الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

المدرسة الوطنية المتعددة التقنيات بالجزائر

Ecole Nationale Polytechnique d'Alger

Département Electronique

MEMOIRE DE PROJET DE FIN D'ETUDES

En vue de l'obtention du

Diplôme d'Ingénieur d'état En Electronique

Estimation de la pose de la caméra basée sur un réseau neuronal convolutif

Réalisé par :

Mlle. Houria HALLALI

Encadré par :

M. Elhaouari KOBZILI Docteur ESTA

M. Cherif LARBES Professeur ENP

Co-encadré par :

M. Abdellah GRINI ESTA

Présentée et soutenue publiquement le 06/07/2022 devant le jury composé des membres :

Président	M. Rachid ZERGUI	M.A.A	ENP
Examinateur	M. Mohamed TAGHI	M.A.A	ENP

ENP 2022

الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

المدرسة الوطنية المتعددة التقنيات بالجزائر

Ecole Nationale Polytechnique d'Alger

Département Electronique

MEMOIRE DE PROJET DE FIN D'ETUDES

En vue de l'obtention du

Diplôme d'Ingénieur d'état En Electronique

Estimation de la pose de la caméra basée sur un réseau neuronal convolutif

Réalisé par :

Mlle. Houria HALLALI

Encadré par :

M. Elhaouari KOBZILI Docteur ESTA

M. Cherif LARBES Professeur ENP

Co-encadré par :

M. Abdellah GRINI ESTA

Présentée et soutenue publiquement le 06/07/2022 devant le jury composé des membres :

Président	M. Rachid ZERGUI	M.A.A	ENP
Examinateur	M. Mohamed TAGHI	M.A.A	ENP

ENP 2022

ملخص

قدم التعلم العميق دقة مثيرة للاهتمام في مجال تصنيف الصور، واكتشاف الأشياء، واستعادة الصور، وتجزئة الصور ومهام الرؤية الأخرى.
تم تطبيق فكرة التعلم العميق مؤخرًا على الانحدار الكاميرا من خلال صورة، انحدار الكاميرا أو التوطين هو مجال بحث مثير للاهتمام للغاية، وهو يمثل مشكلة رئيسية في مجال الروبوتات ورؤية الكمبيوتر. وهو يتألف من العثور على الموضع والتوجه ثلاثي الأبعاد للكاميرا من خلال صورة أو مجموعة من الصور. ويعتبر مهما بالنسبة لتطبيقات مثل التنقل في المركبات المستقلة، والهيكل من الحركة والواقع المعزز والتوطين المتزامن ورسم الخرائط.

في هذا العمل، نقترح طريقة تعليمية عميقة لتوطين الكاميرا. تقوم هذه الطريقة بتوطين الكاميرا من الصور المحددة باستخدام شبكة عصبية تلافيفيه. يتم تدريب الشبكة العصبية على تقدير موضع الكاميرا باستخدام صور التدريب، نظهر ايضا أنه يمكن تحقيق الانحدار الكاميرا باستخدام التعليم المتنقل.

الكلمات المفتاحية: التعلم العميق، الموقع، الكاميرا، الانحدار، التعليم المتنقل، الشبكات العصبية التلافيفية

Abstract:

Deep learning has provided interesting precision for image classification, object detection, image restoration, image segmentation, and other vision tasks.

The idea of deep learning has recently been applied to camera pose regression from an RGB image. Camera poses regression or localization is a very interesting research area, it represents a key problem in terms of robotics and computer vision. It consists in finding the position and the 3D orientation of a camera from an image or a set of images. It is fundamental for such applications as autonomous vehicle navigation, structure from motion (SfM), augmented reality (AR) and simultaneous localization and mapping (SLAM).

In this work, we propose a deep learning-based method for camera localization. This method localizes the camera pose from the given images using a convolutional neural network (CNN).

The neural network is trained for camera pose estimation using training images. We show that camera pose regression could be achieved using transfer learning.

Keywords: Deep Learning, Localization, Camera, Regression, Transfer Learning, CNN.

Résume :

L'apprentissage profond a permis d'obtenir une précision intéressante pour la classification d'images, la détection d'objets, la restauration d'images, la segmentation d'images et pour d'autres tâches de vision.

L'idée de l'apprentissage profond a récemment été appliquée à la régression de la pose de la caméra à partir d'une image RGB. La régression ou la localisation de la pose de la caméra est un axe de recherche très intéressant, elle représente un problème essentiel en termes de robotique et de vision par ordinateur. Elle s'agit de retrouver la position et l'orientation 3D d'une caméra à partir d'une image ou d'un ensemble d'images. Elle est fondamentale pour de telles applications comme la navigation de véhicules autonomes, la structure à partir du mouvement (SfM), la réalité augmentée (AR) et la localisation et la cartographie simultanées (SLAM).

Nous proposons dans ce travail une méthode basée sur l'apprentissage profond pour la localisation de la caméra. Cette méthode permet de localiser la pose de la caméra à partir des images données en utilisant un réseau de neurones convolutifs (CNN).

Le réseau neuronal est entraîné pour l'estimation de la pose de caméra en utilisant des images d'entraînement. Nous montrons que la régression de la pose de la caméra a été rendue possible grâce à l'apprentissage par transfert.

Mots clés : Apprentissage profond, Localisation, Camera, Régression, Apprentissage par transfert, CNN.

Dédicace

À mes chers parents.

À Mes chers frères et sœurs.

À mes chères amies

À tous mes proches.

Remerciements

En préambule à cette thèse, je tiens à exprimer mes sincères remerciements à toutes les personnes qui m'ont aidé et contribué à l'élaboration de ce travail.

Je tiens à remercier M. KOBZILI Elhaouari pour avoir accepté de m'encadrer dans la réalisation et l'élaboration de ce travail, et également pour son dévouement malgré ses nombreuses occupations.

En particulier, je tiens à remercier M. GRINI Abdellah, grâce à ses conseils et le temps qu'il a bien voulu me consacrer, m'a permis de travailler dans les meilleures conditions possibles.

Je remercie aussi M. LARABES Cherif pour son intervention dans les orientations de mon travail et pour son soutien.

Je remercie en outre les membres du jury de cette thèse : Mohamed TAGHI et Rachid ZERGUI pour l'intérêt qu'ils ont porté à mon travail et qui ont accepté de juger ce travail.

Je tiens à complimenter toute l'équipe pédagogique de l'ENPA et les intervenants professionnels de la formation d'Ingénieur pour avoir fourni toutes les bases et nécessaires à ce travail, ainsi que toutes les personnes de l'École Supérieure des Techniques Aéronautiques (ESTA) qui ont collaboré à la réalisation de ce travail.

Enfin, j’adresse mes plus sincères remerciements à tous ceux qui ont participé à la réalisation de ce modeste travail.

Merci à vous tous.

Table des matières

Liste des tableaux

Chapitre 2

Chapitre 3

Chapitre 4

Liste des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3

Chapitre 4

Liste des abréviations

SLAM: Simultaneous localization and mapping.

SFM: Structure from Motion.

GPS: Global Positioning System

VO: Visual Odometry

APR: Absolute Pose Regression

RPR : Relative Pose Regression

IA : Intelligence artificielle

ML: Machine Learning

DL: Deep Learning

CNN: Convolutional Neural Network

RNN: Recurrent neural networks

ResNet: Residual Network

VGG: Visual Geometry Group

MAE: Mean Absolute Error

Introduction Générale

L'estimation de la position et de l'orientation de la caméra qui a capturé une image est un problème fondamental en vision par ordinateur. L'estimation de la pose du capteur est nécessaire pour une grande variété de tâches pratiques telles que la reconstruction 3D, le positionnement d'outils de robots et la réalité augmentée.

Pour calculer la pose de la caméra dans ces scénarios, il est généralement nécessaire d'avoir un système externe pour déterminer la position du capteur, comme les systèmes de capture de mouvement basés sur des points clés. La méthode la plus courante pour calculer la pose de la caméra à partir des images elles-mêmes est la structure à partir du mouvement (SfM). Les étapes de la SfM comprennent souvent l'extraction de caractéristiques, la mise en correspondance des caractéristiques et l'enregistrement des images. Ces étapes nécessitent plusieurs images qui se chevauchent afin de trouver de bonnes correspondances d'images pour la localisation.

Au cours des dernières décennies, différents algorithmes ont été proposés pour l'estimation des paramètres extrinsèques de la caméra (la translation et la rotation).

Récemment, les méthodes basées sur les réseaux de neurones convolutifs (CNN) ont nettement dépassé les résultats de l'état de l'art dans de nombreux problèmes de vision par ordinateur, tels que la classification d'images, la reconnaissance d'objets et la récupération d'images. Un réseau neuronal convolutif (CNN) est entraîné, en général avec un grand nombre de données, pour résoudre le problème spécifique.

Pour le problème de l'estimation de la pose de la caméra, différentes approches basées sur les réseaux neuronaux convolutifs ont également été proposées dernièrement, avec des résultats intéressants.

Dans ce travail, nous montrons comment les CNN peuvent aussi être appliqués pour estimer les poses de la caméra.

Le but de notre travail est de pouvoir estimer la pose de la caméra directement en traitant les images de cette caméra. Nous proposons un modèle basé sur un réseau de neurones convolutif et également des approches d'apprentissage par transfert qui permettent de prédire le vecteur de pose de la caméra à sept dimensions.

Structure de mémoire

Ce travail est subdivisé en quatre chapitres : Un aperçu sur chacun d'eux est donné ci-dessous

• Méthodes de la localisation : Ce chapitre est dédié aux concepts de base et un aperçu sur la localisation et les différentes méthodes de la localisation.

• Réseaux Neurones : dans ce chapitre, nous allons présenter les bases de l'apprentissage en profondeur et les réseaux de neurones convolutifs.

• La localisation par réseaux de neurones convolutifs : Ce chapitre présente le cœur de ce mémoire qui porte sur la conception des modèles CNN pour déterminer la pose de la caméra.

• Résultats expérimentaux et évaluation : Ce chapitre est dédié à l'évaluation des performances, ainsi que les diffèrent résultats obtenus.

Chapitre 1

Les méthodes de Localisation

1.1 Introduction

La performance de nombreuses applications de vision par ordinateur, telles que la navigation de véhicules autonomes et la robotique mobile, dépendent fortement de la bonne localisation du système par rapport à son environnement.

Si un robot est perdu ou incertain de sa position dans l'environnement, il doit déduire sa position à partir d'informations extéroceptives introspectives.

Pour la plupart des techniques de localisation, ceci est accompli en estimant une distribution de probabilité sur les positions possibles du robot dans une carte. Lorsqu'une hypothèse de localisation domine la distribution de probabilité, généralement en raison de mesures suffisamment informatives, la localisation est complète.

Il existe de nombreuses méthodes de localisation, parmi toutes les thématiques, on peut trouver la localisation par GPS. Cette dernière présente de nombreux inconvénients comme la multiplication des signaux dans le cas de la navigation extérieure et elle est absente dans le cas de la localisation intérieure. Les solutions classiques de localisation sont basées essentiellement sur les techniques SLAM mais ces dernières souffrent également de nombreux inconvénients comme la robustesse face aux changements d'environnement sévère.

Le but de ce chapitre est de présenter les principaux capteurs de la localisation proprioceptifs et extéroceptifs de la robotique mobile, puis d'expliquer les différentes techniques de localisation existantes.

1.2 Les capteurs de la localisation

Un capteur est un dispositif électrique/mécanique/chimique qui transforme un attribut de l'environnement en une mesure quantitative. Les robots utilisent différents capteurs pour détecter différentes grandeurs de l'environnement.

Tous les capteurs sont caractérisés par une fréquence d'acquisition (temps de réponse), une résolution de mesure, un bruit sur la grandeur physique mesurée...

De même, le principe de mesure détermine si le capteur effectue une mesure absolue ou relative.

Les capteurs peuvent être classés en deux catégories comme suit :

1.2.1 Les capteurs proprioceptifs

Les capteurs proprioceptifs permettent de mesurer des valeurs internes au système, par exemple les robots mobiles (accéléromètres, gyroscopes, etc.).

1.2.1.1 Les accéléromètres

Le capteur accéléromètre est le dispositif électromécanique utilisé pour mesurer l'accélération de tout corps ou objet dans son cadre de repos instantané.

Le taux de variation de la vitesse d'un corps par rapport au temps est appelé accélération. Selon la théorie relative, en fonction de l'objet relatif pris pour mesurer l'accélération, il existe deux types d'accélération.

L'accélération propre, qui est l'accélération physique du corps par rapport à l'inertie ou à l'observateur qui est au repos par rapport à l'objet mesuré.

L'accélération coordonnée, qui dépend du choix du système de coordonnées et du choix des observateurs. Elle n'est pas égale à l'accélération propre.

1.2.1.2 Les gyroscopes

Le capteur gyroscope est un dispositif capable de mesurer et de maintenir l'orientation et la vitesse angulaire d'un objet. Ils sont plus avancés que les accéléromètres. Ils peuvent mesurer l'inclinaison et l'orientation latérale de l'objet alors que les accéléromètres ne peuvent mesurer que le mouvement linéaire.

Les capteurs gyroscopiques sont également appelés capteurs de vitesse angulaire. Ces capteurs sont installés dans les applications où l'orientation de l'objet est difficile à percevoir par un être humain.

1.2.2 Capteurs extéroceptifs

Les capteurs extéroceptifs acquièrent des informations relatives liées au l'environnement du robot. (Sonar - lasers à distance - caméra -GPS, etc.) [1]

1.2.2.1 Les capteurs sonar

Les capteurs sonars utilisent les ondes acoustiques pour détecter des objets et mesurer les distances entre le capteur et les objets cibles. Ils comportent deux parties principales, à savoir l'émetteur et le récepteur. L'émetteur envoie une courte impulsion ultrasonore et le récepteur reçoit ce qui revient du signal après qu'il se soit réfléchi sur les objets proches. Le capteur mesure le temps de vol (TOF), c'est-à-dire le temps entre l'émission du signal et sa réception.

1.2.2.2 Les capteurs laser

Les capteurs laser peuvent être utilisés dans plusieurs applications liées au positionnement. Il s'agit d'une technique de télédétection pour la mesure de la distance qui consiste à transmettre un laser vers la cible, puis à analyser la lumière réfléchie. Les mesures de distance basées sur le laser dépendent des techniques TOF ou de déphasage. Une courte impulsion laser est

envoyée, et le temps jusqu'à son retour est mesuré. Ce type de capteur est souvent appelé radar laser ou capteur de détection et de télémétrie par la lumière (LIDAR).

1.2.2.3 Les cameras

Les caméras et les systèmes de vision peuvent être utilisés dans des applications robotiques mobiles pour la localisation et l'exécution de diverses tâches.

Une caméra pour la robotique est conçue pour capturer des informations d'images et les envoyer non comprimées pour le traitement. Avec les caméras grand public standard, les données d'image sont compressées, ce qui donne une image lisse, mais n'offre pas la qualité requise pour les applications robotiques.

La caméra idéale pour un système donné dépend des éléments à examiner, de la vitesse à laquelle les objets se déplacent, de l'éclairage, de la température de fonctionnement, de l'espace requis et du coût.

1.3 Types de caméras utilisées en localisation

La localisation peut être classée en fonction du type de caméra utilisé pour estimer le positionnement. Plusieurs types de caméras, tels que les caméras stéréoscopiques, monoculaires, omnidirectionnelles-stéréos et les caméras RGB-D, peuvent être utilisés pour les buts de localisation. La Figure I-1 montre différents types de caméras. [2]

Figure 1-1 : Différents types de caméras. a) Caméra stéréo. b) Stéréo omnidirectionnelle. c) Caméra monoculaire. d) Monoculaire omnidirectionnelle

1.3.1 Cameras passives

1.3.1.1 Caméra monoculaire

Une caméra monoculaire est un type habituel de capteur de vision destiné aux applications de conduite automatisée. Quand elle est montée sur un véhicule ego, cette caméra peut reconnaître des objets, déterminer les limites de la voie et suivre des objets dans une situation donnée.

1.3.1.2 Caméra stéréo

Une caméra stéréo est un modèle de caméra équipé de deux capteurs d'images ou plus. Cela permettant à la caméra de simuler la vision binoculaire humaine et lui donnant ainsi la possibilité de détecter la profondeur.

1.3.1.3 Caméra omnidirectionnelle

Une caméra omnidirectionnelle capable de voir dans toutes les directions (avec un champ de vision de 360 degrés) dans le plan horizontal, ou avec un champ visuel qui couvre un hémisphère ou (à peu près) la totalité de la sphère.

1.3.2 Cameras actives

1.3.2.1 Lidar

Le LIDAR, qui signifie Light Detection and Ranging (détection et télémétrie par la lumière), est un laser pulsé qui mesure le temps nécessaire pour que le signal retourne à la source (à la vitesse d'une nanoseconde), ce qui lui permet de générer un modèle 3D avec une plus grande précision qu'une simple caméra.

