligente est utilisée pour optimiser la gestion des cultures, la surveillance des conditions météorologiques, la gestion de l'eau et des ressources, ainsi que la détection précoce des maladies des plantes. Les systèmes d'intelligence artificielle permettent d'analyser de grandes quantités de données agricoles, telles que les données climatiques, les données du sol et les données des cultures, afin de prendre des décisions éclairées et d'optimiser les pratiques agricoles.

En Algérie, l'agriculture intelligente commence également à gagner en importance. Les agriculteurs et les organisations agricoles commencent à adopter des technologies telles que les capteurs connectés, l'analyse des données et les systèmes de surveillance pour améliorer leur productivité et leur rentabilité. L'utilisation de l'intelligence artificielle dans l'agriculture algérienne peut contribuer à une utilisation plus efficace des ressources, à une meilleure gestion des cultures et à une réduction des pertes.

Des exemples concrets d'application de l'agriculture intelligente en Algérie pourraient inclure l'utilisation de drones pour la surveillance des cultures, l'analyse des données satellitaires pour la prévision des rendements, l'utilisation de modèles d'apprentissage automatique pour la détection précoce

des maladies des plantes, et la mise en place de systèmes d'irrigation intelligents basés sur des capteurs.

En résumé, l'agriculture intelligente, grâce à l'utilisation de l'intelligence artificielle, ouvre de nouvelles perspectives pour l'agriculture en Algérie et à travers le monde. Elle permet d'optimiser les ressources, de prendre des décisions basées sur des données et d'améliorer la productivité agricole de manière durable.

4.3. Quelques exemples concrets d'agriculture intelligente :

Voici des projets réels dans le domaine de l'agriculture intelligente, en particulier dans le domaine de l'intelligence artificielle et de la surveillance des plantes, qui sont présents sur le terrain :

1- **Projet Plantix :** Une application mobile basée sur l'intelligence artificielle pour diagnostiquer les maladies des plantes et fournir des recommandations de traitement appropriées**.**

2- **Projet Phytech :** Une entreprise spécialisée dans les solutions d'irrigation intelligente qui utilise des capteurs de terrain, des algorithmes d'apprentissage automatique et des modèles de plantes pour optimiser l'irrigation en fonction des besoins spécifiques de chaque culture.

3- **Projet Blue River Technology :** Développement de robots agricoles autonomes utilisant des caméras et des algorithmes de vision par ordinateur pour détecter les mauvaises herbes et appliquer sélectivement des herbicides, réduisant ainsi l'utilisation de produits chimiques et minimisant les dommages aux cultures.

Ces projets témoignent de l'application pratique de l'intelligence artificielle et de la surveillance des plantes dans l'agriculture intelligente. Ils utilisent des techniques avancées telles que le traitement des données, l'apprentissage automatique et la vision par ordinateur pour optimiser la gestion des cultures, réduire les coûts, améliorer la productivité et minimiser l'impact sur l'environnement.

4.4. Description du Dataset :

PlantVillage [42] est un ensemble de données disponible publiquement, contenant différentes catégories de maladies des plantes. Cet ensemble de données comprend 38 classes avec 54 305 images. Pour notre analyse expérimentale, nous avons divisé l'ensemble de données en échantillons d'entraînement, d'échantillons de test et d'échantillons de validation. Les

modèles pré-entraînés ont été entraînés avec 80% de l'ensemble de données PlantVillage, tandis que 20% ont été utilisés pour la validation et les tests. De plus, le nombre total d'échantillons disponibles pour les classes de plantes était de 54 305, dont 43 955 échantillons ont été utilisés pour l'entraînement, 4 902 pour la validation et 5 488 pour les tests. Tous ces ensembles d'entraînement, de test et de validation comprennent les 38 classes des différentes maladies des plantes. Les détails de la répartition de l'ensemble de données sont présentés dans le Tableau 1.

Type de plante	Catégories de maladies	Nombre Total des échantillons	Nombre DE Training Échantillons	NomberTest Échantillons	Nombre Validation Échantillons
Pomme	Apple_scab	573	510	63	57
	Apple_black_rot	565	502	63	56
	Apple_cedar_apple_rust	250	222	28	25
	Apple_healthy	1497	1332	165	148
Myrtille	Blueberry_healthy	1366	1218	151	136
Cerise	Cherry_powdery_mildew	957	851	106	95
	Cherry_healthy	777	691	86	77
Maïs	Corn_gray_leaf_spot	466	414	52	47
	Corn_common_rust	1084	964	120	108
	Corn_northern_leaf_blight	896	797	99	89
	Corn_healthy	1057	940	117	105
Raisin	Grape_black_rot	1073	955	118	107
	Grape_black_measles	1258	1119	139	125
	Grape_leaf_blight	979	871	108	97
	Grape_healthy	385	342	43	38

Orange	Orange_haunglongbing	5011	4460	551	496
Pêche	Peach_bacterial_spot	2090	1860	230	207
	Peach_healthy	327	291	36	33
Poivre	Pepper bell_bacterial_spot	997	807	100	90
	Pepper Bell_healthy	1478	1197	148	133
Pomme de terre	Potato_early_blight	1000	810	100	90
	Potato_healthy	1000	810	100	90
	Potato_late_blight	152	122	16	14
Framboise	Raspberry_healthy	664	299	38	34
Soja	Soybean_healthy	5295	4122	509	459
Squash	Squash_powdery_mildew	1669	1485	184	166
Fraise	Strawberry_healthy	1009	898	111	100
	Strawberry_leaf_scorch	415	369	46	41
Tomate	Tomato_bacterial_spot	2127	1722	213	192
	Tomato_early_blight	1000	810	100	90
	Tomato_healthy	1591	1546	191	172
	Tomato_late_blight	1909	770	96	86
	Tomato_leaf_mold	952	1433	178	160
	Tomato_septoria_leaf_spot	1771	1357	168	151
	Tomato_spider_mites_t Wo	1676	1136	141	127
	Tomato_target_spot	1404	4338	536	483
	Tomato_mosaic_virus	373	301	38	34
		3209	1287	160	144

	Tomato_yel-low_leaf_curl_virus				
Total	**----**	**54,305 43**	**43,955**	**5448**	**4902**

Tableau 1 : partager de dataset “PlantVillage”(test,validation).

4.5. Environnement et outils de développement :

Nous avons fait notre l’implémentation des modèles en utilisant python via le Google Colab .

4.5.1. Google Co-laboratory :

Colaboratoire, ou « Colab », est un produit de Google Research qui est Environnement interactif basé sur le Web qui vous permet d’écrire et d’exécuter du code Python sans configuration requise, accès gratuit aux GPU et TPU, et partage facile. Pour profiter de Google Colab, vous devez avoir un compte gmail afin de créer un ordinateur portable Colab où vous pouvez écrire et exécuter du code [43].

Google Colab inclut les cadres et les modules Python nécessaires Colab permet de créer, entraîner et évaluer des modèles d'apprentissage machine et profond. Il est également possible d'installer de nouveaux modules en utilisant la commande "pip install". Les données et les notebooks peuvent être importés depuis le système de fichiers local, Google Drive et Kaggle. De plus, les notebooks Colab ainsi que d'autres types de données peuvent être stockés à l'aide des services de stockage en nuage Google.

- **Durée de la session :**

Google colab offre 12 heures d’exécution continue pour une session.

- **La RAM disponible :**

L’espace disponible dans la mémoire d’accès aléatoire est de 12 Go, sans possibilité d’augmentation.

- **Espace de stockage disponible :**

En plus de Google Drive, l'environnement de Google Colab permet l'utilisation de plus de 30 Go d'espace de stockage pendant une session. Toutefois, il est important de noter que tous les modules Python et fichiers stockés sur cet espace seront supprimés une fois la session terminée.

- **GPU disponibles :**

Sur Colab, vous avez le choix d'exécuter sur un CPU, un GPU ou un TPU. Pour notre travail pratique de formation de modèles, nous avons opté pour les GPU, car ils sont faciles à utiliser et accélèrent l'exécution de modèles sophistiqués tels que BERT.