1.3.2.2 Time of flight

Une caméra 3D à temps de vol (TOF) est une petite caméra de détection en temps réel basée sur le principe du temps de vol (TOF). Elle fonctionne en éclairant la scène avec une source de lumière modulée et en observant la lumière réfléchie. Le déphasage entre l'illumination et la réflexion est mesuré et exprimé en distance.

1.3.2.3 RGB-Depth

Les caméras RGB-D sont des systèmes de détection qui capturent des images RGB avec des informations de profondeur par pixel. Les caméras RGB-D s'appuient sur la stéréo active ou la détection du temps de vol pour générer des estimations de profondeur sur un grand nombre de pixels. [3]

1.3.3 Cameras hybrides

1.3.3.1 Multiples capteurs

Les caméras multi-capteurs permettent de balayer de plus larges surfaces, ce qui rend inutile l'installation de nombreuses caméras à capteur unique. La combinaison de deux capteurs et de deux objectifs dans un seul et même boîtier peut donner à l'unité de caméra la possibilité de couvrir une surface de 180 degrés.

1.4 Types d'environnement

Typiquement, le problème de la localisation consiste à estimer la pose par rapport à une carte de l'environnement.

Les environnements peuvent être divers et varier, posant des problématiques différentes. En robotique, on peut différencier les environnements de plusieurs manières : par exemple, intérieur/extérieur ou statique/dynamique. [4]

- **Environnement intérieur** : Les systèmes de positionnement sont fortement dégradés ou peuvent échouer complètement dans des environnements intérieurs où les signaux satellites ou cellulaires sont interrompus
- **Environnement extérieur** : Dans les scénarios extérieurs, la position du terminal mobile est obtenue avec une grande précision
- **Environnement statique** : Il s'agit d'un type d'environnement dans lequel le seul objet mobile dans l'environnement sera le robot. Autrement dit, pendant toute la période de fonctionnement du robot, chaque objet de l'environnement reste exactement au même endroit.
- **Environnement dynamique** : Il s'agit d'un type d'environnement dans lequel le robot n'est pas le seul objet pouvant être déplacé. Les objets de l'environnement peuvent changer de position et d'orientation. La fréquence de mouvement des objets dynamiques peut varier considérablement.

1.5 La localisation

De manière générale, la localisation d'un objet consiste à trouver une réponse à la question de base « où suis-je ? »

La localisation fait référence à la capacité de déduire automatiquement la pose et la position d'un observateur par rapport à un modèle. [5]

1.6 Types de localisation

1.6.1 Localisation relative

Elle consiste à évaluer la position et l'orientation en utilisant les informations fournies par divers capteurs embarqués.

1.6.2 Localisation absolue

Elle permet d'obtenir la position absolue en utilisant des balises, des points de repère ou des signaux satellitaires (par exemple, GPS). [6]

Pour la localisation absolue, la croissance de l'erreur est réduite lorsque des mesures sont effectuées. La position est déterminée de l'extérieur et la précision est souvent indépendante du temps et de la position.

Le problème de la localisation absolue est que nous ne pouvons pas suivre la trajectoire sur de petites distances.

1.6.3 Localisation hybride

Dans les méthodes de localisation actuelles, les chercheurs associent des techniques de localisation afin de pouvoir bénéficier des avantages de l'une et de réduire les inconvénients de l'autre [7], et afin d'obtenir une précision élevée dans le calcul de la localisation. La localisation hybride associe les données des capteurs extéroceptifs à celles des capteurs proprioceptifs pour estimer la pose du robot. Les informations fournies par les différents capteurs sont fusionnées à l'aide de filtres tels que le filtre de Kalman.

La performance de la localisation hybride en termes de localisation s'est avérée meilleure que les techniques existant avec un nombre croissant de capteurs connus dans le réseau.

1.7 Les différentes techniques de localisation

1.7.1 La localisation par satellites

Les satellites sont les fondements du positionnement et de la navigation modernes. Ils sont utilisés aussi bien par les systèmes de navigation des véhicules que par les smartphones et les géomètres. Sur la base des signaux satellites, le récepteur peut définir sa position n'importe où dans le monde avec une précision de quelques mètres en moins d'une minute. De plus, le temps peut être défini en tant que sous-produit avec une précision d'environ une centaine de nanosecondes.

1.7.1.1 Le système mondial de navigation par satellites (GNSS)

Le système mondial de navigation par satellite (GNSS) est une série de satellites qui transmettent des signaux destinés à être utilisés dans des applications de navigation et de positionnement, partout sur la surface de la terre. Il existe actuellement deux systèmes mondiaux de navigation par satellite en service : le système américain de positionnement global (GPS) et le système russe de navigation globale par satellite (GLONASS). Un troisième système, Galileo, est en cours de développement en Europe. Galileo fournira une large gamme de signaux et de services de positionnement, y compris le service de haute fiabilité requis par les applications critiques pour la sécurité de la vie. Galileo fournira également un haut niveau d'intégrité à l'échelle mondiale et sans système d'augmentation supplémentaire.

1.7.1.2 Le système mondial de positionnement GPS

Le GPS est un système de navigation et de synchronisation exploité par le département de la défense des États-Unis (DoD), et comporte donc un certain nombre d'aspects classifiés. Plusieurs organisations surveillent les signaux GPS de manière indépendante et fournissent des services à partir desquels il est possible d'obtenir les éphémérides des satellites et le comportement des horloges. Des précisions de l'ordre de 5 à 10 cm ne sont pas inhabituelles. Les mesures de la phase de la porteuse des signaux transmis sont généralement effectuées avec une précision supérieure à un millimètre. [8]

Ce système se compose de trois segments : le segment spatial, le segment de contrôle et le segment utilisateur.

Le segment spatial du GPS est constitué d'une constellation de satellites qui transmettent des signaux radio aux utilisateurs. Les États-Unis se sont engagés à maintenir la disponibilité d'au moins 24 satellites GPS opérationnels, 95 % du temps.

Pour garantir cet engagement, l'U.S. Space Force fait voler 31 satellites GPS opérationnels depuis plus de dix ans.

1.7.2 La localisation inertielle

La localisation inertielle est basée sur des mesures fournies par des accéléromètres et des gyroscopes afin de réaliser un système de localisation ou de navigation inertielle

Les systèmes de navigation inertiels (INS) sont des systèmes de navigation capable de calculer la position, soit par rapport à un système/point de référence, soit par rapport à des coordonnées absolues. Un système INS est composé au minimum de trois gyroscopes et de trois accéléromètres permettant au système de dériver une solution de navigation. Cette solution de navigation contient au moins la position.

Le concept de base d'un système INS est la mesure des changements dans le mouvement relatif (par la mesure de l'accélération) pour projeter une position changeante dans un certain cadre de référence inertielle dans le temps. Le cœur d'un système INS est son unité de mesure inertielle (IMU). Ce mécanisme est composé de trois gyroscopes orthogonaux et de trois accéléromètres orthogonaux.

1.7.3 La localisation visuelle

La localisation visuelle, qui permet de déterminer l'emplacement à partir d'images, est une méthode utilisée par les robots et les voitures autonomes pour estimer leur position. Elle est également utilisée dans les applications de réalité augmentée pour interagir avec le monde physique, à l'intérieur comme à l'extérieur.

1.7.3.1 Simultaneous Localization and Mapping (SLAM)

SLAM signifie cartographie et localisation simultanées (parfois appelées localisation et cartographie synchronisées). Il s'agit du processus consistant à cartographier une zone tout en gardant la trace de l'emplacement de l'appareil dans cette zone. C'est ce qui rend la cartographie mobile possible. Cela permet de cartographier de grandes zones dans des délais beaucoup plus courts, car les zones peuvent être mesurées à l'aide de robots mobiles, de drones ou de véhicules. Les systèmes SLAM simplifient la collecte de données et peuvent être utilisés dans des environnements extérieurs ou intérieurs.

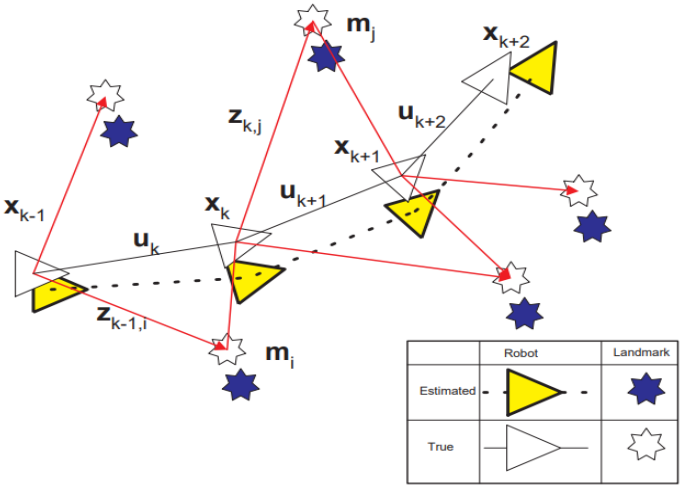

Figure 1-2 : Le problème essentiel du SLAM [9]

Selon les différentes générations de cartes, le SLAM se divise en quatre parties : le SLAM métrique géométrique, le SLAM d'apprentissage, le SLAM topologique et le SLAM à marqueurs.

Le SLAM d'apprentissage est une nouvelle direction de recherche récente. Il peut obtenir la pose de la caméra et la carte 3D, mais a besoin d'un jeu de données préalables pour entraîner le réseau. Les performances du SLAM d'apprentissage dépendent dans une large mesure du jeu de données utilisées et sa capacité de généralisation est faible. Par conséquent, le SLAM d'apprentissage n'est pas aussi flexible que le SLAM à métrique géométrique et sa carte 3D obtenue à l'extérieur.

La plupart du temps, l'ensemble de données utilisé n'est pas aussi précis que le SLAM géométrique. Cependant, le SLAM à apprentissage simultané dispose d'une carte 3D autre que les représentations topologiques.

Le SLAM à marqueurs calcule les positions de la caméra à partir de marqueurs structurés connus sans connaître l'environnement complet.

Le SLAM métrique géométrique comprend le SLAM monoculaire, le SLAM multioculaire et le SLAM à capteurs multiples. [10]

1.7.3.2 Le SLAM visuel (VSLAM)

Le SLAM visuel est un type spécifique de système SLAM qui exploite la vision 3D pour exécuter des fonctions de localisation et de cartographie lorsque ni l'environnement ni l'emplacement du capteur ne sont connus. La technologie SLAM visuel se présente sous différentes formes, mais le concept global fonctionne de la même manière dans tous les systèmes SLAM visuels.

Les systèmes SLAM visuels sont également utilisés dans une grande variété de robots de terrain. Par exemple, les rovers et les atterrisseurs. Les robots de terrain dans l'agriculture, ainsi que les drones, peuvent utiliser la même technologie pour se déplacer de façon autonome dans les champs de culture. Les véhicules autonomes pourraient potentiellement utiliser les systèmes SLAM visuels pour cartographier et comprendre le monde qui les entoure.

1.7.3.3 Structure from motion (SFM)

La fin des années 1980 a également vu le développement de techniques efficaces de structure à partir du mouvement, qui visent à reconstruire simultanément la structure inconnue de la scène 3D et les positions et orientations de la caméra à partir d'un ensemble de correspondances de caractéristiques. [11]

La structure à partir du mouvement (SfM) est le processus d'estimation de la structure tridimensionnelle d'une scène à partir d'un ensemble d'images bidimensionnelles. La SfM est utilisée dans de nombreuses applications, telles que la numérisation 3D, la réalité augmentée et la localisation et la cartographie visuelles simultanées (vSLAM).

La SfM peut être calculée de nombreuses façons différentes. La façon dont vous abordez le problème dépend de différents facteurs, tels que le nombre et le type de caméras utilisées et l'ordre des images. Si les images sont prises avec une seule caméra calibrée, la structure 3D et le mouvement de la caméra ne peuvent être récupérés qu'à l'échelle. L'échelle signifie que vous pouvez modifier la structure et l'amplitude du mouvement de la caméra tout en conservant les observations.

Les méthodes SfM traitent hors ligne un ensemble d'images non ordonnées sans contrainte temporelle pour estimer la pose de la caméra en utilisant des caractéristiques correspondantes parmi des paires d'images. [12]

La SfM et la SLAM construisent toutes deux des cartes 3D de l'environnement et estiment la pose de la caméra pour chaque image. Cependant, elles ne sont applicables que lorsque nous disposons d'une collection d'images qui se chevauchent afin de trianguler des points dans l'espace grâce à la géométrie multi-vues.

1.7.4 La localisation basée vision LBV

La localisation basée vision (LBV) est la tâche de vision par ordinateur qui consiste à retrouver la position d'une caméra à partir d'une image d'interrogation capturée par la caméra. La LBV a fait l'objet d'une attention accrue en tant que sujet de recherche en raison de la forte dépendance d'applications importantes, telles que la réalité augmentée, sur la connaissance de la localisation. [13]

Selon [14] les méthodes LBV sont divisées en plusieurs catégories suivantes :

1.7.4.1 Localisation basée sur l'image

La méthode de localisation basée sur l'image récupère principalement la photo la plus similaire à l'image de la requête dans la base de données d'images et l'utilise pour estimer la pose. Elles sont couramment utilisées pour la reconnaissance de la position et la détection de la fermeture de boucles.

1.7.4.2 Localisation basée sur un modèle d'apprentissage

La localisation basée sur un modèle d'apprentissage consiste à apprendre et à former un modèle de régression, la structure du réseau CNN, etc. et à utiliser ces modèles pour obtenir

directement la pose de l'image correspondante à travers les données de l'image d'interrogation d'entrée.

Il existe également de nombreuses branches dans la localisation basée sur le modèle d'apprentissage.

1.7.4.2 Localisation basée sur la structure 3D

L'idée principale est de construire un modèle de nuage de points 3D à l'aide de l'algorithme SFM, puis d'utiliser la méthode du descripteur de caractéristiques pour extraire les caractéristiques 2D de l'image. Les caractéristiques généralement extraites sont des caractéristiques locales telles que SIFT (Scale Invariant Feature Transform), LIFT (Learned Invariant Feature Transform), puis nous établissons l'association entre les caractéristiques 2D et le modèle 3D.

Ensuite, le résultat de la correspondance 2D-3D est placé dans un algorithme RANSAC (Random Sample Consensus). Le cycle et le solveur de pose n-point est utilisé pour estimer la pose dans cette recirculation.

1.8 Odométrie visuelle

Le terme "odométrie visuelle" a été inventé par Nistér [15]. Ce terme a été choisi parce que la localisation basée sur la vision est similaire à l'odométrie des roues dans la mesure où elle estime de manière incrémentale le mouvement d'un véhicule en intégrant le nombre de tours de ses roues dans le temps. De la même manière, la VO intègre les déplacements des pixels entre les images dans le temps. [2]

La VO est une technique d'odométrie alternative et peu coûteuse qui est plus précise que les techniques conventionnelles, telles que le GPS, l'INS, l'odométrie par roue et les systèmes de localisation par sonar. Elle est utilisée dans de nombreuses applications, telles que les robots mobiles, les voitures à conduite autonome et les drones.

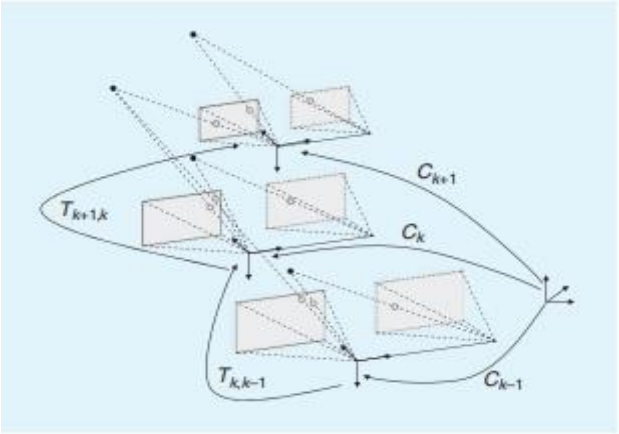

Figure 1-3 : Une illustration du problème de l'odométrie visuelle [16]

1.8.1 Types d'odométrie visuelle

1.8.1.1 Odométrie visuelle monoculaire

Une seule caméra est utilisée pour capturer le mouvement. Généralement, une méthode d'estimation de la pose relative en cinq points est utilisée pour estimer le mouvement, le mouvement étant calculé sur une échelle relative. Typiquement utilisé dans les méthodes hybrides où d'autres données de capteur sont également disponibles.

Le principal défi de l'odométrie visuelle monoculaire est de minimiser la dérive de la trajectoire ainsi que la distorsion de la carte sur de très longs trajets.

1.8.1.2 Odométrie visuelle stéréo

Une paire de caméras stéréo calibrées est utilisée pour calculer la profondeur des caractéristiques entre les images à différents moments. La sortie calculée est le mouvement réel (à l'échelle). Si seules les caractéristiques éloignées sont suivies, cela dégénère en cas de vision monoculaire.

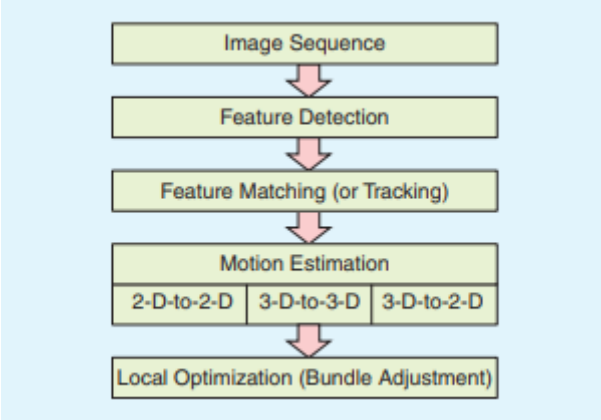

Figure 1-4: Un schéma fonctionnel montrant les principaux composants d'un système VO [16]

1.9 Les approches de localisation visuelle par réseaux de neurones

Ces dernières années, les approches de localisation visuelle par régression de pose absolue (APR) sont devenues populaires. Plutôt que d'utiliser l'apprentissage automatique uniquement pour certaines parties du pipeline de localisation, par exemple les caractéristiques locales, le filtrage des aberrations ou la régression des coordonnées de la scène, ces approches visent à apprendre le pipeline de localisation complet. Étant donné un ensemble d'images d'entraînement et de leurs poses correspondantes.

1.9.1 Régression de la pose absolue de la caméra (APR)

Ces approches entraînent les CNN à régresser la pose de la caméra d'une image d'entrée, représentant ainsi la scène implicitement par les poids des réseaux. Elles suivent toutes le même processus : Les caractéristiques sont extraites à l'aide d'un réseau de base, par exemple VGG ou ResNet, qui sont ensuite intégrées dans un espace à haute dimension. [17]

1.9.2 Régression de la pose relative de la caméra (RPR)

Ces approches prédisent la pose d'une image de test par rapport à une ou plusieurs images d'entraînement au lieu d'utiliser les coordonnées absolues de la scène.

La prédiction est à nouveau gérée par un CNN entraîné pour la régression. Les images d'entraînement pertinentes peuvent être trouvées en utilisant une étape de recherche d'images explicite ou en représentant implicitement les images dans le CNN. [17]

Les techniques APR entraîne des réseaux neuronaux à convolution (CNN) pour régresser directement la pose de la caméra à partir d'une image.

1.10 Conclusion

Dans ce chapitre, nous avons défini la localisation. Cette dernière est basée sur la capacité des robots à se localiser et à construire un modèle de leurs environnements. Ensuite, nous avons montré les différents types de capteurs (proprioceptifs et extéroceptifs) qui permettent aux différents robots de réagir à leur environnement, puis nous avons présenté les différents types de localisation. Enfin, on a présenté les différentes techniques de localisation et les approches de localisation visuelle par réseaux de neurones. Le chapitre suivant présente les concepts de base de l'apprentissage profond et les réseaux de neurones convolutifs (CNN).

Chapitre 2

Réseaux Neurones

2.1 Introduction

Le domaine du Deep Learning a pris une grande importance au cours des dernières années, car il permet de traiter de manière efficace des ensembles de données massifs et de rendre les systèmes informatiques suffisamment performants pour résoudre les difficultés de traitement.

Le Deep Learning est une branche de l'intelligence artificielle qui est complètement basée sur les réseaux neuronaux artificiels. Comme les réseaux neuronaux sont destinés à imiter le cerveau humain

Il est courant de développer un modèle de réseau neuronal Deep Learning pour un problème de régression ou de classification, mais pour certaines tâches de modélisation prédictive.

Les algorithmes de régression et de classification sont connus sous le nom d'algorithmes d'apprentissage supervisé et sont utilisés pour prédire et fonctionner avec des ensembles de données étiquetées.

Dans ce chapitre, nous allons présenter tout d'abord les notions en relation avec l'apprentissage profond (Deep Learning) et les réseaux neurones artificiels et enfin nous allons aborder les notions de Transfer Learning.

2.2 Définitions

2.2.1 L'intelligence Artificielle (Artificial intelligence)

L'intelligence artificielle (IA) consiste à créer des machines qui peuvent penser intelligemment comme des humains et imiter leurs actions. Le terme peut également s'appliquer à toute machine qui présente des caractéristiques associées à l'esprit humain, comme l'apprentissage et la résolution de problèmes.

2.2.2 L'apprentissage Automatique (Machine Learning)

L'apprentissage automatique est une façon d'effectuer les mêmes tâches en utilisant des algorithmes pour extraire des informations des données dans une programmation conventionnelle. Et en utilisant à la fois les données et les résultats, nous extrayons les règles qui définissent leurs relations. En d'autres termes, nous aurons des logiciels de plus en plus précis pour prédire des résultats sans avoir été expressément conçus pour cela.

Afin d'anticiper les valeurs futures, les algorithmes d'apprentissage automatique utilisent des données historiques en entrée.

2.2.3 L'apprentissage profond (Deep Learning)

L'apprentissage profond est un nouveau concept apparu au cours des années 2000. Il consiste également à utiliser les données et les résultats pour extraire les règles, qui définissent leurs relations. Donc l'apprentissage profond est encore une façon de faire cela, mais en utilisant un algorithme spécifique appelé un réseau neuronal.

Pour résumer, l'apprentissage profond est un domaine de l'apprentissage automatique qui concerne les réseaux de neuronaux artificiels. Qu'ils sont des algorithmes inspirés de la structure et du fonctionnement du cerveau humain.

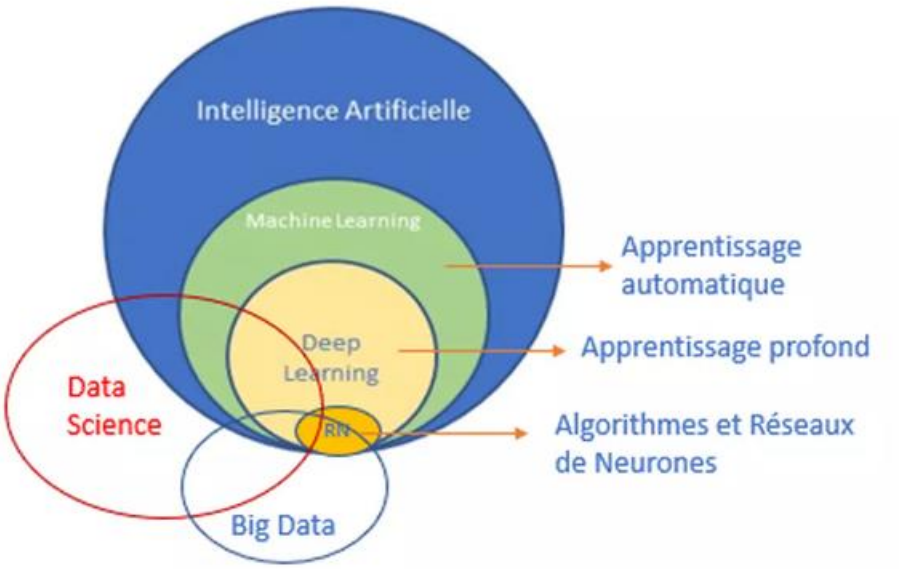

Figure 2-1 : Interactions entre Intelligence Artificielle, Machine Learning et Deep Learning [18]

2.3 Présentation de l'apprentissage profond

2.3.1 Historique de l'apprentissage profond

Depuis les années 2000, l'apprentissage en profondeur et ses succès technologiques ont donné lieu à un intérêt renouvelé pour le connexionnisme. Cette progression pendulaire concerne principalement le monde informatique, alors que du côté statistique, les développements sont plus doux.

Année	Contribution
1873	Alexander Bain (1873) a proposé le concept de réseau neuronal artificiel
1943	On entend parler du premier neurone artificiel en 1943 quand Warren Mcculloch et Walter Pitt ont publié leur premier modèle mathématique et informatisé du neurone artificiel intitule « A Logical Calculus of Ideas Immanent in Nervous Activity » [19]
1949	La règle de Hebb est un postulat proposé par Donald Hebb en 1949 [20]. Il s'agit d'une règle d'apprentissage qui décrit comment les activités neuronales influencent la connexion entre les neurones.
1958	Que le Perceptron fut inventé par Frank Rosenblatt au laboratoire aéronautique de Cornell. En se basant sur les premiers concepts de neurones artificiels, il proposa la ” règle d’apprentissage du Perceptron “. [21]
1974	Paul Werbos introduction de la retro propagation
1996	Robert Werbos introduction propose une méthode de régression « méthode du Lasso » pour traiter un grand nombre de variables d’entrée
1998	Réseau de Neurone convolutif Lenet (modèle CNN) a été conçu par Yan Lucan.
2006	Les débuts de Deep Learning, Geoffrey Hinton optimise le fonctionnement des Réseaux neuronaux multicouches (A Fast Learning Algorithm For Deep Belif Nets)
2007	La complétion de matrices sous les feux de la rampe
2010	Le challenge d’ImageNet
2014	La reconnaissance de visage avec DeepFace (Facebook)
2017	Alpha GO développé par l’entreprise britannique Deep Mind rachetée par Google en 2014

Tableau 2-1 : Les étapes majeures d’apprentissage profond

2.3.2 Pourquoi l'apprentissage profond

L'apprentissage profond est très important parce qu'il peut gérer la complexité, alors que les algorithmes typiques d'apprentissage ne le peuvent pas.

Depuis des décennies, l'apprentissage automatique est utilisé pour classer les images et les textes, mais il existe un seuil de performance. Au-delà de ce seuil, il ne peut pas être performant. En fait, la plupart du temps, les données dont nous disposons sont énormes et les relations entre les données sont très complexes. Dans cette situation, seul l'apprentissage profond nous permet de traiter efficacement de telles données

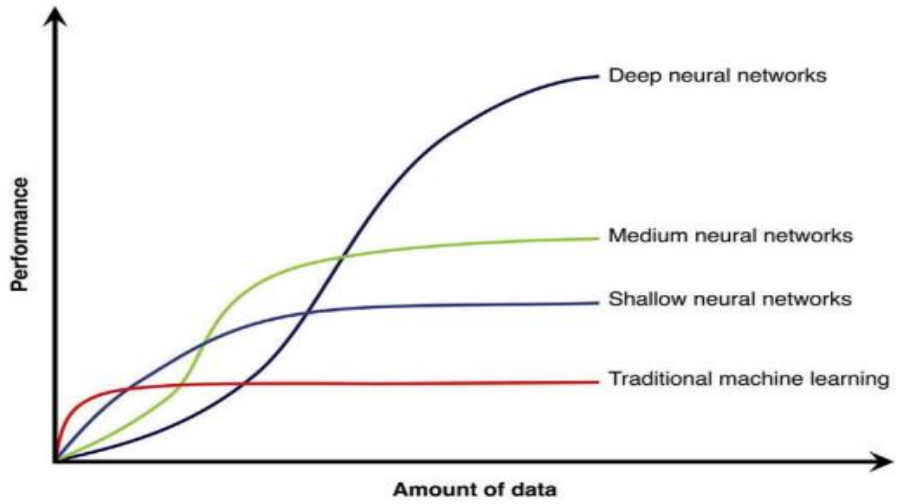

Figure 2-2 : La différence de performance entre le Deep Learning et la plupart des algorithmes de ML en fonction de la quantité de données [22]

2.3.3 Domaines d'application du Deep Learning

2.3.3.1 Le domaine de la vision

Cela implique la reconnaissance faciale, la reconnaissance des empreintes digitales, la télédétection, la détection des maladies dans l'imagerie médicale avec une précision proche de 100% aujourd'hui.

Les domaines de la médecine où les réseaux de neurones bénéficient le plus des résultats obtenus en imagerie médicale, comme la détection de plus en plus précise des maladies (l'IA de Google peut détecter le cancer du sein mieux que les humains).

2.3.3.2 Traitement du langage

L'un des domaines d'application Deep Learning les plus impressionnants, ces résultats proviennent donc de la reconnaissance vocale, des agents virtuels, des robots conversationnels, de la traduction automatique, etc.

2.3.3.1 La défense

La puissance militaire d'une nation détermine sa position sur la scène internationale. Les systèmes militaires équipés de l'IA et du Deep Learning peuvent traiter efficacement des

volumes de données plus importants, une infrastructure de cette nature constitue un élément essentiel de la guerre moderne en raison de l'augmentation des capacités de calcul et de prise de décision.

Le deep Learning peut être appliqué à d'autres exemples d'applications du deep Learning dans l'armée, notamment la reconnaissance de cibles, la simulation et l'entraînement au combat, les plateformes de guerre, la logistique et le transport, la surveillance des menaces en matière de cybersécurité, etc.

2.3.3.4 Le marketing

Le deep Learning peut décoder des données complexes non structurées cet attribut est utile pour obtenir des informations sur les clients qui sont indispensables pour créer un plan de vente et de marketing efficace.

Les spécialistes du marketing peuvent accéder à des informations provenant de l'analyse vidéo, des images, de la reconnaissance faciale, de la reconnaissance vocale et de l'analyse de texte, ils peuvent analyser les retours des clients et évaluer leurs attentes en temps réel.

2.3.3.5 La cybersécurité

Les derniers modèles de cybersécurité basées sur le deep Learning étaient capables d'atteindre un taux de détection plus élevé et un taux de faux positifs plus faible pour des logiciels malveillants inconnus jusqu'alors, par rapport aux solutions traditionnelles.

2.3.4 Types d’apprentissage profond :

Le Deep Learning utilise l'apprentissage supervisé dans des situations telles que la classification d'images ou la détection d'objets, car le réseau est utilisé pour prédire une étiquette ou un nombre (l'entrée et la sortie sont toutes deux connues). Comme les étiquettes d'image sont connues, le réseau sert à réduire le taux d'erreur, il est donc "supervisé".

De plus, les réseaux de neurones peuvent aussi être utilisés pour regrouper des images basées sur des similarités. Les caractéristiques peuvent être extraites en utilisant un réseau neuronal, puis une méthodologie non supervisée peut-être déployée

Un réseau neuronal peut prendre la forme d'un réseau de neurone profond semi-supervisé.

Les auto-encodeuses permettent de compresser et de reconstruire des images. Ces auto-encodeuses sont considérées comme des réseaux neuronaux auto-apprenants.

Enfin, il est possible d'utiliser l'apprentissage par renforcement avec des réseaux de neurones. C'est la méthodologie qui a permis au DeepMind de remporter le jeu de Go.

Par conséquent, le Deep Learning peut être supervisé, non supervisé, Semi-supervisé, auto-supervisé ou par renforcement, et cela dépend principalement de la façon dont le réseau neuronal est utilisé.

2.3.4.1 L'apprentissage supervisé

L'apprentissage supervisé est une forme d'apprentissage automatique ou l'algorithme est guidé par un ensemble d'observations décrit par des variables explicatives ainsi que la variable cible, la réponse l'étiquette. L'algorithme d'apprentissage supervisé abordera une fonction de prédiction qui permet de mapper les données depuis les observations jusqu'à la réponse.

2.3.4.2 L'apprentissage non supervisé

L'apprentissage non supervisé correspond au cas où les observations du jeu de données ne sont pas étiquetées, elle ne dispose pas de leur variable de sortie. L'algorithme est livré à lui-même et va apprendre les structures et les relations caractérisant le jeu de données.

2.3.4.3 L'apprentissage semi-supervisé

L'apprentissage semi-supervisé est un hybride entre l'apprentissage supervisé et l'apprentissage non supervisé. Il utilise à la fois des données étiquetées et non étiquetées pendant la formation et il est utilisé parce que l'obtention de données étiquetées est difficile à bien des égards, mais il peut produire une amélioration considérable de la précision de l'apprentissage.

L'acquisition de données non étiquetées étant relativement peu coûteuse, l'ajout d'une petite partie d'une étiquette coûteuse à ces données est globalement peu coûteux et produit d'excellents résultats par rapport à des données non étiquetées uniquement.

2.3.4.4 L'apprentissage par renforcement

L'apprentissage par renforcement est un domaine de l'apprentissage automatique qui s'intéresse à la manière dont les agents logiciels doivent entreprendre des actions dans un environnement afin de maximiser une certaine notion de récompense cumulative. Il s'agit de l'un des trois paradigmes de base de l'apprentissage automatique, avec l'apprentissage supervisé et l'apprentissage non supervisé. Il se distingue de l'apprentissage supervisé par le fait qu'il n'est pas nécessaire de présenter des paires entrée-sortie étiquetées et que les actions sous-optimales ne doivent pas être explicitement corrigées.

Au lieu de cela, l'accent est mis sur l'adoption d'une action appropriée pour maximiser la récompense dans une situation particulière. L'apprentissage par renforcement consiste donc à prendre des décisions de manière séquentielle.