4.5.2. Modules Python utilisés :

Python, un langage de programmation interprété orienté objet de haut niveau, dispose d'une sémantique dynamique. Il offre des structures de données de haut niveau efficaces et une technique de programmation orientée objet simple mais efficace. Grâce à sa belle syntaxe et à sa frappe dynamique, ainsi qu'à sa nature interprétée, Python est un excellent choix pour le développement rapide d'applications et le Scripting sur diverses plates-formes [44].

- **Numpy :**

Est une bibliothèque fondamentale pour les calculs scientifiques en Python. En science des calculs, l'algèbre linéaire est couramment utilisée pour transformer les vecteurs en matrices, normaliser les données, effectuer des partitionnements de données et effectuer d'autres opérations mathématiques.

- **Tensor Flow :**

Est un Framework open source créée par des chercheurs de chez Google pour faciliter l'exécution de tâches liées à l'apprentissage automatique, l'apprentissage profond et d'autres travaux d'analyse prédictive et statistique [58]. Dans notre cas, nous avons utilisé la version 2.12.0 de Tenso rFlow.

- **Keras :**

Est une interface de programmation d'application (API) de réseaux neuronaux de premier plan. Elle est écrite en Python et est compatible avec plusieurs moteurs de calcul de réseaux neuronaux [59]. TensorFlow a choisi Keras comme API de haut niveau pour TensorFlow 2.12.0.

- **Matplotlib :**

Est une bibliothèque de visualisation en Python. Elle fournit un ensemble de commandes qui permettent de créer des graphiques et des visualisations de manière aisée. Ces commandes sont similaires à celles utilisées dans MATLAB, telles que la création d'une figure, la création d'une zone de tracé dans une figure, etc.

- **Sklearn :**

Il sert à évaluer et calculer la matrice de confusion, la précision, le rappel, la sensibilité, la spécificité et le score F1, permettant ainsi de déterminer la précision et le rappel.

4.6. Modèles pré-entraînées utiliser :

Les modèles pré-entraînées sont choisis en fonction de leur applicabilité pour la tâche de classification des maladies des plantes. Les détails de l'architecture du modèle sont donnés dans le tableau 3. Quatre modèles populaires, à savoir VGG-16, Inception V3, ResNet-50 et DenseNet-201, ont gagné beaucoup d'attention pour leurs capacités remarquables dans diverses tâches de vision par ordinateur. Chacun de ces modèles à ses caractéristiques architecturales uniques, y compris le nombre de couches, les couches pooling, les couches denses, les tailles des filtres, les strides et les paramètres d'entraînement.

Les modèles	**VGG-16**	**Inception V3**	**ResNet-50**	**DenseNet-201**
Total des couches	16	48	50	201
Max pool	5	3	1	4
Couches denses	3	-	3	-
Drop-out	2	-	2	-
Couches de Flatten	1	-	1	-
Tailles des filtres	3×3	3×3 ,1×1,5×5	3×3	3×3 ,1×1
Strides	2×2	2×2	2×2	2×2
Paramètres de trainable	134,268,544	23,851,784	25,636,712	20,242,984

Tableau 2 : L'architecture des Modèles pré-entraînée.

4.7. Les hyperparamètres des modèles pré-entraîné :

Le modèle présente des avantages significatifs par rapport à la construction de modèles à partir de zéro. Il offre une capacité d'apprentissage plus rapide, car il peut capitaliser sur les connaissances préalables acquises lors de l'entraînement sur de vastes ensembles de données. De plus, ce modèle permet de geler certaines couches, empêchant ainsi leurs poids d'être modifiés, tandis que les dernières couches sont spécifiquement adaptées pour une classification plus précise. Cette approche permet d'optimiser les performances du modèle en utilisant efficacement les caractéristiques apprises par les couches pré-entraînées.

Dans notre étude, nous avons également effectué Différent test sur valeur des hyperparamètres pour différents modèles pré-entraînés. Cela implique d'ajuster les valeurs des hyperparamètres tels que le taux d'apprentissage, les classes sorties, le nombre des Epoch et batch size, afin de trouver les configurations optimales pour chaque modèle. Cette étape permet de tirer pleinement parti des avantages du modèle en optimisant ses performances lors de l'entraînement. Grâce à cette approche, nous sommes en mesure d'obtenir des résultats Intéressants tout en réduisant le temps nécessaire pour entraîner le modèle. Le tableau 2 exprime les hyperparamètres de les modèles pré-entraîné.

Les hyperparamètres	**Les valeurs**
Epochs	50
Le taux d'apprentissage	0,0001
Batch Size	32
Les classes de sorties	38

Tableau 3 : Les hyperparameters de les modèles Pre-Trained.

Les modèles ont été optimisés en utilisant la descente en gradient stochastique. Le taux d'apprentissage initiaux des modèles DenseNet-201, ResNet-50, VGG-16 et Inception V3 ont été fixés à 0,0001 et le Batch Size est 32 et Les classes de sorties est 38 classes. On a initialisé les époques avec 50 et on a utilisé Les fonctions « callbacks » définie dans Keras sont des fonctions, qui sont utilisé durent spécifiques pendant l'entraînement d'un modèle. Elles peuvent être utilisées pour effectuer des actions telles que l'enregistrement de modèles, la visualisation de métriques d'entraînement en temps réel et l'arrêt prématuré de l'entraînement si certaines conditions ne sont pas remplies. Les paramètres couramment utilisés pour la fonction « callbacks » dans Keras sont :

1. **"ModelCheckpoint" :** Cette fonction de rappel permet de sauvegarder le meilleur modèle pendant l'entraînement en fonction d'une métrique spécifique. Elle permet de suivre la performance du modèle et de sauvegarder les poids du modèle à chaque amélioration significative.
2. **"EarlyStopping" : Cette** fonction de rappel permet d'arrêter l'entraînement du modèle prématurément si aucune amélioration significative n'est observée pendant un certain nombre d'époques. Elle aide à éviter le surapprentissage et à économiser du temps de calcul.

Ces fonctions de rappel peuvent être utilisées lors de la compilation du modèle en tant que paramètres callbacks pour améliorer le processus d'entraînement et obtenir de meilleurs résultats. Figure 28 exprime Les paramètres couramment utilisés pour la fonction « callbacks ».



Figure 28: Fonction « Call Backs ».

4.7.1. Détails du réglage VGG-16 :

Les dimensions de l'image d'entrée pour le network sont 224× 224× 3, et il a 64 canaux dans les deux premières couches avec une taille de filtre de 3×3 et stride de 2. Les deux couches suivantes dans le VGG-16 ont 256 canaux avec 3× 3 filtres ; suivi d'une couche de max-pooling avec stride de 2. Après la couche de pooling, il y a deux couches de convolution avec 256 canaux avec une taille de filtre de 3 ×3. Après les deux couches de convolution, il y a deux séries de trois couches de convolution, avec une couche de pooling, avec 3 ×3 filtres. Le network comprend une couche Flatten, cinq couches de max pool et deux couches denses.

4.7.2. Détails du réglage de Inception V3 :

L'architecture Inception V3 utilise des images d'entrée de dimensions 224× 224× 3. Elle se compose de plusieurs modules Inception, chacun comportant des filtres de différentes tailles (1×1, 3×3, 5×5) pour capturer des motifs à différentes échelles. Ces modules sont empilés les uns sur les autres, suivis de couches de max-pooling. Le réseau comprend également des couches de classification et des couches entièrement connectées pour la prédiction finale.

4.7.3. Détails du réglage de ResNet-50 :

ResNet-50 utilise des images d'entrée de dimensions 224× 224× 3. Il se compose de plusieurs blocs résiduels, où chaque bloc contient plusieurs couches de convolution avec des filtres de taille 3×3. La particularité de

ResNet-50 réside dans l'utilisation des connexions résiduelles, qui permettent aux informations de contourner les couches et de passer directement à des couches plus profondes. Le réseau se termine également par des couches de classification et des couches entièrement connectées.