2.4 Réseaux neuronaux artificiels (RNA)

Les réseaux neuronaux artificiels (RNA) sont des systèmes de traitement informatique qui s'inspirent fortement du fonctionnement des systèmes nerveux biologiques (tels que le cerveau humain). Les RNA sont principalement composés d'un grand nombre de nœuds de calcul interconnectés (appelés neurones), qui travaillent de manière distribuée pour apprendre collectivement des données d'entrée afin d'optimiser la sortie finale. [23]

2.4.1 Bio-inspiration des Réseaux Neurones

Le cerveau est en fait composé d'un grand nombre de neurones (Nombre de neurones dans le cerveau humain : environ 100 milliards), chacun d'entre eux étant étroitement relié par des milliers de connexions (Nombre moyen de connexions par neurone : 10 000).

Chaque neurone est une cellule spéciale qui peut générer, transmettre et recevoir des signaux électrochimiques. Comme toutes les cellules vivantes, les neurones ont un processus qui transporte les informations vers le péricaryon, les neurones (dendrites), et envoie les informations collectées par les neurones (axones). Les axones d'une cellule sont reliés par des synapses aux dendrites d'une autre cellule. Lorsqu'un neurone est activé, il envoie un signal électrochimique à l'axone. Cette impulsion peut traverser les synapses et atteindre des milliers d'autres neurones, signalant l'ensemble du système nerveux (c'est-à-dire le cerveau biologique). Les neurones n'émettent des impulsions que lorsque le signal envoyé des dendrites au corps cellulaire dépasse un seuil spécifique appelé seuil de déclenchement. [24]

2.4.2 Neurone biologique et Neurone Artificiel

Un neurone formel, ou neurone, c’est un modèle mathématique simplifié d'un neurone biologique considère comme une fonction algébrique non linéaire et bornée, dont la valeur dépend de paramètres appelés coefficients ou poids. Les variables de cette fonction sont habituellement appelées « entrées » du neurone et la valeur de la fonction est appelée « sortie ». Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de programme informatique.

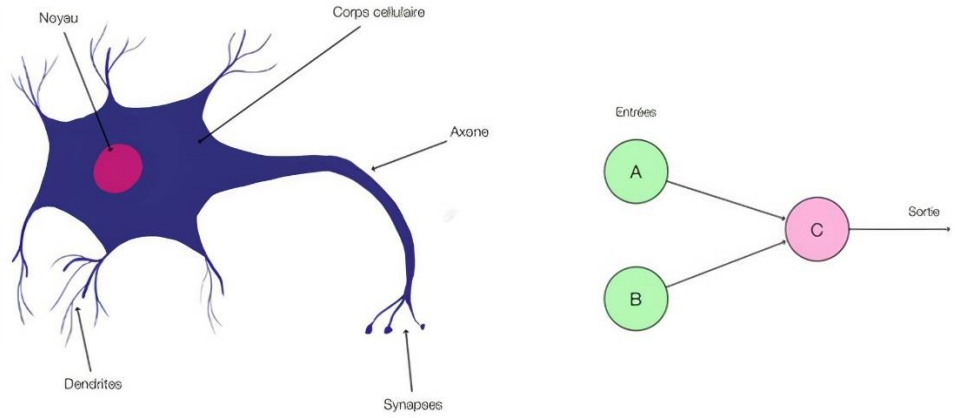

Figure 2-3 : Neurone biologique et premier neurone formel [25]

2.4.3 Perceptron

Les réseaux de neurones formels de Mcculloch et Pitt se sont figés leur coefficient synaptique sont constants un tel réseau ne peut apprendre. On considère que le perceptron est le 1er réseau de neurones artificiels évolutif, c'est à dire capables d'apprentissage.

Le perceptron avait pour but de reconnaître des lettres de l'alphabet avec des cellules photoélectriques comme capteur.

En réalité, le Perceptron est une fonction mathématique. Les données d'entrée (x) sont multipliées par les coefficients de poids (w). Le résultat produit est une valeur numérique. Cette valeur peut être positive ou négative. Le neurone artificiel s'active si la valeur est positive. Il ne s'active donc que si le poids calculé des données d'entrée dépasse un certain seuil.

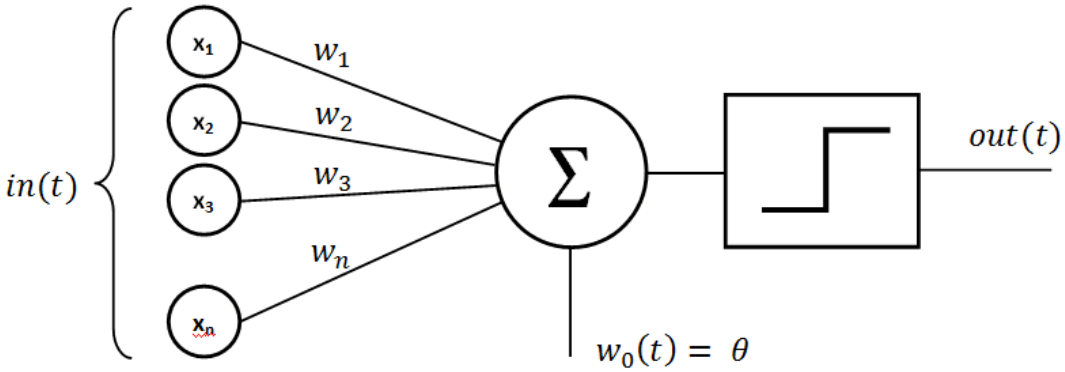

Figure 2-4 : Réseau monocouche [26]

2.4.4 Les perceptrons multicouches MLP

Les perceptrons multicouches sont des réseaux neuronaux feed-forward avec couches intermédiaires, chaque neurone d'une couche est relié à l'ensemble des neurones de la couche

suivante et a pour objectif de classer des données plus complexes que celles classées par un perceptron.

Pour cela, le perceptron multicouche observe chacune des données qu'il possède et met à jour chaque poids de chaque neurone de chaque couche de son réseau afin de classifier au mieux cette base de données. L'algorithme que les perceptrons multicouches utilisent pour mettre à jour leurs poids s'appelle la rétropropagation du gradient de l'erreur.

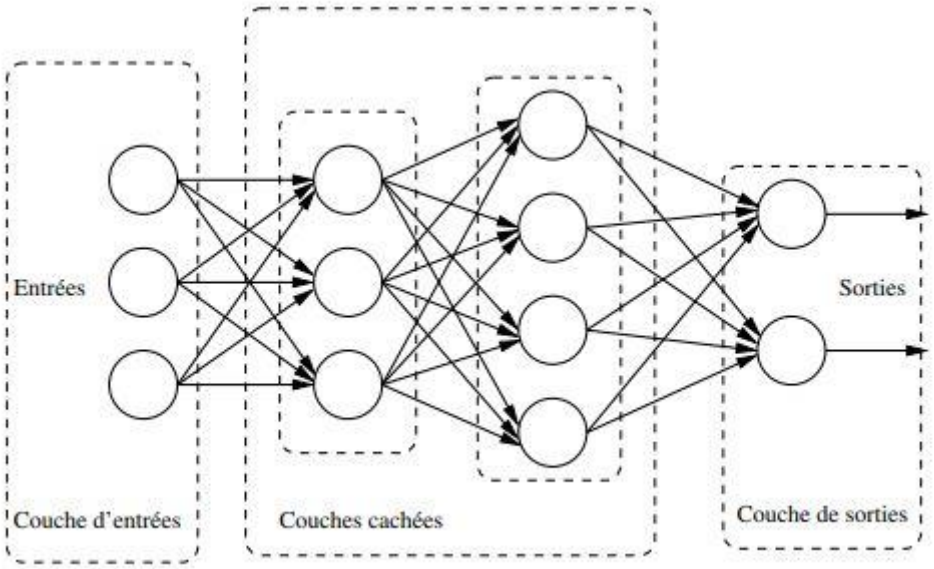

Figure 2-5 : Le perceptron multicouches [27]

2.4.5 Fonction d'activation

La fonction d'activation, ou fonction de transfert, est une fonction qui doit renvoyer un réel proche de 1 quand les "bonnes" informations d'entrée sont données et un réel proche de 0 quand elles sont "mauvaises". On utilise généralement des fonctions à valeurs dans l'intervalle réel [0,1]. Quand le réel est proche de 1, on dit que l'unité (le neurone) est active alors que quand le réel est proche de 0, on dit que l'unité est inactive. [28]

2.4.5.1 Les différentes fonctions d'activation

- **Rectified Linear Unit (Relu) :**

La fonction Rectified Linear Unit (Relu) est la fonction d'activation la plus simple et la plus utilisée.

Elle donne x si x est supérieur à 0, 0 sinon. Autrement dit, c'est le maximum entre x et 0 :

Fonction Relu(x) = max (x, 0)

- **Sigmoïde :**

La fonction Sigmoïde donne une valeur entre 0 et 1, une probabilité. Elle est donc très utilisée pour les classifications binaires, lorsqu'un modèle doit déterminer seulement deux labels.

Fonction_ Sigmoïde(x) = 1 / (1 + exp(-x))

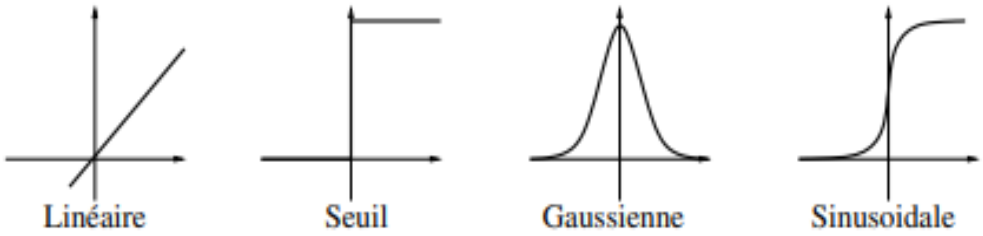

Figure 2-6 : Les fonctions d'activation usuelles [27]

2.4.6 Classification des Réseaux Neurones

Il existe deux grandes catégories d'architectures de réseau, en fonction du type de connexions entre les neurones : les "réseaux neuronaux non bouclés ou feed forward " et les "réseaux neuronaux boucle ou récurrents".

S'il n'y a pas de "rétroaction" des sorties des neurones vers les entrées du réseau, celui-ci est appelé "réseau de neurones à action directe". Sinon, s'il existe une telle rétroaction, c'est-à-dire une connexion synaptique des sorties vers les entrées (soit leurs propres entrées, soit les entrées d'autres neurones), le réseau est appelé " réseau neuronal récurrent ". [29]

2.4.6.1 Les réseaux de neurones feed-forward

Un réseau neuronal statique (FEED FORWARD) est généralement considéré, dans sa forme la plus simple, comme un perceptron à une seule couche « Le perceptron mono-couche ». Dans ce modèle, une série d'entrées entre dans la couche et est multipliée par les poids. Chaque valeur est ensuite additionnée pour obtenir une somme des valeurs d'entrée pondérées. Si la somme des valeurs est supérieure à un seuil spécifique, généralement fixé à zéro, la valeur produite est souvent 1, tandis que si la somme est inférieure au seuil, la valeur de sortie est -1. Le perceptron mono-couche est un modèle important de réseaux neuronaux statiques et est souvent utilisé dans les tâches de classification.

2.4.6.2 Les réseaux de neurones récurrents

Un réseau neuronal récurrent (RNN) est un type de réseau neuronal artificiel qui utilise des données séquentielles ou des données de séries temporelles.

Les réseaux de neurones récurrents utilisent des données de formation pour apprendre. Ils se distinguent par leur "mémoire", car ils utilisent les informations des entrées précédentes pour influencer l'entrée et la sortie actuelles. Alors que les réseaux neuronaux profonds traditionnels supposent que les entrées et les sorties sont indépendantes les unes des autres, la sortie des réseaux neuronaux récurrents dépens des éléments antérieurs de la séquence. Alors que les événements futurs seraient également utiles pour déterminer la sortie d'une séquence donnée.

Le RNN fonctionne sur le principe de l'enregistrement de la sortie d'une couche particulière et de sa réinjection dans l'entrée afin de prédire la sortie de la couche.

La figure montre comment convertir un réseau de neurones à action directe en un réseau de neurones récurrent :

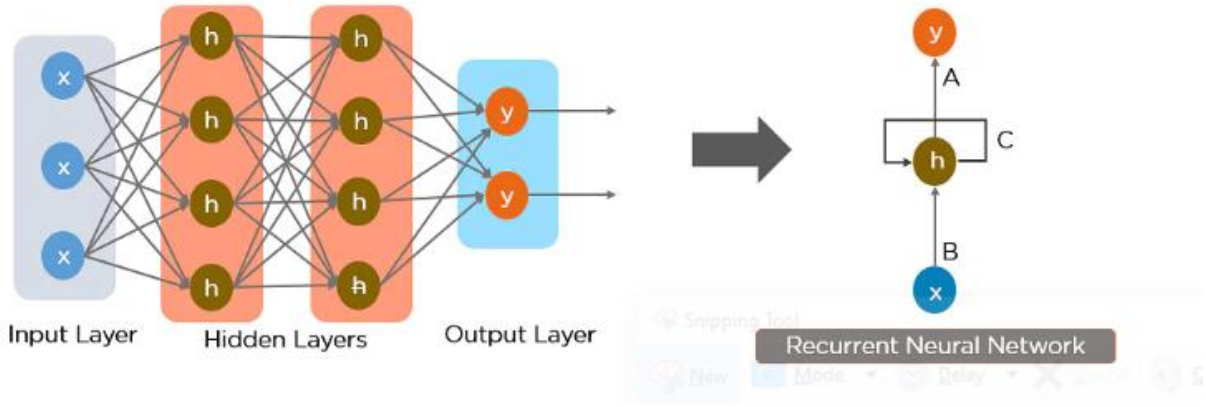

Figure 2-7 : Réseau neuronal récurrent simple [30]

2.5 Réseaux de Neurones convolutifs CNN

2.5.1 Définition

Les réseaux neuronaux convolutifs, aussi appelés CNN ou ConvNets, sont des modèles de réseau neuronal artificiel à action directe. Il est utilisé en deep Learning pour évaluer les informations visuelles. Ces réseaux sont capables de traiter un grand nombre de tâches impliquant des images, des sons, des textes, des vidéos et d'autres médias.

2.5.2 Le rôle de différentes couches

L'architecture d'un CNN se compose de deux parties principales :

Un outil de convolution qui sépare et identifie les différentes caractéristiques de l'image à des fins d'analyse dans un processus appelé extraction de caractéristiques.

Une couche entièrement connectée qui utilise la sortie du processus de convolution et prédit la classe de l'image en fonction des caractéristiques extraites lors des étapes précédentes. Généralement, on distingue trois types de couches pour un réseau neuronal convolutif : la couche convolutive, la couche de pooling et la couche entièrement connectée fully-connected.

2.5.2.1 La couche convolutionnelles

La couche convolutive est le composant clé des réseaux de neurones convolutifs et est toujours au moins leur première couche. Son but est de détecter la présence d'un ensemble de caractéristiques dans les images reçues en entrée. Cela se fait par filtrage convolutif : le principe est de "glisser" une fenêtre représentant la caractéristique sur l'image, et de calculer le produit de convolution entre la caractéristique et chaque portion de l'image numérisée.

La couche convolutive reçoit donc plusieurs images en entrée et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux caractéristiques que nous voulons trouver dans les images.

Nous obtenons pour chaque paire (image, filtre) une carte de caractéristiques, qui nous indique où se trouvent les caractéristiques dans l'image.

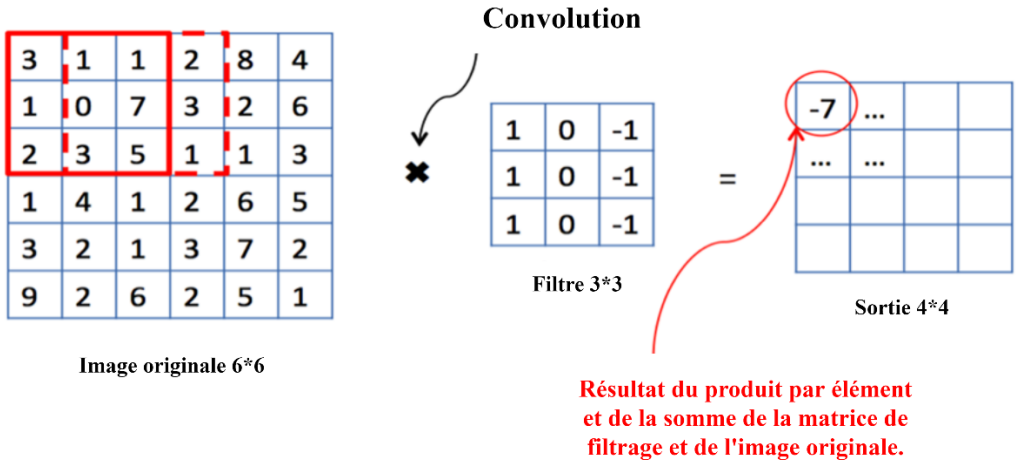

Figure 2-8 : Opération d'une convolution sur image de 6*6 pixels [31]

2.5.2.2 La couche de Pooling

Dans la plupart des cas, une couche convolutive est suivie d'une couche de pooling. L'objectif principal de cette couche est de diminuer la taille de la carte de caractéristiques convolutionnelle afin de réduire les coûts de calcul. Ceci est réalisé en diminuant les connexions entre les couches et en opérant indépendamment sur chaque carte de caractéristiques. Selon la méthode utilisée, il existe plusieurs types d'opérations de pooling.