4.7.4. Détails du réglage de DenseNet-201 :

Le modèle DenseNet-201 utilise des images d'entrée de dimensions 224× 224× 3. Il se compose de plusieurs blocs Dense, où chaque bloc est connecté à tous les blocs précédents. Chaque bloc dense contient des couches de convolution avec des filtres de taille 3×3 et des connexions directes (skip connections) qui fusionnent les caractéristiques des blocs précédents. Le réseau se termine par des couches de classification et des couches entièrement connectées.

4.8. Modification apportées aux modèles utiliser :

On a utilisé les modèle pré-entraînées que nous avons présente précédent afin de faire extraction (tous les couches en garde leur poids initial) en suit nous avons les couches :

- Une Couche Flatten.
- Une couche Dense avec 512 neurones et une fonction d'activation « relu».
- Une couche Dense avec 128 neurones et une fonction d'activation « relu ».
- Une couche Dense avec le nombre des classes et une fonction d'activation « softmax ».

Figure 29 montre les couches ajoutées.

```
x = Flatten()(vgg.output)
x = Dense(512, activation='relu')(x)
x = Dense(128, activation='relu')(x)
prediction = Dense(n_classes, activation='softmax')(x)

model = Model(inputs=M.input, outputs=prediction)
```

Figure 29 : Extraction des couches ajoute aux modèles.

4.9. Prétraitement et augmentation des données :

PlantVillage se compose de 38 classes comprenant 26 maladies et 14 espèces de cultures différentes. Dans le cadre de nos expériences, nous avons utilisé les images en couleur de l'ensemble de données PlantVillage, Les images ont été redimensionnées en 256 x 256 pixels, une taille standardisée, étant donné que nous avons utilisé différents modèles de réseaux pré-entraînés qui nécessitent des tailles d'entrée différentes. Pour les modèles VGG-16, DenseNet-201, Inception V3 et ResNet-50, la taille d'entrée est de 224 x 224 x 3 (hauteur, largeur et canaux de couleur).

Bien que le data set soit volumineux, avec environ 54 000 images représentant différentes maladies des cultures, les images reflètent celles réellement capturées par les agriculteurs en utilisant différentes techniques d'acquisition d'images telles que les capteurs Kinect, les caméras haute définition et les smartphones. Cependant, un ensemble de données de cette taille peut poser des problèmes de capacité de stockage. Afin de surmonter cette limitation, des techniques de régularisation excessive, telles que l'augmentation des données après le prétraitement, ont été utilisées. Les processus d'augmentation des données appliqués aux images prétraitées comprenaient la rotation horaire et antihoraire, le retournement horizontal et vertical, le zoom et le redimensionnement. Les images n'ont pas été dupliquées, mais augmentées pendant le processus d'entraînement, de sorte que des copies physiques des images augmentées n'ont pas été stockées, mais ont été utilisées temporairement lors du processus. Cette technique d'augmentation des données permet non seulement d'éviter la surcharge et la perte de modèle, mais elle renforce également la robustesse du modèle, de sorte que lorsqu'il est utilisé pour classer de véritables images de maladies végétales, il peut le faire avec une meilleure précision.

4.10. Résultats et Discussion :

4.10.1. Pour le model VGG16 :

On a observé que la courbe d’accuracy (couleur bleue) et la courbe validation_ accuracy (couleur orange) présentent une tendance à la hausse. À l’époque initiale (époque 0), l’accuracy commence à 0,68, tandis que validation_ accuracy commence à 0,83. À mesure que le training progresse, les deux courbes convergent progressivement, atteignant une valeur de 0,9 à l’époque 2. À partir de ce point, l’accuracy et validation_ accuracy continuent d’augmenter jusqu’à atteindre leurs sommets respectifs à l’époque finale (époque 10), où l’accuracy est de 0,98 et validation_ accuracy est de 0,91. Ces résultats indiquent que la performance du modèle s’améliore régulièrement au cours de la formation comme en témoignent les valeurs croissantes d’accuracy

des ensembles de data de training et de validation. Comme le montre Figure 30.

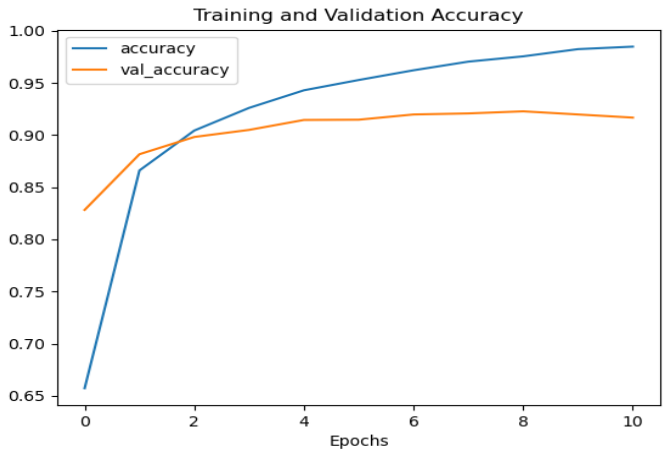


Figure 30 : Courbes de résultats de model VGG-16 affichage validation_ accuracy et accuracy.

De même, On a observé une tendance décroissante dans la courbe de loss (couleur bleue) et la courbe de validation_loss (couleur orange). Au début de l'entraînement, loss commence à une valeur relativement élevée de 1,22, tandis que validation_loss débute à 0,6. Au fur et à mesure de l'entraînement, les deux courbes montrent une diminution régulière, atteignant 0,34 à la fois pour loss et validation_loss à l'époque 2. Cette tendance à la baisse se poursuit jusqu'à la dernière époque (époque 10), où validation_loss atteint sa valeur minimale de 0,28 et loss atteint sa valeur minimale de 0,1. Comme le montre Figure 31.

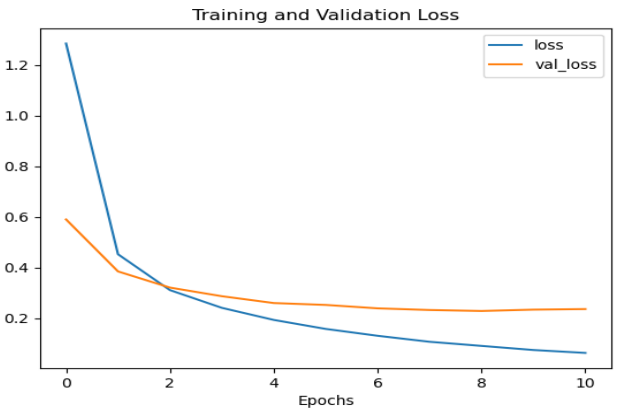


Figure 31 : Courbes de résultats de model VGG-16 affichage loss et validation_loss.

4.10.2. Pour le model ResNet-50 :

On a observé également une tendance à la hausse dans la courbe d'Accuracy(rouge) et la courbe validation_ accuracy (bleue) pour ResNet-50. À l'époque initiale (époque 0), l'Accuracy commence à 0,5, tandis que validation_ accuracy n commence à 0,67. À mesure que l'entraînement progresse, les deux courbes montrent une augmentation régulière, atteignant une valeur de 0,76 à l'époque 2. À partir de ce moment, l'Accuracy et validation_ accuracy continuent d'augmenter jusqu'à atteindre leurs sommets respectifs à l'époque finale (époque 10), où l'Accuracy est de 0,91 et validation_ accuracy est de 0,81. Ces résultats indiquent que la performance du modèle s'améliore de manière constante au cours de l'entraînement, comme le montrent les valeurs croissantes d'Accuracy pour les ensembles de données de training et de validation. Comme le montre Figure 32.

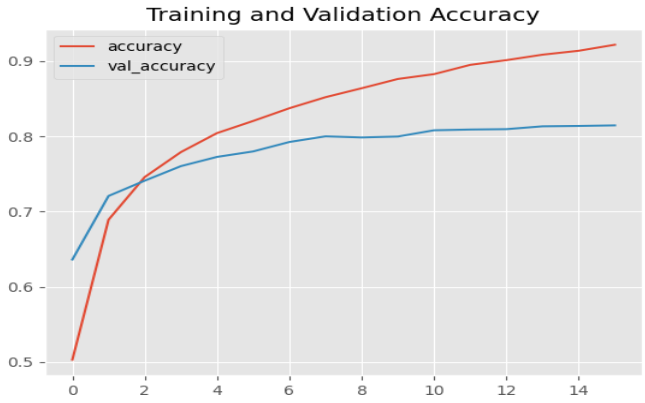


Figure 32: Courbes de résultats de model ResNet-50 affichage validation_ accuracy et accuracy.