- **Maxpooling** : Il choisit l'élément le plus significatif de la carte de caractéristiques. Les éléments significatifs de la carte de caractéristiques sont stockés dans la couche résultante de maxpooling. Il s'agit de la méthode la plus populaire car elle produit les meilleurs résultats.
- **Average pooling** : Il s'agit de calculer la moyenne pour chaque région de la carte de caractéristiques. La somme totale des éléments de la section prédéfinie est calculée dans Sum Pooling**.**

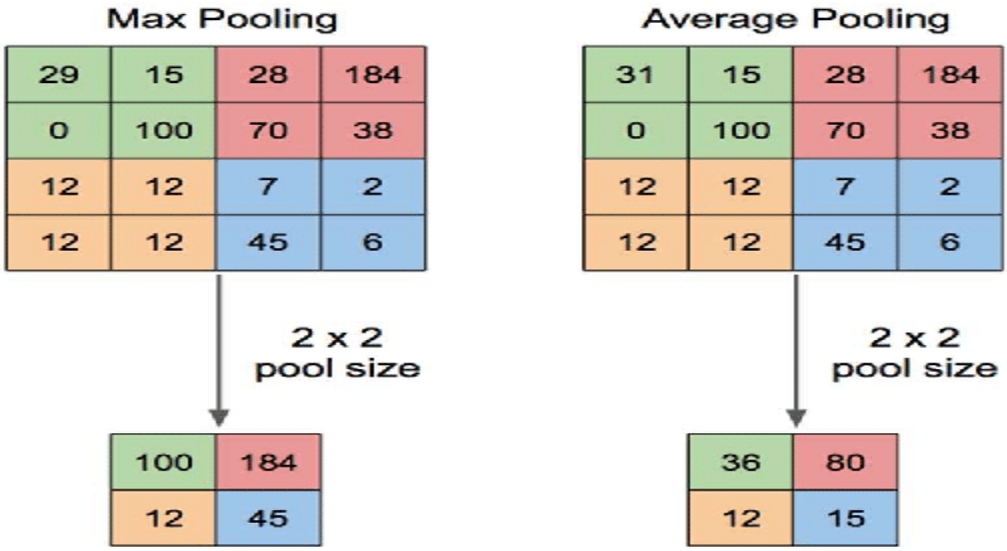

Figure 2-9 : Illustration du Max Pooling et du Average Pooling [32]

2.5.2.3 Unités linéaires rectifiées (ReLU)

La fonction d'activation linéaire rectifiée, ou ReLU, est une fonction linéaire par morceaux qui, lorsque l'entrée est positive, elle produit directement l'entrée ; sinon, elle produit zéro comme un modèle qui l'utilise est plus rapide à former et produit généralement de meilleures performances, elle est devenue la fonction d'activation par défaut pour de nombreux types de réseaux neuronaux.

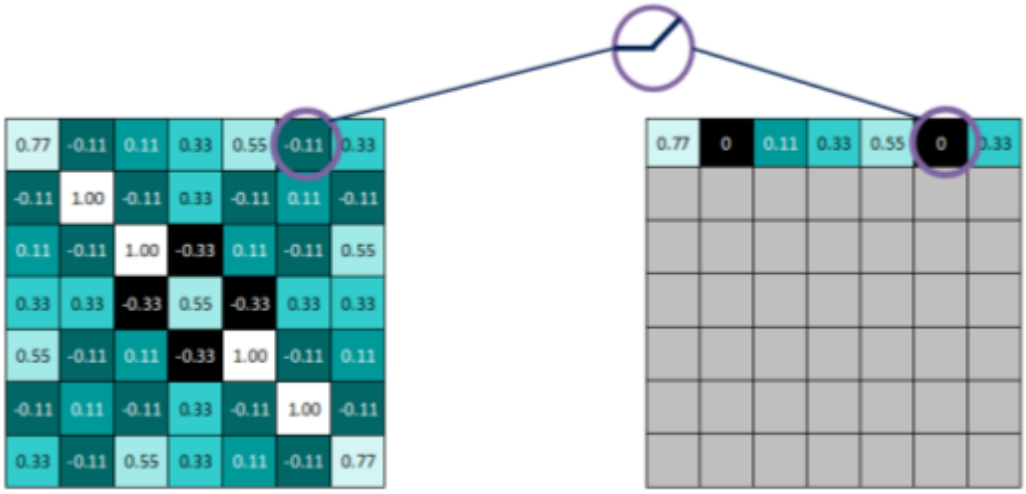

Figure 2-10 : Unités Rectifié Linéaire (ReLU) [25]

2.5.2.4 La couche entièrement connectée (FC)

La couche entièrement connectée (FC) se compose des poids et des biais avec les neurones et sert à connecter les neurones entre deux couches différentes. Ces couches sont généralement placées avant la couche de sortie et forment les dernières couches d'une architecture CNN.

Dans cette couche, l'image d'entrée des couches précédentes est aplatie et transmise à la couche FC. Le vecteur aplati passe ensuite par quelques couches FC supplémentaires où les opérations de fonctions mathématiques sont effectuées.

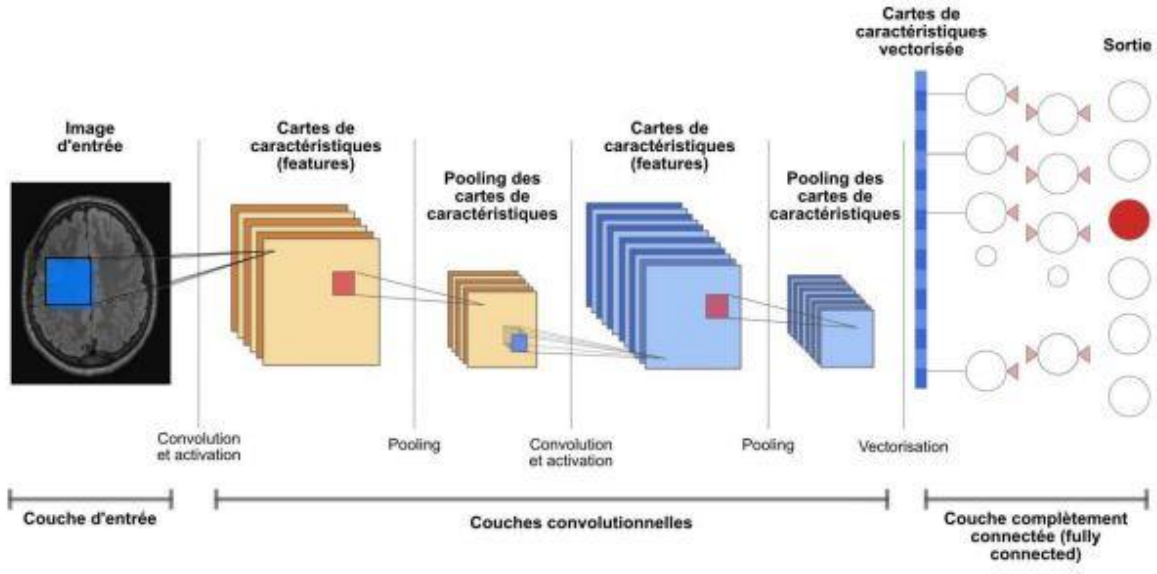

Figure 2-11 : Schéma général de principe de fonctionnement d'un réseau de neurones convolutifs [25]

2.5.3 L'entrainement d'un Réseau de Neurone

Le processus d'ajustement de la valeur des poids est défini comme la "entrainement" du réseau neuronal.

Tout d'abord, le CNN démarre avec des poids aléatoires. Pendant la formation du CNN, le réseau neuronal est alimenté par un grand ensemble de données d'images étiquetées avec leurs étiquettes de classe correspondantes. Le réseau CNN traite chaque image avec ses valeurs attribuées de manière aléatoire, puis effectue des comparaisons avec l'étiquette de classe de l'image d'entrée.

Si la sortie ne correspond pas à l'étiquette de classe, le CNN effectue un petit ajustement des poids de ses neurones afin que la sortie corresponde correctement à l'étiquette de classe.

Les corrections de la valeur des poids sont effectuées par une technique connue sous le nom de "rétropropagation".

La rétropropagation optimise le processus de réglage et facilite les ajustements pour une meilleure précision. Chaque exécution de l'entrainement de l'ensemble de données d'image est appelée une "epoch".

Le CNN passe par plusieurs séries d'epochs au cours du processus de l'entrainement, en ajustant ses poids en fonction des petites quantités requises.

Après chaque étape d'epoch, le réseau neuronal devient un peu plus précis dans la classification et la prédiction correcte de la classe des images d'entraînement. Au fur et à mesure que le réseau neuronal s'améliore, les ajustements apportés aux poids deviennent de plus en plus petits.

Après avoir formé le CNN, nous utilisons un ensemble de données de test pour vérifier sa précision. L'ensemble de données de test est un ensemble d'images étiquetées qui n'ont pas été incluses dans le processus de l'entrainement. Chaque image est introduite dans le CNN, et la sortie est comparée à l'étiquette de classe réelle de l'image de test. Essentiellement, l'ensemble de données de test évalue la performance de prédiction du CNN.

Si la précision d'un CNN est bonne sur ses données d'apprentissage mais mauvaise sur les données de test, on dit qu'il est "surajusté". Cela se produit en raison de la taille réduite de l'ensemble de données. [33]

2.5.4 Classification par CNN

La classification dans le Deep Learning et les statistiques est une méthode d'apprentissage supervisé dans laquelle le programme informatique apprend à travers les données qui lui sont fournies et réalise de nouvelles observations ou classifications.

La classification est un processus de catégorisation d'un ensemble donné de données en plusieurs catégories. Elle peut être effectuée sur des données structurées ou non structurées. Le processus commence par la prédiction de la classe des points de données donnés. Les classes sont souvent appelées cibles, étiquettes ou catégories.

La modélisation prédictive de classification est la partie de l'approximation de la fonction de correspondance entre les variables d'entrée et les variables de sortie discrète. Le but principal est d'identifier dans quelle classe/catégorie les nouvelles données vont se classer.

2.5.4.1 Les termes de classification dans DL

-Classificateur : Il s'agit d'un algorithme qui est utilisé pour faire correspondre les données d'entrée à une classe particulière.

-Modèle de classification : Le modèle prédit ou détermine une situation à partir des données d'entrée fournies pour l'entraînement, il prédit la classe ou la catégorie des données.

-Caractéristique : Une caractéristique est une propriété spécifique mesurable du phénomène observé.

-Classification binaire : elle s'agit d'un type de classification avec deux résultats

-Classification multi-classes : elle s'agit d'une classification comportant plus de deux classes. Dans la classification multi-classes, chaque échantillon est affecté à une et une seule étiquette ou cible.

-Classification à étiquettes multiples : elle est un type de classification où chaque échantillon est affecté à un ensemble d'étiquettes ou de cibles.

-Initialiser : Il s'agit d'affecter le classificateur à utiliser pour la classification multi-classes.

-Entraînement du classificateur : Chaque classificateur de scikit learn utilise la méthode fit (X, y) pour ajuster le modèle d'entraînement du train X et de l'étiquette d'entraînement y.

-Prédire la cible : Pour une observation non étiquetée X, la méthode predict(X) renvoie l'étiquette prédite y.

-Évaluer : Il s'agit essentiellement de l'évaluation du modèle, c'est-à-dire du rapport de classification, du score de précision, etc.

2.5.5 Régression par CNN

La régression désigne un ensemble de méthodes permettant de modéliser la relation entre une ou plusieurs variables indépendantes et une variable dépendante. Dans les sciences naturelles et les sciences sociales, l'objectif de la régression est le plus souvent de caractériser la relation entre les entrées et les sorties.

Il existe plusieurs algorithmes pour la régression : Régression linéaire, Régression polynomiale, Régression logistique et Régression quantile etc.

2.5.5.1 Régression Linéaire

Il s'agit d'un algorithme d'apprentissage supervisé dont le but est de prédire des valeurs numériques continues à partir de données d'entrée données. D'un point de vue géométrique, chaque échantillon de données est un point. La régression linéaire tente de trouver les paramètres de la fonction linéaire, afin que la distance entre tous les points et la ligne soit la plus petite possible. L'algorithme utilisé pour la mise à jour des paramètres est appelé descente de gradient. [34]

La fonction qui permet d'estimer les valeurs y d'un échantillon en fonction des éléments d'entrée x s'écrit comme suit :

$$\hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m$$

Où :

- $\hat{y}$ est la sortie estimée (résultat),
- xj est une caractéristique d'entrée,
- θj est le poids de cette caractéristique

2.5.5.2 Régression logistique

La régression logistique est un algorithme de classification par apprentissage supervisé et non pas de régression. Mais cette méthode est considérée comme une méthode de régression, car elle est utilisée pour évaluer la probabilité d'appartenir à une certaine catégorie. Utilisé pour prédire la probabilité d'une variable cible. La nature de la variable cible ou dépendante est binaire, ce qui signifie qu'il n'y a que deux classes possibles.

La régression logistique prédit le résultat d'une variable dépendante catégorique. Par conséquent, le résultat doit être une valeur catégorique ou discrète. Mais au lieu de donner la valeur exacte comme 0 et 1, elle donne les valeurs probabilistes qui se situent entre 0 et 1.

La régression logistique est très similaire à la régression linéaire, sauf en ce qui concerne leur utilisation. La régression linéaire est utilisée pour résoudre les problèmes de régression, alors que la régression logistique est utilisée pour les problèmes de classification. [34]

Avantages/Limites	La régression linéaire	La régression logistique
Avantages	-Facile à comprendre et à interpréter -Efficace pour l'analyse des données.	-Elle est également utilisable pour la classification, en indiquant les probabilités des sorties. -Le modèle logistique peut être actualisé facilement.
Limites	-Elle fournit de faibles performances s'il n'y a pas de relation linéaire. -La majorité des phénomènes réels ne correspondent pas à ce que suppose le modèle linéaire. -Sensible aux données extrêmes	-Les performances sont faibles en cas de seuils de décision multiples ou de seuils de décision non linéaires.

Tableau 2-2 : Les avantages et les limites de la régression linéaire/logistique.

2.5.5.3 Les fonctions de coût de la régression

Une fonction de coût est utilisée pour mesurer à quel point le modèle se trompe dans la recherche d'une relation entre l'entrée et la sortie. Elle indique à quel point le modèle se comporte mal ou prédit mal.

Les tâches de régression traitent des données continues. Les fonctions de coût qui existent pour la régression sont les suivantes,

- **Erreur absolue moyenne : (Mean Absolute Error MAE)**

L'erreur absolue moyenne (MAE) est la différence absolue moyenne entre les valeurs réelles et les valeurs prédites. MAE est plus robuste aux valeurs aberrantes. Cette insensibilité aux

valeurs extrêmes est due au fait qu'elle ne permet pas de pénaliser les erreurs élevées causées par les valeurs extrêmes.

- **Erreur quadratique moyenne :(Mean Squared Error MSE)**

L'erreur quadratique moyenne (MSE) est la différence quadratique moyenne entre les valeurs réelles et prédites. La MSE pénalise les erreurs élevées causées par les valeurs aberrantes en élevant les erreurs au carré. Les algorithmes d'optimisation bénéficient de la pénalisation car elle permet de trouver les valeurs optimales des paramètres.

- **La racine d'erreur quadratique moyenne : (The Root Mean Squared Error RMSE)**

L'erreur quadratique moyenne (RMSE) est la moyenne quadratique de la différence entre les valeurs réelles et prédites. La RMSE peut être utilisée dans les situations où nous voulons pénaliser les erreurs élevées, mais pas aussi fortement que la MSE.

- **La racine d'erreur logarithmique moyenne quadratique : (Root Mean Squared Logarithmic Error RMSLE)**

L'erreur logarithmique moyenne quadratique (RMSLE) est très similaire à la RMSE, mais le logarithme est appliqué avant de calculer la différence entre les valeurs réelles et prédites. Les grandes erreurs et les petites erreurs sont traitées de la même manière. La RMSLE peut être utilisée dans les situations où la cible n'est pas normalisée ou mise à l'échelle.

2.6 L'apprentissage par transfert (Transfer Learning)

Il s'agit d'une approche courante dans le deep Learning, où des modèles pré-entraînés sont utilisés comme point de départ pour les tâches de vision par ordinateur et de traitement du langage naturel, étant donné les vastes ressources en temps et en calcul nécessaires pour développer des modèles de réseaux neuronaux sur ces problèmes et les énormes sauts de compétence qu'ils permettent de réaliser sur des problèmes connexes.

Dans l'apprentissage par transfert, nous pouvons remplacer la dernière couche d'un réseau pré-entraîné et l'adapter à un nouveau problème.

2.6.1 Les différents types de transfer learning

Il existe plusieurs types d'apprentissage par transfert

L'apprentissage par transfert inductif : Dans ce cas, les domaines source et cible sont les mêmes, mais les tâches source et cible sont différentes l'une de l'autre. Les algorithmes tentent d'utiliser les biais inductifs du champ source pour aider à améliorer la tâche cible. Selon que le champ source contient ou non des données étiquetées.

L'apprentissage par transfert non supervisé : Ce scénario est similaire au transfert inductif lui-même, avec un accent sur les tâches non supervisées dans le champ cible. Les champs source et cible sont similaires, mais les tâches sont différentes. Dans ce cas, les données étiquetées ne sont pas disponibles dans l'un ou l'autre des champs.

L'apprentissage par transfert transductif :Dans ce cas, il existe des similarités entre les tâches source et cible, mais les domaines correspondants sont différents. Dans ce cas, le domaine source possède de nombreuses données étiquetées, alors que le domaine cible n'en possède aucune.

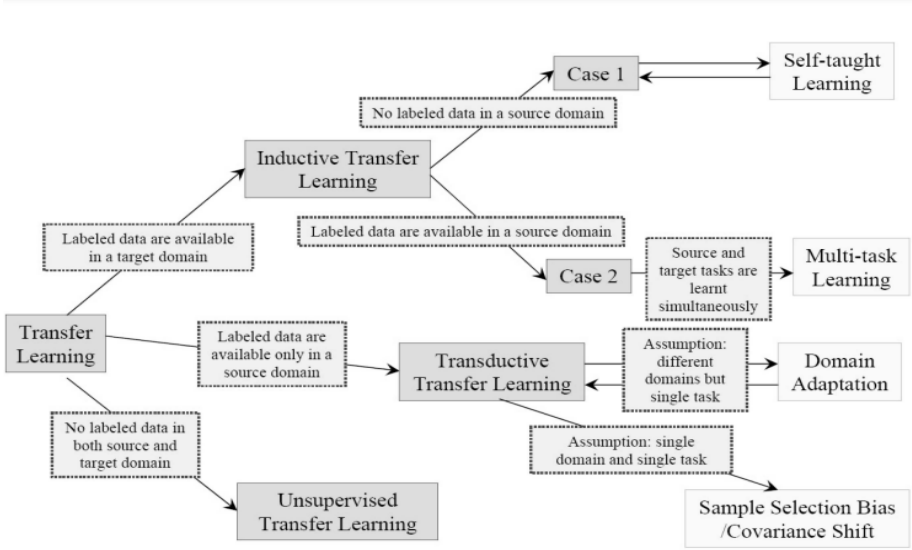

Figure 2-12 : Les types de l'apprentissage par transfert [35]

2.6.2 Les réseaux de neurones convolutifs pré-entraînés populaires

2.6.2.1 AlexNet

Cette architecture est un réseau de neurones convolutif classique. Il se compose de convolutions, de max pooling et de couches denses comme éléments de base. Des convolutions groupées sont utilisées afin d'adapter le modèle à deux GPU.

Le réseau se compose de 5 couches convolutionnelles (CONV) et de 3 couches entièrement connectées (FC). L'activation utilisée est l'unité linéaire rectifiée (ReLU). Le réseau a un total de 62 millions de variables entraînables. L'entrée du réseau est un lot d'images RVB de taille 227x227x3 et sort un vecteur de probabilité 1000x1 correspondant à chaque classe. [36]

2.6.2.2 VGGNet

VGGNet est apparu suite à la nécessité de réduire le nombre de paramètres dans les couches CONV et d'améliorer le temps de formation.

Il existe plusieurs variantes de VGGNet (VGG16, VGG19, etc.) qui ne diffèrent que par le nombre total de couches du réseau. VGG16 a un total de 138 millions de paramètres. Le point important à retenir ici est que tous les noyaux conv sont de taille 3x3 et que les noyaux maxpooling sont de taille 2x2 avec un stride de 2. L'idée d'avoir des noyaux de taille fixe est que tous les noyaux convolutionnelles de taille variable utilisés dans AlexNet (11x11, 5x5, 3x3) peuvent être reproduits en utilisant plusieurs noyaux 3x3 comme blocs de construction. La reproduction se fait en termes de champ réceptif couvert par les noyaux. [36]

2.6.2.3 ResNet

ResNet tient compte de ce réseau en introduisant deux types de "connexions raccourcies" Il existe plusieurs versions des architectures ResNetXX, où "XX" désigne le nombre de couches. Les plus courantes sont ResNet50 et ResNet101. Depuis que le problème du gradient de fuite a été résolu, le réseau CNN a commencé à devenir de plus en plus profond.

ResNet utilise principalement la convolution 3x3, qui est similaire au VGG. [36]

2.6.2.4 Inception

Les modules d'Inception sont utilisés dans les réseaux neuronaux convolutifs pour permettre un calcul plus efficient et des réseaux plus profonds grâce à une réduction de la dimensionnalité avec des convolutions 1×1 superposées. Les modules ont été conçus pour résoudre le problème des dépenses de calcul, ainsi que celui de l'overfitting, entre autres. La solution, en bref, est de prendre plusieurs tailles de filtres à noyau dans le CNN, et plutôt que de les empiler de manière séquentielle, de les ordonner pour qu'ils fonctionnent au même niveau. [36]

2.7 Conclusion

Dans ce chapitre, nous avons abordé les principaux concepts liés au Deep Learning. Nous avons également abordé en détail les réseaux de neurones artificiels et leurs différents types. Pour cela, dans le prochain chapitre, nous allons présenter le problème de la localisation de la pose de la caméra et les différentes approches proposées qui sont basées sur l'apprentissage profond.

Chapitre 3

La localisation par réseaux de neurones convolutifs

3.1 Introduction

Dans les deux premiers chapitres, nous avons détaillé les concepts de base de la localisation et du deep learning. Dans ce chapitre, nous allons décrire les étapes principales de la localisation par les réseaux de neurones convolutifs. Le chapitre commence par une description du problème de la localisation et de la base de données utilisée dans ce travail, puis décrit l'architecture générale des réseaux de neurones. Enfin, nous passons à la présentation des approches d'apprentissage par transfert utilisées dans notre projet.

3.1 Description du problème

Pour une image capturée par une caméra, un estimateur de pose absolu cherche à prédire l'orientation et l'emplacement de la caméra en coordonnées mondiales, déterminées pour un modèle 3D de référence arbitraire.

La translation de la caméra par rapport à l'origine (emplacement) est spécifiée par un vecteur $x \epsilon \mathbb{R}^3$. L'orientation de la caméra peut être décrite par plusieurs représentations alternatives, telles qu'une $3x3$ matrice de rotation, quaternion et angles d'Euler. La représentation quaternion est la plus souvent utilisée, en spécifiant l'orientation comme un vecteur $q \epsilon \mathbb{R}^4$. [37]

La représentation d'une rotation sous forme de quaternion (4 éléments) est plus compacte que la représentation sous forme de matrice orthogonale (9 éléments).

Cette représentation élimine le besoin d'orthonormalisation, qui est nécessaire pour les matrices de rotation et peut être convertie en une rotation (légitime) en la normalisant à la longueur unitaire [38].

Les réseaux convolutifs ont récemment été utilisés avec succès pour régresser la pose d'une caméra à partir d'une image RGB. La précision des résultats obtenus était moins bonne que celle des solutions classiques basées sur les caractéristiques, mais cela a donné une nouvelle idée des approches d'estimation de la pose basées sur l'apprentissage.

Notre objectif est d'estimer la pose de la caméra directement en traitant d'images capturées par caméra. Nous proposons un modèle basé sur un réseau de neurones convolutifs et aussi des approches d'apprentissage par transfert qui prédisent un vecteur de pose de la caméra à 7 dimensions contenant le vecteur d'orientation (quaternion à 4 dimensions), et la position (vecteur à 3 dimensions).

3.3 Description de la base de données utilisée

Afin de pouvoir entraîner les réseaux CNN pour traiter le problème de la pose de la caméra, il est nécessaire de posséder un ensemble de données large et uniforme. Dans cette partie, nous présentons la base de données utilisées pour notre travail qui sont téléchargées depuis le site web de Microsoft.

3.3.1 7-Scenes dataset

Le 7-Scenes dataset est une collection d'images de caméras RGB-D. Ce jeu de données peut être utilisé pour l'évaluation de différentes méthodes, telles que les techniques de suivi et de cartographie dense et de relocalisation.

Toutes les scènes ont été enregistrées à partir d'une caméra Kinect RGB-D à résolution 640×480. L'implémentation du système KinectFusion permet d'obtenir les trajets de la caméra et un modèle 3D dense. Plusieurs séquences ont été enregistrées par scène par différents utilisateurs, et divisées en ensembles distincts de séquences d'entraînement et de test.

Pour chaque scène, un fichier zip contenant plusieurs séquences est fourni. Chaque séquence est un flux continu d'images de caméra RGB-D suivies. Le repérage a été effectué à l'aide de l'ICP et de l'alignement image-modèle par rapport à une reconstruction dense.

Chaque séquence (seq-XX.zip) est composée de 500 à 1000 images. Chaque séquence est composée d'un deux fichiers :

- Couleur : frame-XXXXXX.color.png (RGB, 24-bit, PNG)
- Pose : frame-XXXXXX.pose.txt (matrice caméra-monde 4×4 en coordonnées homogènes).

Cet ensemble de données contient une variation importante de la hauteur de la caméra et il a été conçu pour la relocalisation RGB-D. Il est extrêmement difficile à relocaliser de manière purement visuelle à l'aide de caractéristiques de type SIFT, car il contient de nombreuses caractéristiques ambiguës sans structure. [39]

Dans notre travail, nous travaillerons seulement avec 5scence (Chess, Fire, Heads, Pumpkin, Staires).

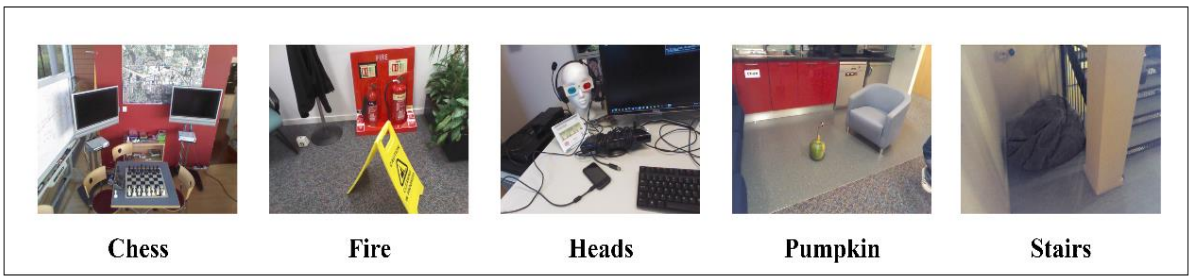

Figure 3-1 : 7scence (Chess, Fire, Heads, Pumpkin, Staires).

Dataset	Espace étendue (m)	Nombre d'images d'entraînement	Nombre d'images de test
Chess	3 x 2 x 1m	4000	2000
Fire	2.5 x 1 x 1m	2000	2000
Heads	2 x 0.5 x 1m	1000	1000
Pumpkin	2.5 x 2 x 1m	4000	2000
Staires	2.5 x 2 x 1.5m	2000	1000

Tableau 3-3 Détails du jeu de données

3.3.2 La représentation en quaternion

Les quaternions constituent une représentation mathématique pratique pour l'orientation et la rotation d'objets en 3 dimensions. Ils sont plus utiles comme moyen de représenter des orientations (rotations). Un avantage des quaternions par rapport aux matrices de rotation est que l'axe et l'angle de rotation sont faciles à interpréter.

Les quaternions sont en fait une extension des nombres complexes de 4 composantes, l'un est un nombre scalaire réel et les 3 autres forment un vecteur dans l'espace imaginaire

$$q = q_0 + iq_1 + jq_2 + kq_3 \ avec\ i^2 = j^2 = k^2 = ijk = -1$$

- q_0 est une valeur scalaire qui représente l'angle de rotation
- q_1, q_2, et q_3 correspondent à un axe de rotation autour duquel l'angle de rotation est effectué.

La représentation quaternion de la rotation peut être exprimée par :

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\left[iu_x + ju_y + ku_z\right]$$

L'interprétation des fichiers "pose" en quaternions se fait à l'aide d'un code MATLAB qui nous permet de convertir la matrice de rotation en un quaternion en utilisant cette formule générale :

$$M = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ 0 & 0 & 0 & 1 \end{bmatrix} => \begin{cases} q_0 = \frac{1}{2}\sqrt{1 + m11 + m22 + m33} \\ q_1 = \frac{1}{4q_0}[m32 - m23] \\ q_2 = \frac{1}{4q_0}[m13 - m31] \\ q_3 = \frac{1}{4q_0}[m21 - m12] \end{cases} \quad avec \begin{cases} x = m14 \\ y = m24 \\ z = m34 \end{cases}$$

3.4 Présentation des outils

3.4.1 Les softwares

3.4.1.1 Python

Le langage de programmation Python a été conçu à la fin des années 1980 et sa mise en œuvre a débuté en décembre 1989 par Guido van Rossum.

Python est le langage de programmation le plus utilisé dans le domaine de l'apprentissage automatique, du Big Data et de la Data Science. Il s'agit d'un langage de programmation interprété, orienté objet, de haut niveau. Son haut niveau construit dans les structures de données, combiné avec le typage dynamique et la liaison dynamique.

Grâce à sa simplicité de syntaxe et à sa flexibilité, le développement en data science et pour la production d'algorithmes de deep learning avec Python est rapide par comparaison avec d'autres langages de programmation.

3.4.1.2 Colaboratory (Colab)

Colab est un environnement Jupyter notebook gratuit qui fonctionne entièrement dans le cloud. Il a été développé par Google pour fournir un accès gratuit aux GPU et TPU à tous les utilisateurs pour construire des modèles d'apprentissage automatique ou de Deep Learning.

Jupyter Notebook (anciennement IPython Notebooks) est un environnement de calcul interactif basé sur le web permettant de créer des documents Jupyter notebook. Le Notebook offre une interface web puissante et innovante pour Python.

Figure 3-2 Représentation de l'interface de Colab.

3.4.1.3 Tensorflow

Tensorflow est une plateforme open-source développée par les chercheurs de Google pour effectuer de l'apprentissage automatique, du Deep Learning et d'autres analyses statistiques et prédictives. Cette bibliothèque de mathématiques symboliques utilise le flux de données et la programmation différentiable pour effectuer diverses tâches liées à l'entrainement et à la détection de réseaux neuronaux profonds.

3.4.1.4 Keras

Keras est une API de deep Learning de haut niveau écrite en Python (sous licence MIT) basée principalement sur les travaux du développeur de Google François Chollet pour mettre en application des réseaux neuronaux. Elle est utilisée pour faciliter la mise des réseaux neuronaux. Elle prend également en charge le calcul de plusieurs réseaux neuronaux en arrière-plan.

3.4.1.5 OpenCV (Open-Source Computer Vision Library)

OpenCV est une bibliothèque logicielle open source de vision par ordinateur et d'apprentissage automatique. OpenCV a été construit pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception artificielle. Lorsqu'il est intégré à diverses bibliothèques, telles que NumPy, python est capable de traiter la structure de tableau OpenCV pour l'analyse. Pour identifier le modèle d'image et ses différentes caractéristiques.

Autres bibliothèques utilisées

- NumPy
- Matplotlib
- Pandas

3.5 Architectures de réseau de neurones

Pour estimer la pose relative de la caméra à partir d'une image. Nous allons tout d'abord former un réseau neuronal convolutif classique. Ce réseau se compose de deux blocs partie convolution et partie régression.

La partie convolution intègre des branches CNN qui partagent les poids et autres paramètres. Généralement, ces branches sont composées de couches convolutionnelles, couche Maxpooling et la fonction d'activation sera ReLU. La partie régression est composée de couches entièrement connectées, pour l'estimation du vecteur de pose à 7 dimensions.

Chaque branche de notre partie convolution conçue à l'origine pour une image d'entrée de taille fixe (224×224). Une telle limitation peut conduire une réduction de la précision de l'estimation de la pose de la caméra, ce qui dégrade les résultats du système en général.

L'image RGB de taille 224*224 pixels est introduite à l'entrée du réseau et les paramètres du réseau neuronal convolutif sont adaptés pour la localisation de la caméra par apprentissage.

Pour avoir des estimations plus précises, il pourrait être bénéfique de traiter des images plus grandes afin de pouvoir extraire plus d'informations de la structure de la scène. Théoriquement, les couches convolutionnelles acceptent des images d'entrée de taille arbitraire, mais elles produisent également des sorties de dimensions réduites. Cependant, la couche FC de la partie régression nécessite des vecteurs de longueur fixe en entrée.