En ce qui concerne la perte, on a observé une tendance à la baisse dans la courbe de loss (rouge) et la courbe de validation_loss (bleue). Au début de l'entraînement (époque 0), la loss commence à une valeur relativement élevée de 1,8, tandis que validation_loss commence à 1,25. À mesure que l'entraînement progresse, les deux courbes montrent une diminution régulière, atteignant une valeur de 0,85 pour loss et validation_loss à l'époque 2. Cette tendance à la baisse se poursuit jusqu'à l'époque finale (époque 10), où loss atteint sa valeur la plus basse de 0,23 et validation_loss atteint 0,6. Comme le montre Figure 33.

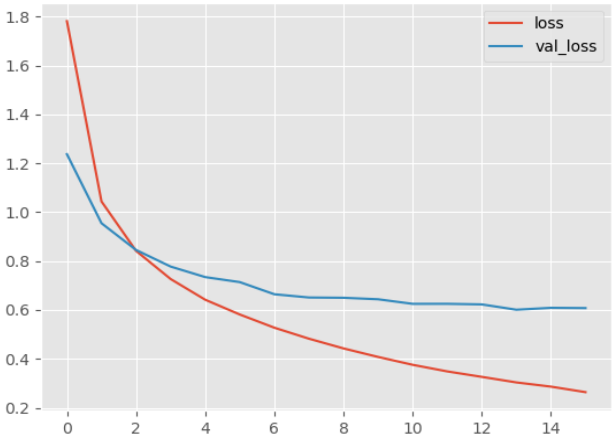


Figure 33: Courbes de résultats de model ResNet-50 affichage loss et validation_loss.

4.10.3. Pour le model InceptionV3 :

On a observé également une tendance à la hausse dans la courbe d'accuracy (bleue) et la courbe validation_ accuracy (orange) pour Inception-v3. À l'époque initiale (époque époch0), l'accuracy commence à 0,32, tandis que validation_ accuracy commence à 0,84. À mesure que l'entraînement progresse, les deux courbes convergent progressivement, atteignant une valeur de 0,87 à l'époque 1. À partir de ce moment, l'accuracy et validation_ accuracy continuent d'augmenter jusqu'à atteindre leurs sommets respectifs à l'époque 8, où l'accuracy est de 0,99 et validation_ accuracy est de 0,9. Ces résultats indiquent que la performance du modèle s'améliore de manière régulière au cours de la formation, comme le montrent les valeurs croissantes d'exactitude pour les ensembles de données de training et de validation. Comme le montre Figure 34.

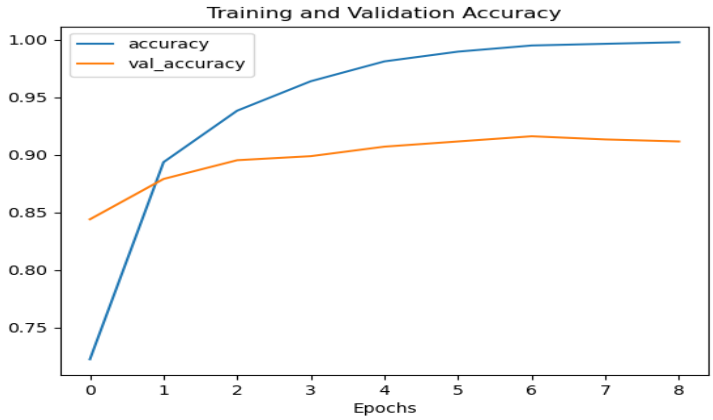


Figure 34 : Courbes de résultats de model Inception-v3 affichage validation_ accuracy et accuracy.

En ce qui concerne loss, On a observé une tendance à la baisse dans la courbe de loss (bleue) et la courbe de validation_loss (orange). Au début de l'entraînement (époque 0), loss démarre à une valeur relativement élevée de 1, tandis que validation_loss démarre à 0,5. À mesure que l'entraînement progresse, les deux courbes montrent une diminution régulière, atteignant une valeur de 0,4 pour loss et validation_loss à l'époque 2. Cette tendance à la baisse se poursuit jusqu'à l'époque 8, où loss atteint sa valeur la plus basse de 0,02 et validation_loss atteint 0,28. Comme le montre Figure 35.

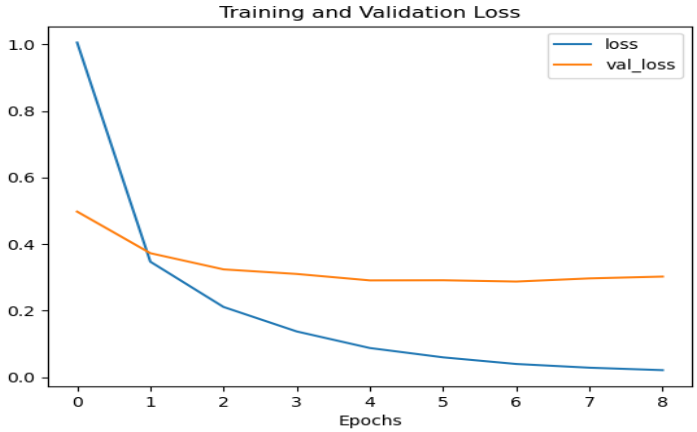


Figure 35 : Courbes de résultats de model Inception-v3 affichage loss et validation_loss.

4.10.4. Pour le model Dense Net-201 :

On a observé que la courbe d'accuracy (couleur rouge) et la courbe validation_ accuracy (couleur bleue) présentent une tendance à la hausse. À l'époque initiale (époque 0), l'accuracy commence à 0,4, tandis que validation_ accuracy commence à 0,93. Au fur et à mesure que l'entraînement progresse, les deux courbes convergent progressivement, atteignant une valeur de 0,9 à l'époque 1. À partir de ce point, l'accuracy et validation_ accuracy continuent d'augmenter jusqu'à atteindre leurs sommets respectifs à l'époque finale (époque 8), où l'accuracy est de 0,99 et validation_ accuracy est de 0,9. Ces résultats indiquent que la performance du modèle s'améliore régulièrement au cours de la formation, comme en témoignent les valeurs croissantes d'accuracy pour l'ensemble de données d'entraînement et de validation. Comme le montre Figure 36.

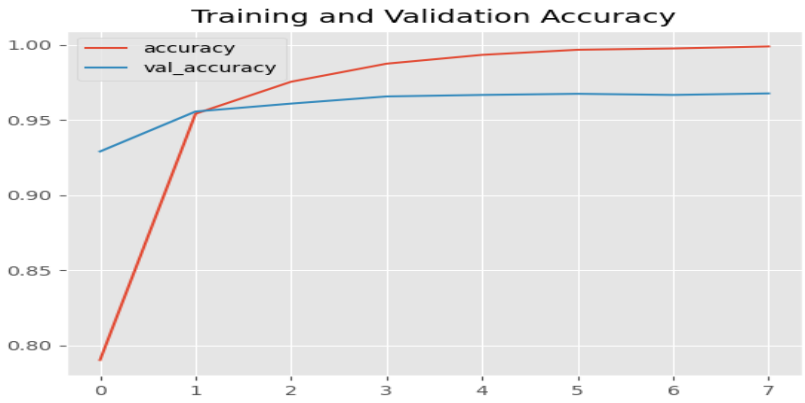


Figure 36 : Courbes de résultats de model DenseNet-201 affichage validation_ accuracy et accuracy.

De même, nous observons une tendance à la baisse dans la courbe de loss (couleur bleue) et la courbe de validation_loss (couleur orange). Au début de l'entraînement, loss démarre à une valeur relativement élevée de 0,85, tandis que validation_loss démarre à 0,25. Au fur et à mesure que l'entraînement progresse, les deux courbes montrent une diminution régulière, atteignant 0,19 pour loss et validation_loss à l'époque 1. Cette tendance à la baisse se poursuit jusqu'à l'époque finale (époque 7), où validation_loss atteint sa valeur la plus basse de 0,09, et loss atteint sa valeur la plus basse de 0,012. Comme le montre Figure 37.

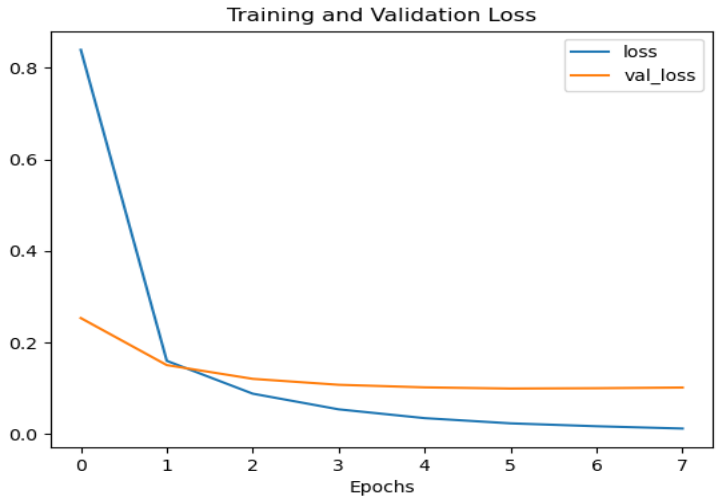


Figure 37: Courbes de résultats de model DenseNet-201 affichage loss et validation_loss.

4.11. Comparez les résultats des modelés :

Modelés	TRAINING ACCURACY	TRAINING LOSS	TEST ACCURACY	TEST LOSS
VGG 16	98.48%	0.0635	91.40%	0.2295
RESNET50	92.16%	0.2639	81.29%	0.6156
INCEPTION_V3	99.80%	0.0211	90.73%	0.2846
DENSENET201	99.89%	0.0120	96.80%	0.0925

Tableau 4 : Représenté les résultats des Modèles.

En termes d'Accuracy training, DENSENET201 présente la meilleure performance avec 99,89%, suivi de près par INCEPTION_V3 avec 99,80%. VGG16 obtient également un bon résultat avec l'Accuracy training, de 98,48%. RESNET50 affiche l'Accuracy training, la plus basse parmi les modèles comparés avec 92,16%.

En ce qui concerne loss-training, DENSENET201 enregistre la valeur la plus basse de 0,0120, suivie d'INCEPTION_V3 avec 0,0211. VGG16 présente une loss-training de 0,0635, tandis que RESNET50 a la plus haute loss- training de 0,2639.

En ce qui concerne les performances sur les tests, DENSENET201 se démarque à nouveau avec Accuracy de 96,80% et loss de 0,0925. INCEPTION_V3 affiche une Accuracy de 90,73% et loss de 0,2846. VGG16 obtient Accuracy de 91,40% et loss de 0,2295, tandis que RESNET50 a Accuracy la plus basse avec 81,29% et loss de 0,6156.

4.12. Conclusion :

En conclusion, DENSENET201présente les meilleures performances globales avec une haute Accuracy et une faible loss sur les ensembles d'entraînement et de tests. INCEPTION_V3 est également un bon choix avec Accuracy élevée et loss relativement basse. VGG16 obtient des résultats solides, mais RESNET50 montre des performances légèrement inférieures en termes d'Accuracy et de loss. Dans le dernier chapitre, nous allons sélectionner un modèle pour lequel nous développerons une application Android. Cette application permettra de faciliter utilisation.

Chapitre 5 : Le déploiement sur une application mobile

5.1. Introduction :

Dans ce chapitre, nous avons penché sur les aspects essentiels du déploiement d'applications basées sur l'apprentissage en profond et évaluons leur pertinence pour les utilisateurs et les applications réelles.

Lors de notre discussion sur le choix du modèle approprié pour notre application, nous avons délibéré sur les options disponibles telles que DenseNet-201 et Inception-V3, qui ont démontré de meilleures performances que VGG16 dans certains contextes. Cependant, nous avons finalement opté pour VGG16 pour des raisons spécifiques liées à notre infrastructure et aux besoins de l'application.

Dans ce chapitre, nous nous concentrerons sur la manière dont nous avons testé et évalué l'application basée sur VGG16, en mettant l'accent sur les performances, la convivialité et l'expérience utilisateur.

5.2. Pourquoi le VGG16 :

Il est important de souligner que notre choix de modèle, en l'occurrence VGG16, n'est pas basé uniquement sur sa performance par rapport à d'autres modèles d'apprentissage en profond. Nous prenons également en compte des facteurs tels que la compatibilité technique, les contraintes de taille et l'adéquation spécifique aux besoins de notre application.

La principale raison de ne pas avoir choisi DenseNet-201 ou Icenption-V3 est liée à la taille du modèle. Dans notre cas, il était crucial de maintenir la taille de TF Lite du modèle en dessous de 200 mégaoctets afin de garantir une intégration fluide dans notre environnement Java et d'optimiser les performances de l'application, nous n'avons malheureusement pas trouvé de solution satisfaisante pour ces deux modèles. Par conséquent, nous avons opté pour VGG16, qui a montré de bons résultats tout en respectant la contrainte de taille imposée.

5.3. Environnement et outils de développement :

5.3.1. JavaScript :

En utilisant JavaScript dans le développement d'applications Android pour le déploiement d'un modèle d'apprentissage en profond basé sur TensorFlow Lite, nous pouvons bénéficier de son rôle essentiel dans la création d'interfaces utilisateur interactives et réactives.

JavaScript est principalement utilisé dans le développement d'applications Web, mais grâce aux FrameWorks tels que React Native, il est également

possible d'utiliser JavaScript pour le développement d'applications Android. Avec React Native, nous pouvons écrire du code JavaScript et le compiler en une application Android native.[45]

Le rôle de JavaScript dans le déploiement d'un modèle d'apprentissage en profond à une application Android utilisant TensorFlow Lite est multiple. Voici quelques-uns de ses aspects clés :

1. Intégration du modèle : JavaScript facilite l'intégration du modèle d'apprentissage en profond dans l'application Android. Nous pouvons utiliser les fonctionnalités de TensorFlow Lite pour charger le modèle pré-entraîné, exécuter des inférences et obtenir des prédictions à partir des données d'entrée.

2. Gestion des interactions utilisateur : JavaScript permet de gérer les interactions utilisateur dans l'application Android. Par exemple, nous pouvons utiliser des bibliothèques telles que React Native pour créer des interfaces utilisateur interactives et réactives, capturer des entrées utilisateur, afficher les résultats du modèle d'apprentissage en profond et gérer les événements.

3. Communication avec les API : JavaScript facilite la communication avec les API et les services Web. Dans le cas du déploiement d'un modèle d'apprentissage en profond, nous pouvons utiliser JavaScript pour envoyer des requêtes aux serveurs d'API qui exécutent le modèle, envoyer des données d'entrée et recevoir les résultats du modèle.

4.Manipulation des données : JavaScript offre des fonctionnalités de manipulation des données, ce qui est important lorsqu'il s'agit de prétraiter les données d'entrée pour les adapter au format attendu par le modèle d'apprentissage en profond. Nous pouvons utiliser JavaScript pour effectuer des opérations telles que le redimensionnement des images, la normalisation des données, etc [46].