Nous allons faire le même travail pour les différentes scènes (Chess, fire, heads, office, pumpkin, redkitchen, stairs)

Cette approche comprend trois étapes : **Le prétraitement de l'image, L'extraction de caractéristiques, le résultat de la régression.**

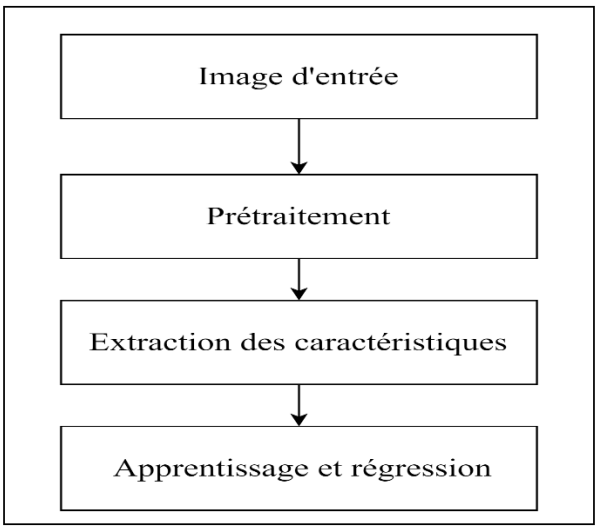

Figure 3-3 : Architecture générale de l'approche proposée

Les données de la "pose" peuvent facilement être importées en tant que DataFrame Pandas.

```
dataset_training=pd.read_csv('/content/drive/MyDrive/chess/quaternions chess/quaternions chess train.txt')
a1=dataset_training['x']
a2=dataset_training['y']
a3=dataset_training['z']
a4=dataset_training['q0']
a5=dataset_training['q1']
a6=dataset_training['q2']
a7=dataset_training['q3']
a1.to_numpy
a2.to_numpy
a3.to_numpy
a4.to_numpy
a5.to_numpy
a6.to_numpy
a7.to_numpy
pose=np.stack((a1,a2,a3,a4,a5,a6,a7))
train_pose=pose.T
```

Figure 3-4 : Code Source de traitement des poses.

3.5.1 Pré-traitement des images

La première étape consiste à prétraiter la base de données d'images 7-Scenes.

A. Redimensionnement des images : les images ont été prétraitées à l'aide d'**OpenCV.**

Pour redimensionner les images afin que leur hauteur et leur largeur soient similaires (224*224). Le module **Pickle** permet de stocker et de restaurer les images prétraitées.

L'avantage d'utiliser pickle est qu'il peut mettre en série pratiquement tous les objets Python, sans avoir besoin d'ajouter du code supplémentaire.

```
path = glob.glob("/content/drive/MyDrive/HEADS/seq-02 PNG/*.png")
from keras.preprocessing import image
for file in path :
  image0=cv2.imread(file)
  image1 = cv2.resize(image0, (224, 224))
  x1 = image.img_to_array(image1)
  x1 = np.expand_dims(x1, axis=0)
  x1= imagenet_utils.preprocess_input(x1)
  x_scratch.append(x1)

train_images= np.vstack(x_scratch)
train_images.shape

pickle.dump(train_images,open("/content/drive/MyDrive/HEADS/train", 'wb'), protocol=4)
pickle_in = open("/content/drive/MyDrive/HEADS/train","rb")
train_images= pickle.load(pickle_in)
```

Figure 3-5 : Code Source de redimensionnement des images.

B. Division des images : le jeu de données est subdivisé en deux sous-ensembles (Train, test).

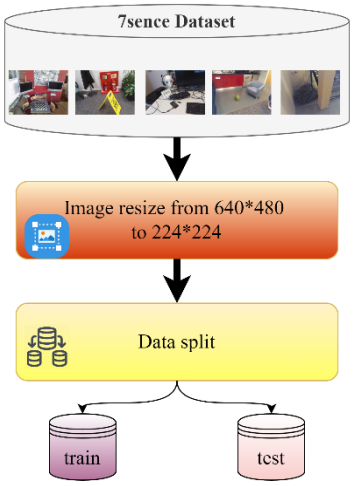

Figure 3-6 : Pré-traitement des images.

3.5.2 L'extraction de caractéristiques

Les résultats de ces expériences ont été testés avec différents modèles, mais celui que nous avons trouvé avec une précision satisfaisante est l'utilisation des architectures donner par les sections suivantes :

3.5.2.1 Architecture basée sur notre propre model CNN

Une fois l'image ayant subi un prétraitement, elle procède à une extraction de caractéristiques, en constituant un réseau neuronal convolutifs avec 4 couches de convolutions et 4 couches de maxpooling.

L'architecture est résumée dans la figure suivante :

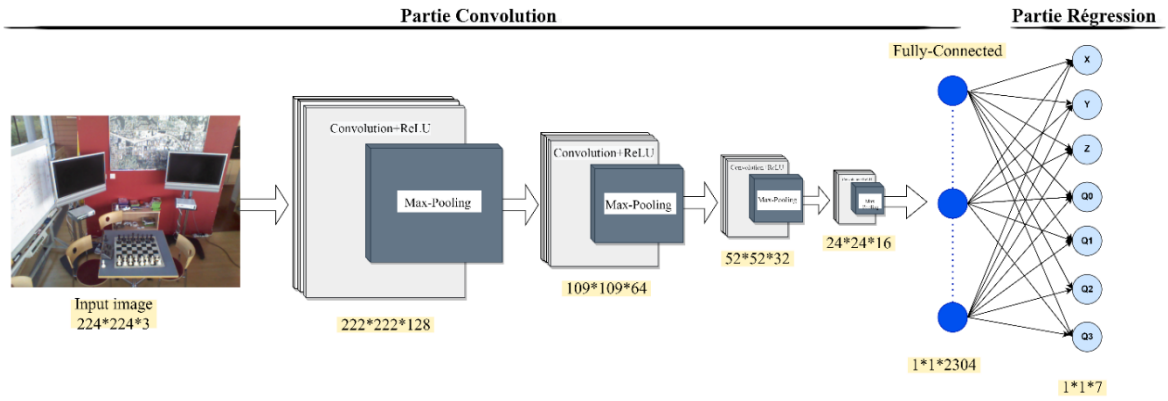

Figure 3-7 : Architecture basée sur notre propre model CNN.

```
input_shape = (224, 224, 3)
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(16, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(7, activation="linear"),
    ]
```

Figure 3-8 : Code source de l'extraction des caractéristiques et de la régression.

La figure suivante présente le résumé d'un réseau neuronal convolutif :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 111, 111, 128)    0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      73792

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 24, 24, 16)        4624

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 16)       0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 7)                 16135

=================================================================
Total params: 116,599
Trainable params: 116,599
Non-trainable params: 0
```

Figure 3-9 : Le résume du model.

Plusieurs moyens permettent d'améliorer l'apprentissage des réseaux de neurones d'apprentissage profond. Optimiser les paramètres, en jouant par exemple sur la façon de visualiser les différentes données, adopter le bon algorithme d'optimisation, etc.

Les optimiseurs sont des algorithmes ou des méthodes utilisés pour modifier les attributs de votre réseau neuronal, tels que les poids et le taux d'apprentissage, afin de réduire les pertes. On s'intéressera donc à une façon d'optimiser le processus d'apprentissage en utilisant des algorithmes d'optimisation basés sur la méthode de descente du gradient.

A. la descente de gradient

La descente de gradient est la plus célèbre méthode pour optimiser un réseau neuronal elle est utilisée pour déterminer les valeurs des paramètres d'une fonction qui réduisent le mieux possible la fonction de coût.

Afin que la descente de gradient arrive au minimum local, nous devons définir le taux d'apprentissage à une valeur adaptée, qui ne doit être ni trop faible ni trop élevée. Cela est important parce que si les mesures qu'elle prend sont trop élevées, elle risque de ne pas atteindre le minimum local à cause des rebonds entre les fonctions convexes de la descente de gradient. Si nous définissons le taux d'apprentissage à une valeur très faible, la descente par gradient finira par atteindre le minimum local, mais cela pourra prendre un certain temps.

B. RMSProp

RMSProp "Root Mean Squared Propagation", est une extension de l'algorithme d'optimisation par descente de gradient.

La RMSProp est conçue pour accélérer le processus d'optimisation, par exemple en diminuant le nombre des évaluations de fonctions nécessaires pour atteindre les optimums, ou bien pour améliorer la capacité de l'algorithme d'optimisation, par exemple en permettant d'obtenir un meilleur résultat au final.

RMSProp élimine la nécessité d'ajuster le taux d'apprentissage. De plus, RMSProp choisit un taux d'apprentissage différent pour chaque paramètre.

L'étape suivante après la configuration de l'architecture du modèle avec Keras consiste à effectuer l'opération de compilation. Au moment de la compilation, il est souvent nécessaire de spécifier trois paramètres « Loss, Optimizer, Metrics).

La figure ci-dessous montre le code permettant d'utiliser l'algorithme d'optimisation RMSProp dans l'architecture CNN.

```
model.compile(loss='mean_squared_error',optimizer=optimizers.RMSprop(lr=1e-5), metrics=['MeanAbsoluteError'])
```

Figure 3-10 : Code source d'optimiseur RMSprop.

- **Optimizer** est l'un des deux arguments nécessaires à la compilation d'un modèle Keras. Les optimiseurs sont des algorithmes ou des méthodes utilisés pour minimiser une fonction d'erreur (fonction de perte) ou pour maximiser l'efficacité de la production.
- **Loss** est correspond à la fonction de coût qui sera utilisée par le modèle pour réduire au minimum les erreurs.
- **Metrics** est une fonction qui est utilisée pour évaluer la performance de notre modèle. Dans les problèmes de régression, on utilise l'erreur absolue moyenne (EAM) pour calculer le rapport entre les valeurs prédites et les valeurs cibles.

Les modèles Keras sont formés sur des tableaux NumPy d'entrées et d'étiquettes. Pour entraîner notre modèle CNN, nous utilisons en général la fonction fit.

```
history = model.fit(train_images, train_pose,
                    epochs=150,
                    batch_size=32,
                    validation_data=(test_images, test_pose))
```

Figure 3-11 : Code source d'entraînement de model.

- **Batch_size** : La taille du batch est un hyperparamètre qui définit le nombre d'échantillons à traiter avant de mettre à jour les paramètres du modèle interne.
- **Epochs** : Le nombre d'époques est un hyperparamètre qui définit le nombre de répétitions de l'algorithme d'apprentissage sur l'ensemble des données entraînées.

La figure ci-dessous illustre le résultat d'exécution de la fonction fit sur les images d'entrainement et des images de validation.

```
Epoch 1/150
125/125 [==============================] - 32s 153ms/step - loss: 19.6219 - mean_absolute_error: 3.2503 - val_loss: 5.5875 - val_mean_absolute_error: 1.8777
Epoch 2/150
125/125 [==============================] - 17s 135ms/step - loss: 3.3313 - mean_absolute_error: 1.4223 - val_loss: 1.7704 - val_mean_absolute_error: 1.0555
Epoch 3/150
125/125 [==============================] - 17s 134ms/step - loss: 1.0385 - mean_absolute_error: 0.8045 - val_loss: 0.7586 - val_mean_absolute_error: 0.6940
Epoch 4/150
125/125 [==============================] - 17s 136ms/step - loss: 0.5037 - mean_absolute_error: 0.5611 - val_loss: 0.4375 - val_mean_absolute_error: 0.5262
Epoch 5/150
125/125 [==============================] - 17s 135ms/step - loss: 0.3204 - mean_absolute_error: 0.4471 - val_loss: 0.3126 - val_mean_absolute_error: 0.4421
Epoch 6/150
125/125 [==============================] - 17s 136ms/step - loss: 0.2361 - mean_absolute_error: 0.3826 - val_loss: 0.2582 - val_mean_absolute_error: 0.3996
Epoch 7/150
125/125 [==============================] - 17s 136ms/step - loss: 0.1900 - mean_absolute_error: 0.3420 - val_loss: 0.2165 - val_mean_absolute_error: 0.3651
Epoch 8/150
```

Figure 3-12 : Résultat d'exécution de la fonction fit.

- **Loss :** la valeur de la fonction de perte pour les images d'entraînement.
- **Mean_absolut_error :** valeur de l'erreur pour les images d'apprentissage.
- **Val_loss :** valeur de la fonction de perte pour les images de validation.
- **Val_mean_absolut_error :** valeur de l'erreur pour les images de validation.

Une fois que la construction du modèle est fixée, nous pourrons l'affiner. Les hyperparamètres les mieux ajustés sont :

- **La taille des mini-batchs** : Si la taille du batch est trop petite, la descente du gradient ne sera pas régulière. Le modèle est plus lent à apprendre et la valeur du Loss peut osciller. Si la taille du batch est trop élevée, le temps d'exécution d'une itération d'apprentissage sera long avec des rendements relativement faibles**.**
- **Taux d'apprentissage :** Le taux d'apprentissage est un hyperparamètre qui contrôle dans quelle mesure le modèle doit être modifié en réponse à l'erreur estimée à chaque fois que les poids du modèle sont actualisés.
- **Hyperparamètres spécifiques aux différentes couches (comme le dropout) :**
  Le Dropout est une méthode de régularisation qui permet d'approximer l'entrainement d'un grand nombre de réseaux neuronaux avec des architectures différentes en parallèle. Dropout généralement compris entre 20 et 50 %. On peut commencer avec 20%. Si le modèle est surajusté, on augmente cette valeur.

3.5.2.2 Architecture basée sur VGG16

Le réseau a été entraîné sur le jeu de données **ImageNet**, qui contient plus de 14 millions d'images haute résolution correspondant à 1000 labels différents. ImageNet est une base de données d'images organisée selon la hiérarchie de **WordNet** dans laquelle chaque nœud de la hiérarchie est décrit par des centaines et des milliers d'images.

Cette architecture utilise un modèle pré-entraîné, le VGG16. Ce dernier reçoit en entrée l'image, et en sortie il fournit la position et l'orientation, la figure suivante montre l'architecture du modèle pré-entraîné VGG16.

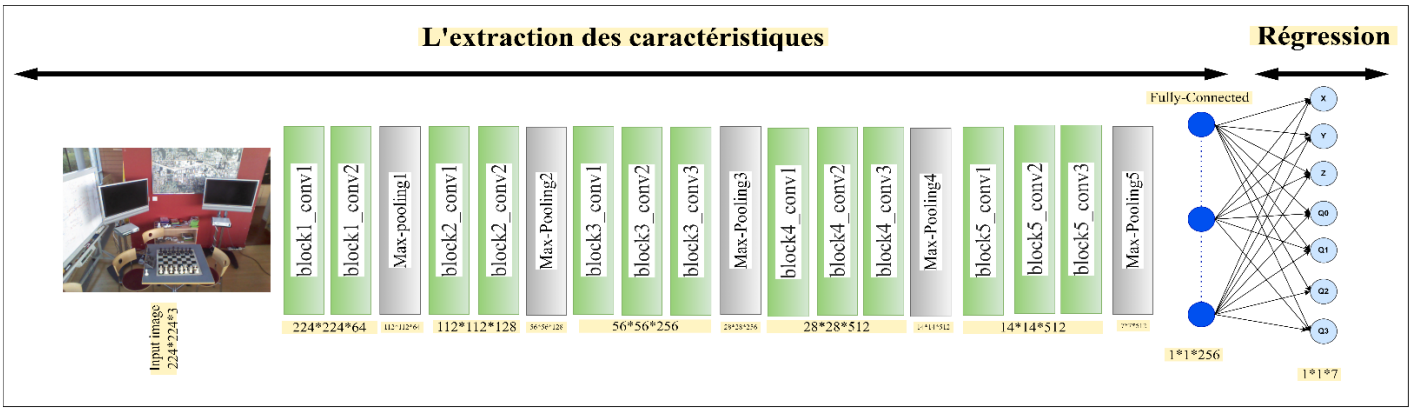

Figure 3-13 : l'architecture du modèle pré-entraîné VGG16.

Nous avons utilisé « **Include_top=False »** pour retirer la couche de classification qui a été entraînée sur le jeu de données ImageNet et définir le modèle comme non entraînable.

- **Include_top** : inclure les couches entièrement connectées en haut du réseau.
- **Weights :** soit None (initialisation aléatoire) ou imagenet (ImageNet).

Dans cette architecture, la première étape est de définir le modèle pré-entraîné VGG16 comme indiqué dans la figure suivante.

```
model_vgg16 = vgg16.VGG16(weights='imagenet',include_top=False,input_shape=(224, 224, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

```
model_vgg16.summary()
```

Figure 3-14 : code source de modèle pré-entrainé VGG16.

Deuxièmement, la fonction d'extraction de caractéristiques est définie pour les deux bases. (Train, test)

```
[ ] train_features = model_vgg16.predict(train_images,batch_size=32)
```

```
[ ] test_features = model_vgg16.predict(test_images,batch_size=32)
```

```
[ ] train_images=[]
    test_images=[]
```

```
[ ] train_features.shape
```

```
    (4000, 7, 7, 512)
```

```
[ ] test_features.shape
```

```
    (2000, 7, 7, 512)
```

```
[ ] train_features_flatten = np.reshape(train_features, (4000, train_features.shape[1]*train_features.shape[2]*train_features.shape[3]))
    test_features_flatten = np.reshape(test_features, (2000, test_features.shape[1]*test_features.shape[2]*test_features.shape[3]))
```

Figure 3-15 : code source d'extraction des caractéristiques.