5.3.2. Android Studio :

Nous avons utilisé Android Studio pour environnement de développement intégré (IDE) principal pour créer l'application Android. Android Studio offre un ensemble complet d'outils et de fonctionnalités qui facilitent le développement, les tests et le déploiement d'applications Android. Nous avons spécifiquement travaillé avec la version 4.1.2 d'Android Studio, qui offrait un environnement de développement stable et fiable. De plus, nous avons utilisé la version 34.0.1 du kit de développement logiciel Android (SDK), qui fournissait une collection riche de bibliothèques, d'API et d'outils pour le développement d'applications Android. À des fins de test, on a utilisé l'appareil Redmi Note 11, fonctionnant sur la plateforme Android avec le niveau d'API 32. Cela nous a permis de garantir la compatibilité et d'évaluer les performances de notre application sur

un véritable appareil. Dans l'ensemble, Android Studio et le SDK ont joué un rôle essentiel en nous permettant de développer et de tester notre application de manière efficace et efficiente. [46]

5.4. Notre application :

L'application est conçue pour fonctionner avec deux types de fruits et un type de céréale. Ces choix ont été faits en se basant sur la ville de Médéa, célèbre pour sa production de pommes, de raisins et de maïs.

5.4.1. Fonctionnent d'application :

Fonctionnent d'application est composée de 4 interfaces qui donner les fonctionnements de cette application. La première interface est une page d'accueil qui affiche un message de bienvenue. Comme représenté Figure 38.



Figure 38 *:* Interface de bienvenue.

Ensuite, en appuyant sur le bouton 'Next', l'utilisateur est dirigé vers les trois interfaces de conseils d'utilisation. Comme représenté Figure 39.

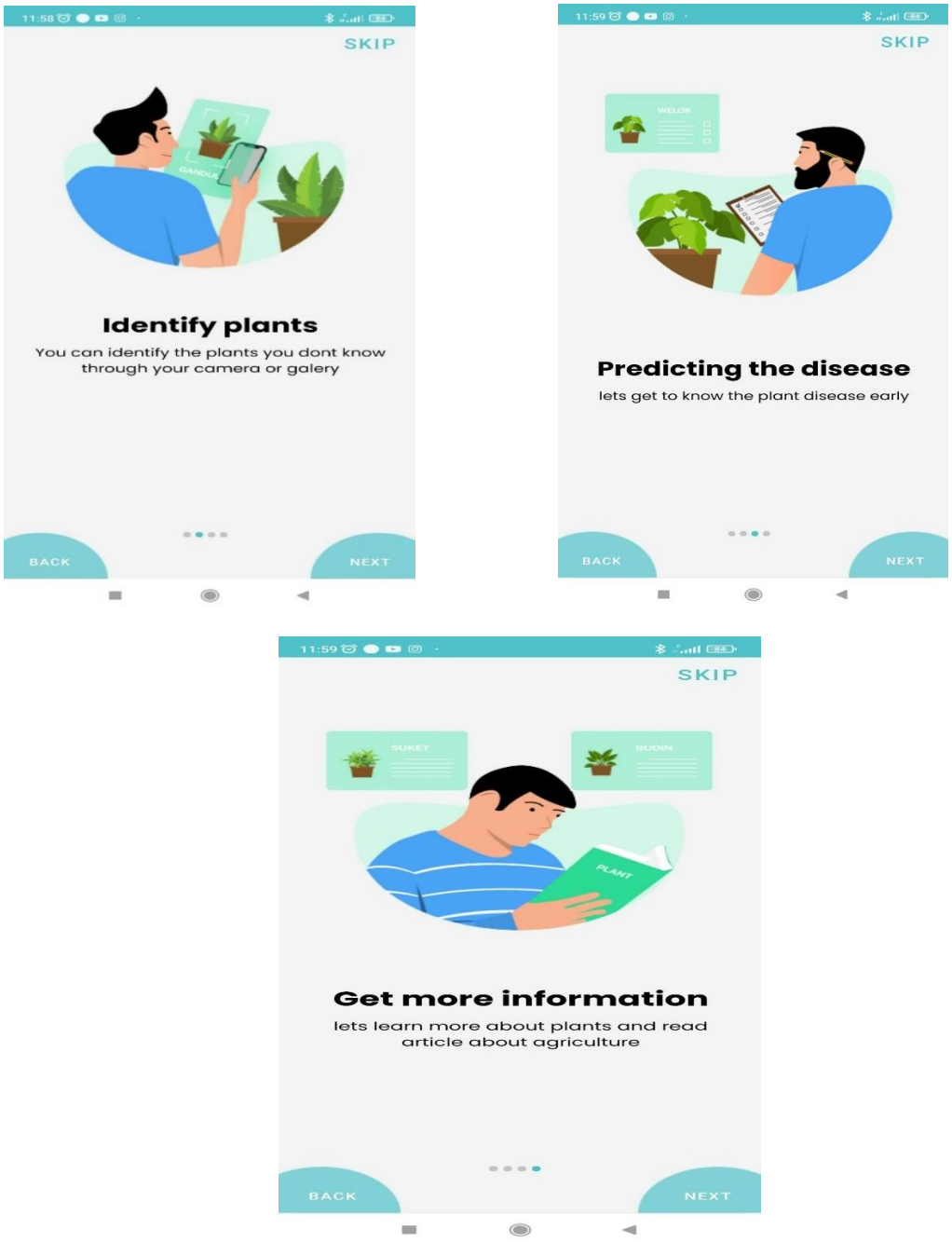


Figure 39: Interfaces de conseils d'utilisation

Après avoir passé par ces interfaces, l'interface suivante apparaît celui de la sélection des plantes**.**

5.4.2. Sélectionnez votre plante :

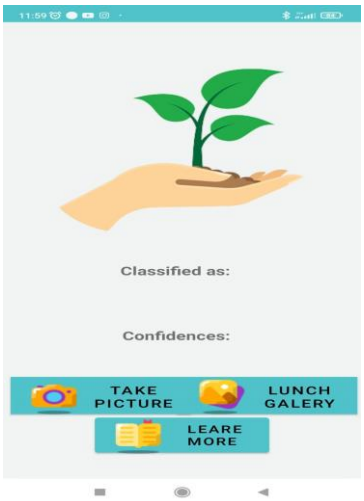
Cette interface permet à l'utilisateur de choisir le fruit qu'il souhaite explorer ou analyser, en l'occurrence les pommes et les raisins et le maïs. La figure 40 représente cette interface.



Figure 40 : Interface « Sélectionnez votre plante ».

5.4.3. Interface de prédiction :

Cette interface offre à l'utilisateur la possibilité de faire des prédictions ou des analyses sur les fruits sélectionnés en utilisant des photos soit prises directement ou sélectionnées depuis la galerie. Elle utilise des techniques d'apprentissage automatique ou d'apprentissage en profond pour fournir des informations précises sur les maladies ou les caractéristiques des plantes. De plus, cette interface contient un bouton qui permet d'accéder à d'autres sections pour en savoir plus sur les maladies associées. La figure 41 représente ces interfaces.





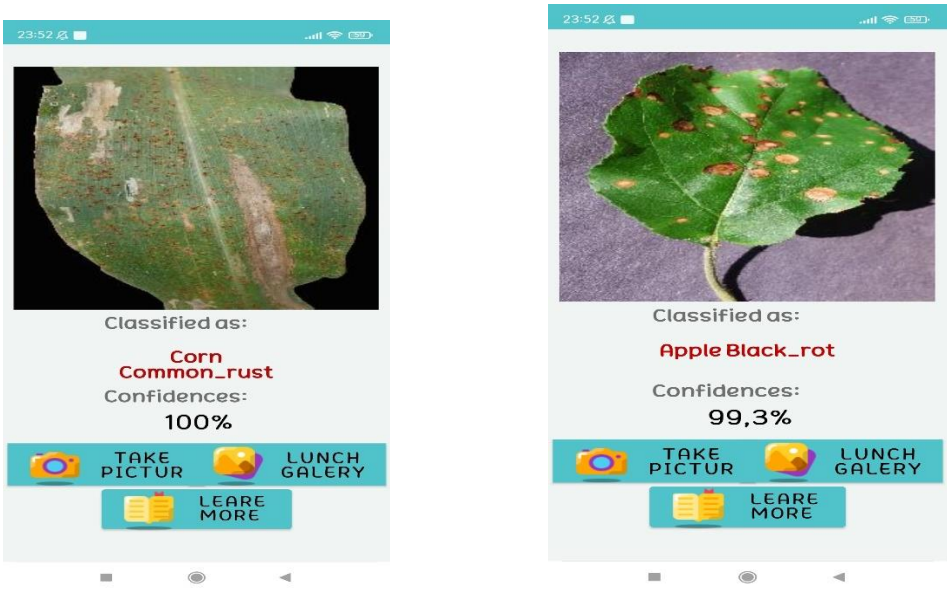


Figure 41: Interfaces « de prédiction ».

5.4.4. La liste des maladies :

Cette interface permet aux utilisateurs d'accéder à liste de maladie, y compris des études sur les maladies, les méthodes de traitement, les innovations dans l'agriculture et d'autres sujets connexes. Ils peuvent explorer les découvertes scientifiques récentes et approfondir leur compréhension des aspects techniques des fruits. La figure 42 représente cette interface.

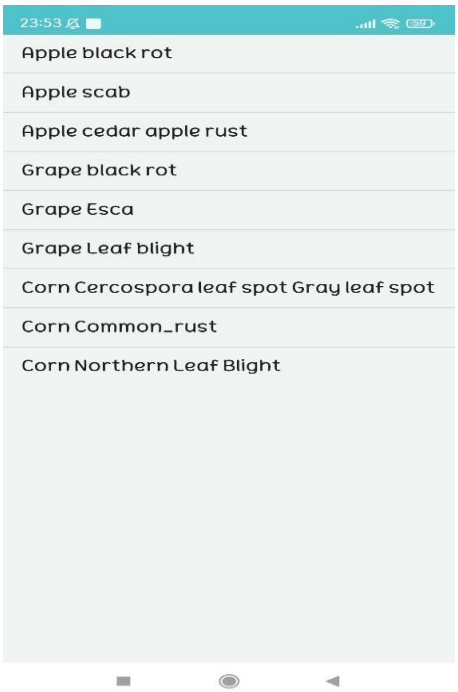


Figure 42: Interface « La liste des maladies ».

5.4.5. Accès aux blogs d'informations :

Sur cet écran, les utilisateurs peuvent accéder à des articles informatifs et éducatifs sur la culture, les maladies et les bonnes pratiques concernant les pommes et les raisins. Cela leur permet d'approfondir leurs connaissances sur ces fruits, d'apprendre comment prévenir et traiter les maladies spécifiques qui les affectent, et de découvrir des conseils et des astuces pour maximiser leur production et leur qualité. L'objectif de cette interface est d'éduquer les utilisateurs et de les aider à mieux comprendre les aspects liés à la culture et à la santé de ces fruits. La figure 43 représente un exemple de ces interfaces.

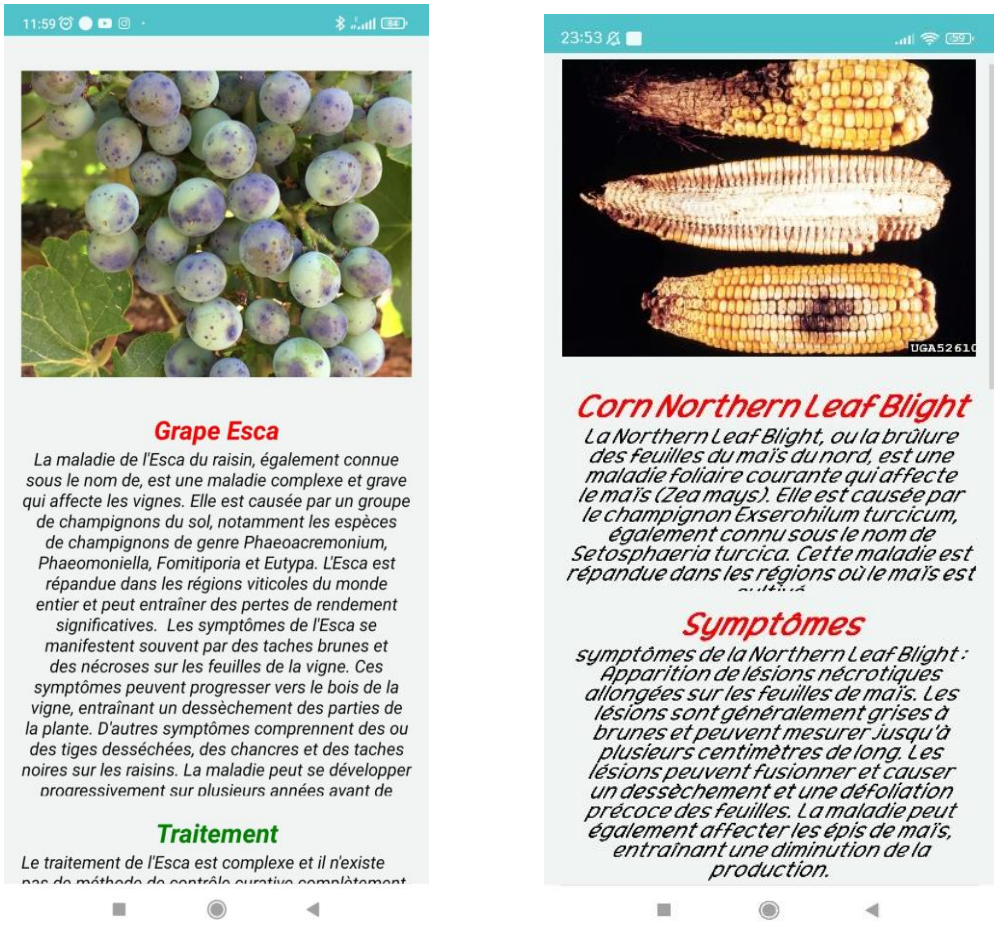


Figure 43: Exemple des interfaces « accès aux blogs d'informations ».

5.5. Conclusion :

En conclusion, la création d'une application basée sur l’apprentissage par transfert pour l'analyse des maladies des pommes et des raisins. Nous avons abordé différents aspects, notamment le choix du modèle de pré-entraînés. Le déploiement de l'application sur Android Studio et les différentes interfaces de l'application.

Pour le modèle, nous avons évalué plusieurs options telles que VGG16, DENSENET201 et INCEPTION_V3. Nous avons choisi VGG16 en raison de sa performance satisfaisante et de sa taille adaptée à nos besoins. Nous avons également pris en compte la contrainte de taille de l'application en évitant des modèles plus lourds comme DENSENET201 et INCEPTION_V3.

Les interfaces de l'application comprennent un écran d'introduction pour guider les utilisateurs, une interface de « sélection les plantes » pour choisir entre les pommes et les raisins et le maïs. Une interface de prédiction permettant d'analyser les maladies à l'aide de photos, un interface « Accès aux blogs d’informations » offrant des articles informatifs sur la culture des fruits et des conseils sur la prévention des maladies.

En résumé, cette application combine les avantages de l'apprentissage par transfert, de l'interface conviviale d'Android Studio et des connaissances agricoles pour offrir une solution pratique et informatique aux agriculteurs, aux passionnés de jardinage et à tous ceux qui s'intéressent à la santé des pommes, des raisins et du maïs.

Conclusion Générale :

En conclusion générale, ce projet de fin d'études a abordé plusieurs aspects essentiels du l'apprentissage en profond et du l'apprentissage par transfert, ainsi que les méthodes de déploiement et les tests d'évaluation. Les cinq chapitres du projet ont permis d'explorer ces sujets en profondeur et d'aboutir à des résultats significatifs pour le développement d'applications basées sur l'apprentissage par transfert.

On a jeté les bases du l'apprentissage en profond en fournissant une compréhension approfondie des concepts clés, des architectures de réseaux neuronaux profonds et des techniques d'apprentissage par transfert. Cette partie a permis d'établir les fondements nécessaires pour la suite du projet.

On a été consacré à l'étude approfondie des méthodes de déploiement des modèles d'apprentissage en profond. Différentes approches ont été examinées, notamment le déploiement sur serveur, sur le Cloud et sur des dispositifs mobiles. Les avantages et les considérations liées à chaque méthode ont été discutés, mettant en évidence l'importance de choisir la solution appropriée en fonction des besoins spécifiques du projet.

On a été consacré aux tests et à l'évaluation du l'apprentissage par transfert. Différentes stratégies ont été explorées, et des expériences ont été menées pour évaluer les performances des modèles pré-entraînés sur différents ensembles de données. Les résultats obtenus ont fourni des indications précieuses sur l'efficacité du l'apprentissage par transfert dans différentes situations.

On a abordé le développement d'une application concrète basée sur l'apprentissage en profond. Des outils tels qu'Android Studio ont été utilisés pour créer une interface utilisateur conviviale et fonctionnelle. Les différentes étapes de développement ont été discutées, y compris le choix du modèle, la gestion des ressources et l'intégration de fonctionnalités spécifiques. Des tests ont également été effectués pour évaluer les performances de l'application et garantir son bon fonctionnement.

Enfin, on a présenté les résultats obtenus à partir de l'application développée. Les performances du modèle d'apprentissage en profond ont été évaluées en termes d'accuracy et de loss et de validation. Les résultats ont démontré l'efficacité du modèle dans l'analyse des maladies des plantes, offrant ainsi des informations précieuses pour les agriculteurs et les passionnés de culture fruitière.

Dans l'ensemble, ce projet de fin d'études a permis d'approfondir les connaissances sur l’apprentissage en profond et l’apprentissage par transfert, ainsi que sur les méthodes de déploiement et les tests d'évaluation. L'application développée a démontré l'applicabilité concrète de ces concepts dans des domaines tels que l'agriculture. Les résultats obtenus ouvrent la voie à de futures recherches et à de nouvelles applications dans le domaine du l’apprentissage en profond.

Bibliographie :

[1] Jordan, M.I. and Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), pp.255-260.

[2] John D. Kelleher. Deep Learning. The MIT Press, 2019, p. 283. isbn: 9780262537551.

[3] https://www.jedha/vraie-difference-machine-learning-deep-learning (visite en 03\06\2023)

[4] https://omdena.com/blog/supervised-and-unsupervised-machine-learning/ (visite en 01\07\2023)

[5] Wu, Z., Xiong, Y., Yu, S.X. and Lin, D., 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3733-3742).

[6] Hu, François. "Semi-supervised learning in insurance: fairness and active learning." PhD diss., Institut polytechnique de Paris, 2022.

[7] Baesmat, K.H., Masoudipour, I. and Samet, H., 2021. Improving the performance of short-term load forecast using a hybrid artificial neural network and artificial bee colony algorithm amélioration des performances de la prévision de la charge à court terme à l'aide d'un réseau neuronal artificiel hybride et d'un algorithme de colonies d'abeilles artificielles. *IEEE Canadian Journal of Electrical and Computer Engineering*, *44*(3), pp.275-282.

[8] https://developersbreach.com/convolution-neural-network-deep-learning (visite en 01\07\2023)

[9] https://medium.com/swlh/simple-explanation-of-recurrent-neural-network (visite en 05\05\2023)

[10] Hirasawa, K., Ohbayashi, M., Koga, M. and Harada, M., 1996, June. Forward propagation universal learning network. In *Proceedings of international conference on neural networks (ICNN'96)* (Vol. 1, pp. 353-358). IEEE.

[11] Goh, A.T., 1995. Back-propagation neural networks for modeling complex systems. *Artificiel intelligence in engineering*, *9*(3), pp.143-151.

[12] Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[13] Hastomo, W., Karno, A.S.B., Kalbuana, N. and Meiriki, A., 2021, June. Characteristic parameters of epoch deep learning to predict Covid-19 data in Indonesia. In *Journal of Physics: Conference Series* (Vol. 1933, No. 1, p. 012050). IOP Publishing.

[14] Schmeiser, B., 1982. Batch size effects in the analysis of simulation output. *Operations Research*, *30*(3), pp.556-568.

[15] Batista, G.E., Prati, R.C. and Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, *6*(1), pp.20-29.

[16] Ng, A.Y., 1997, July. Preventing" overfitting" of cross-validation data. In *ICML* (Vol. 97, pp. 245-253).

[17] Braiek, H.B. and Khomh, F., 2020. On testing machine learning programs. *Journal of Systems and Software*, *164*, p.110542.

[18] https://www.superannotate.com /overfitting-and-underfitting

[19] Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S. and Jorge Cardoso, M., 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3* (pp. 240-248). Springer International Publishing.

[20] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Han, J., 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

[21] https://www.mygreatlearning.com/blog/deep-learning-applications/ (visite en 18\06\2023)

[22] Rastegari, M., Ordonez, V., Redmon, J. and Farhadi, A., 2016, September. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV* (pp. 525-542). Cham: Springer International Publishing.

[23] Jung, W., Jung, D., Kim, B., Lee, S., Rhee, W. and Ahn, J.H., 2019. Restructuring batch normalization to accelerate CNN training. *Proceedings of Machine Learning and Systems*, *1*, pp.14-26.

[24] Roska, T. and Chua, L.O., 1993. The CNN universal machine: an analogic array computer. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *40*(3), pp.163-173.

[25] https://www.kaggle.com/code/jagdmir/all-you-need-to-know-about-cnns (visite en 16\04\2023)

[26] Ba, J. and Frey, B., 2013. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, *26*.

[27] Wang, Z. and Majewicz Fey, A., 2018. Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery. *International journal of computer assisted radiology and surgery*, *13*, pp.1959-1970.

[28] Yang, Q., Zhang, Y., Dai, W. and Pan, S.J., 2020. *Transfer learning*. Cambridge University Press

[29] Qassim, H., Verma, A. and Feinzimer, D., 2018, January. Compressed residual-VGG16 CNN model for big data places image recognition. In *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)* (pp. 169-175). IEEE.

[30] Jaworek-Korjakowska, J., Kleczek, P. and Gorgon, M., 2019. Melanoma thickness prediction based on convolutional neural network with VGG-19 model transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0-0).

[31] Sarwinda, D., Paradisa, R.H., Bustamam, A. and Anggia, P., 2021. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. *Procedia Computer Science*, *179*, pp.423-431.

[32] Szyc, K., 2020. An impact of different images color spaces on the efficiency of convolutional neural networks. In *Engineering in Dependability of Computer Systems and Networks: Proceedings of the Fourteenth International Conference on Dependability of Computer Systems DepCoS-RELCOMEX, July 1–5, 2019, Brunów, Poland* (pp. 506-514). Springer International Publishing.

[33] https://www.researchgate.net/figure/DenseNet201. (Visite en 18\05\2023)

[34] Christian Szegedy, W.L., Liu, W., Jia, Y. and Sermanet, P., 2014. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*.

[35]] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[36] MOÏSE, K.K., MPIA, H.N. and VINGI, M.B., 2023. Déploiement mobile d'un réseau de neurones profond pour l'identification des étudiants de l'UAC/ISDA en salle d'examens [Mobile deployment of a deep neural network to identify UAC/ISDA'students in exam rooms]. *International Journal of Innovation and Applied Studies*, *39*(1), pp.136-155.

[37] Yaganteeswarudu, A., 2020, June. Multi disease prediction model by using machine learning and Flask API. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)* (pp. 1242-1246). IEEE.

[38] Hadullo, K.O. and Getuno, D.M., 2021. Machine learning software architecture and model workflow. A case of Django rest framework. *American Journal of Applied Sciences*, *18*(1), pp.152-164.

[39] Openja, M., Majidi, F., Khomh, F., Chembakottu, B. and Li, H., 2022, June. Studying the practices of deploying machine learning projects on docker. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022* (pp. 190-200).

[40] https://pypi.org/project/gradio/ (visite en 11\03\2023)

[41] https://www.tensorflow.org/lite (visite en 11\03\2023)

[42] https://plantvillage.psu.edu/ (visite en 10\03\2023)

[43] https://research.google.com/colaboratory/ (visite en 10\03\2023)

[44] https://www.python.org/doc/. (Visite en 10\03\2023)

[45] https://www.analyticsvidhya.com/blog/tensorflow (visite en 12\03\2023)

[46] https://developer.android.com/studio (visite en 12\03\2023)