Le modèle compte plus de 14 millions de paramètres entraînés et se termine par une couche de maxpooling qui fait partie de la partie "extraction des caractéristiques" du réseau.

Après l'extraction des caractéristiques on va lancer la compilation et l'entrainement du modèle.

```
history = model.fit(train_features_flatten, train_pose,
                    epochs=10,
                    batch_size=32,
                    validation_data=(test_features_flatten, test_pose),callbacks=[checkpoint])
```

Figure 3-16 : Code source d'entraînement de model VGG16.

Les callbacks de Tensorflow sont des fonctions ou des blocs de code qui sont exécutés à un moment précis lors de l'entrainement d'un modèle.

- **ModelCheckpoint** : La fonction de rappel ModelCheckpoint est utilisée en conjonction avec l'apprentissage à l'aide de model.fit () pour sauvegarder un modèle ou des poids (dans un fichier de contrôle) à un certain intervalle.
- **Model.save** permet d'enregistrer l'architecture, les poids et la configuration de formation d'un modèle dans un seul fichier/dossier. Cela permet d'exporter le modèle afin qu'il puisse être utilisé sans avoir accès au code initial.

```
model.save('/content/drive/MyDrive/chess/model_best_vgg16.h5')
```

Figure 3-17 : Code source d'enregistrement de modèle.

3.5.2.3 Architecture basée sur VGG19

L'architecture du VGG-19 est très semblable à celle du VGG16. Le réseau VGG19 comporte 3 couches convolutives supplémentaires. L'architecture est la suivante :

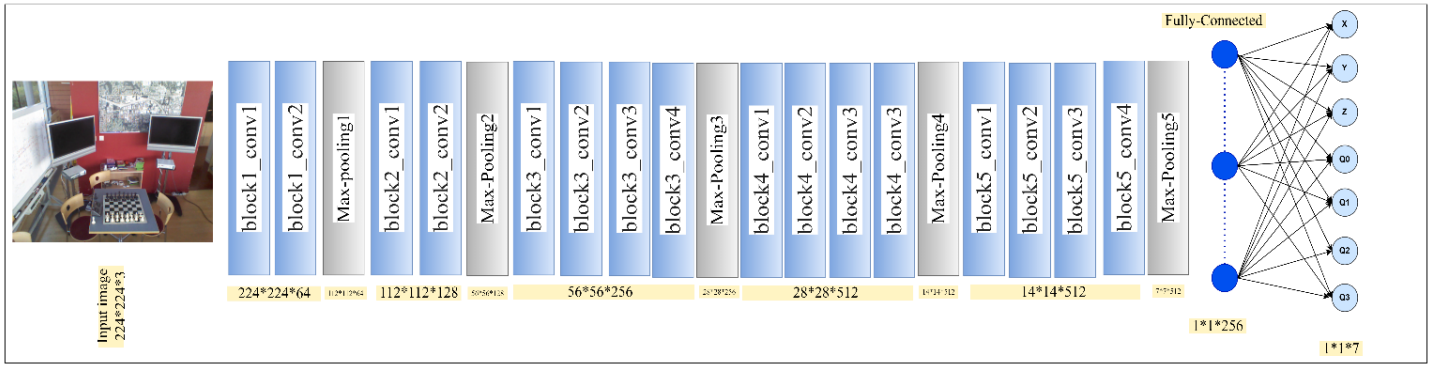

Figure 3-18 : L'architecture du modèle pré-entraîné VGG19

```
conv_base = vgg19.VGG19(weights='imagenet',include_top=False,input_shape=(224, 224, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19
80142336/80134624 [==============================] - 0s 0us/step
80150528/80134624 [==============================] - 0s 0us/step
```

Figure 3-19 : code source de modèle pré-entrainé VGG19

Le modèle VGG19 possède plus de 20 millions de paramètres entraînables

Pour ce modèle, les paramètres de compilation et d'entrainement utilisés sont les mêmes que ceux du modèle vgg16 précédent.

3.5.2.4 Architecture basée sur ResNet50

Le modèle ResNet-50 est constitué de 5 étapes, chacune comportant un bloc de convolution et un bloc d'identité. Chaque bloc de convolution comporte 3 couches de convolution et chaque bloc d'identité comporte également 3 couches de convolution. Le modèle ResNet-50 possède plus de 23 millions de paramètres entraînables.

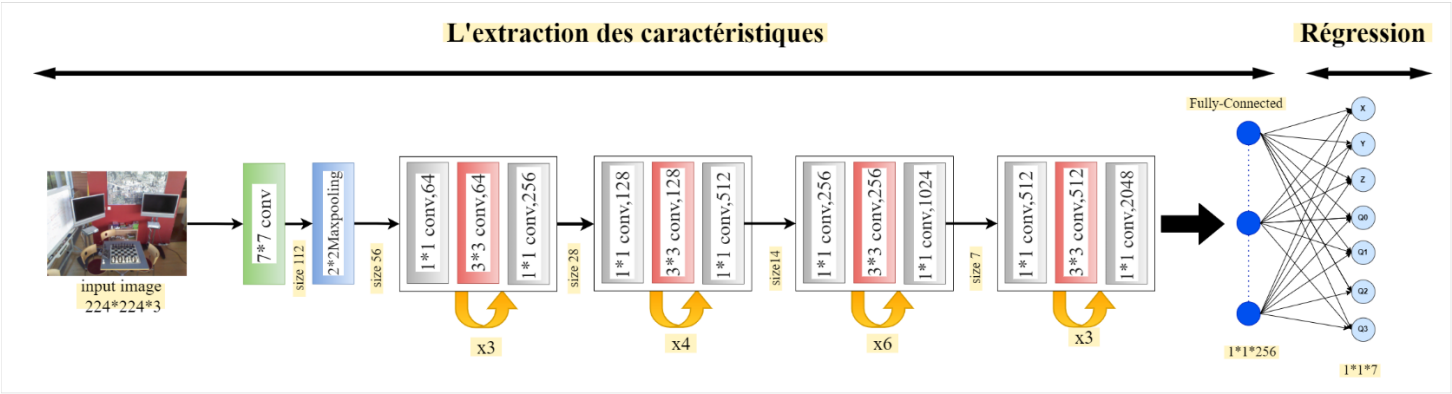

Figure 3-20 : L'architecture du modèle pré-entraîné ResNet50.

L'architecture ResNet 50 contient les éléments suivants :

- Une convolution avec une taille de kernel de (7*7) et 64 kernels différents, tous avec un stride de taille 2, ce qui donne 1 couche.
- Max pooling avec également un stride de taille 3.
- La convolution suivante, il existe un kernel de (1*1) puis un kernel de (3*3) et enfin un kernel de (1*1). Ces trois couches sont répétées au total 3 fois, ce qui nous donne 9 couches dans cette étape.
- Ensuite, il y a un kernel de (1*1), puis un kernel de (3*3) et enfin un kernel de (1*1). Cette étape a été répétée 4 fois, ce qui nous donne 12 couches dans cette étape.

- Après cela il y a un kernel de (1*1) et deux autres kernel avec (3*3) et (1*1) et ceci est répété 6 fois nous donnant un total de 18 couches.
- Et encore un kernel de (1*1) et deux autres de (3*3) et (1*1) et ceci a été répété 3 fois ce qui nous donne un total de 9 couches.
- Après cela, nous terminons avec une couche entièrement connectée contenant 256 nœuds et à la fin une fonction linéaire, ce qui donne les 7 éléments.

Pour ce modèle, les paramètres de compilation et d'entrainement utilisés sont les mêmes que ceux du modèle VGG16 et VGG19 précédents.

```
model_res =resnet.ResNet50 (weights='imagenet',include_top=False,input_shape=(224, 224, 3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_
94773248/94765736 [==============================] - 1s 0us/step
94781440/94765736 [==============================] - 1s 0us/step
```

Figure 3-21 : code source de modèle pré-entrainé ResNET50

3.6 Conclusion

Dans ce chapitre, nous venons de présenter l'approche proposée pour la régression de la pose de la caméra.

En effet, cette approche est basée sur deux notions distinctes, l'entraînement de réseaux de neurones et l'apprentissage par transfert. Ces solutions sont évaluées via des métriques d'évaluation pour déterminer leur performance et leur efficacité. Les résultats de test seront présentés dans le prochain chapitre.

Chapitre 4

Résultats Expérimentaux

Et Evaluations

4.1 Introduction

Dans ce dernier chapitre, nous présenterons les étapes de l'implémentation de l'approche proposée dans le cadre de l'estimation de la pose de camera. Ce chapitre est composé de deux parties, les résultats expérimentaux des tests et l'implémentation. Nous commencerons par les tests réalisés et enfin, les résultats obtenus et quelques discussions sur les performances de nos modèles.

4.2 Analyse des performances de la régression

Dans le but de vérifier la performance des méthodes proposées, toutes les expériences sont réalisées en utilisant Colab. Les différentes architectures évaluées dans ce travail ont été implémentées toutes en utilisant la bibliothèque Tensorflow et Keras, et entraînées avec un GPU.

L'évaluation des performances d'un modèle de régression nécessite une des métriques différentes de celles utilisées pour évaluer les modèles de classification. Les modèles de régression estiment des valeurs continues et donc, les métriques de performance de régression quantifient la proximité des prédictions du modèle avec les valeurs réelles.

4.2.1 Erreur absolue moyenne (MAE)

C'est la métrique la plus simple parmi toutes les métriques. Elle est mesurée en prenant la moyenne de la différence absolue entre les valeurs réelles et les prédictions.

L'erreur absolue moyenne est donnée par :

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i| \qquad (4.1)$$

y_i et $\hat{y}_i$: Sont respectivement les valeurs réelles et prédites et $n_{samples}$ est le nombre d'échantillons de l'ensemble test.

- **Avantages de MAE**

La MAE obtenue est dans la même unité que la variable de sortie. Elle est plus robuste par rapport aux valeurs aberrantes.

- **Inconvénients de MAE**

Le graphique de la MAE n'est pas différentiable, il faut donc appliquer différents optimiseurs comme la descente de gradient qui peut être différentiable.

4.2.2 Erreur quadratique moyenne (MSE)

L'erreur quadratique moyenne ou MSE (Mean Squared Error) est la fonction de perte la plus simple et la plus courante. Pour calculer la MSE, il faut prendre la différence entre les

prédictions de notre modèle et les valeurs réelles, puis la mettre au carré et en faire la moyenne sur l'ensemble des données.

La MSE ne sera jamais négative, puisque nous mettons toujours les erreurs au carré. La MSE est formellement définie par l'équation suivante :

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2 \qquad (4.2)$$

4.3 Résultats obtenus et discussion

Pour montrer les résultats obtenus pour les approches discutées précédemment, les résultats sont illustrés en termes de métriques d'évaluation mentionnées ci-dessus.

Modèle	Nombre	CNN		Vgg16		Vgg19		ResNet50	
Scène	d'epoch	MAE	Loss	MAE	Loss	MAE	Loss	MAE	Loss
Chess	150	0.2632	0.1398	0.2162	0.1112	0.2160	0.1109	0.2152	0.1101
Fire	150	0.2793	0.1145	0.2305	0.1093	0.2302	0.1096	0.2295	0.1089
Heads	150	0.1845	0.1637	0.1401	0.0455	0.1400	0.0454	0.1398	0.0452
Pumpkin	150	0.2813	0.1462	0.2305	0.1130	0.2304	0.1127	0.2300	0.1126
Stairs	150	0.2156	0.1673	0.1968	0.1106	0.1965	0.1104	0.1960	0.1101

Tableau 4-1: La valeur de (MAE) et perte (Loss) pour chaque modèle.

D'après les résultats montrés dans le tableau précédemment, on conclut que l'algorithme qui a donnée de meilleur résultat est le ResNet50.

Les modèles d'apprentissage par transfert ont initialement les mêmes performances. Cependant, en fin de compte, le pré-entraînement avec ResNet50 est plus performant,

Les résultats expérimentaux montrent que l'approche proposée permet de réduire les erreurs dans les résultats obtenus (rotation et translation).

L'apprentissage par transfert permet non seulement d'améliorer considérablement la vitesse de l'entrainement, mais aussi les performances finales.

Nos résultats montrent que l'apprentissage par transfert peut être utilisé de manière efficace pour entraîner un modèle de régression de pose de caméra.

D'autre part, notre approche basée sur notre propre CNN donne des performances plus faibles que celles de l'apprentissage par transfert.

4.3 Les graphes des architectures

Les modèles proposés précédemment nous ont permis d'obtenir les valeurs du MAE et des pertes pour les différentes scènes.

Le MAE et les pertes sont présentés graphiquement dans les graphiques suivants, dans lesquels chaque modèle est représenté avec différentes epochs.

Pour ne pas nous répéter, nous n'avons affiché que le graphique d'une seule scène pour chaque architecture.

A. Les graphes de l'architecture basée sur notre propre CNN :

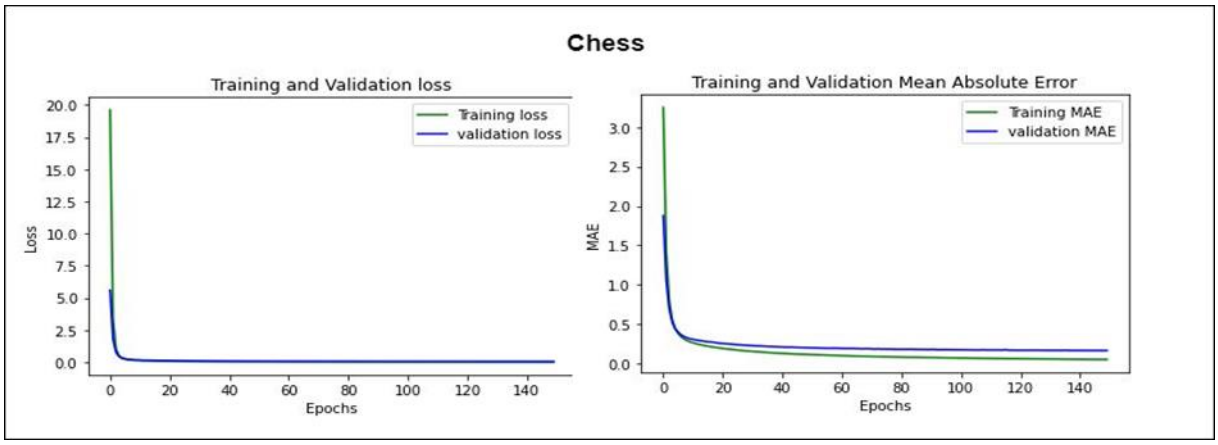

Figure IV-1 : Graphe de (MAE) et perte (Loss) pour Chess.

B. Les graphes de l'architecture basée sur VGG16 :

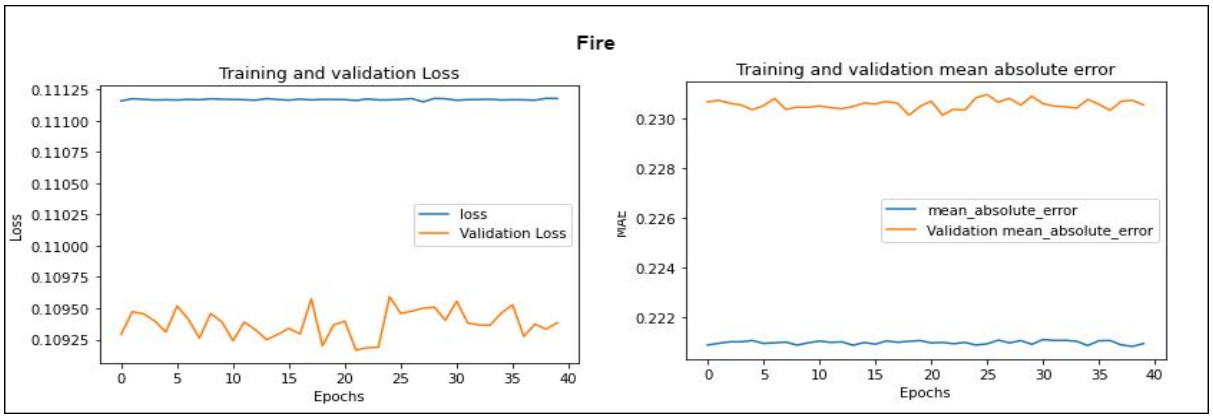

Figure IV-2 : Graphe de (MAE) et perte (Loss) pour Fire.

C. Les graphes de l'architecture basée sur VGG19 :

Figure 4-3 : Graphe de (MAE) et perte (Loss) pour Heads.

D. Les graphes de l'architecture basée sur ResNet50 :

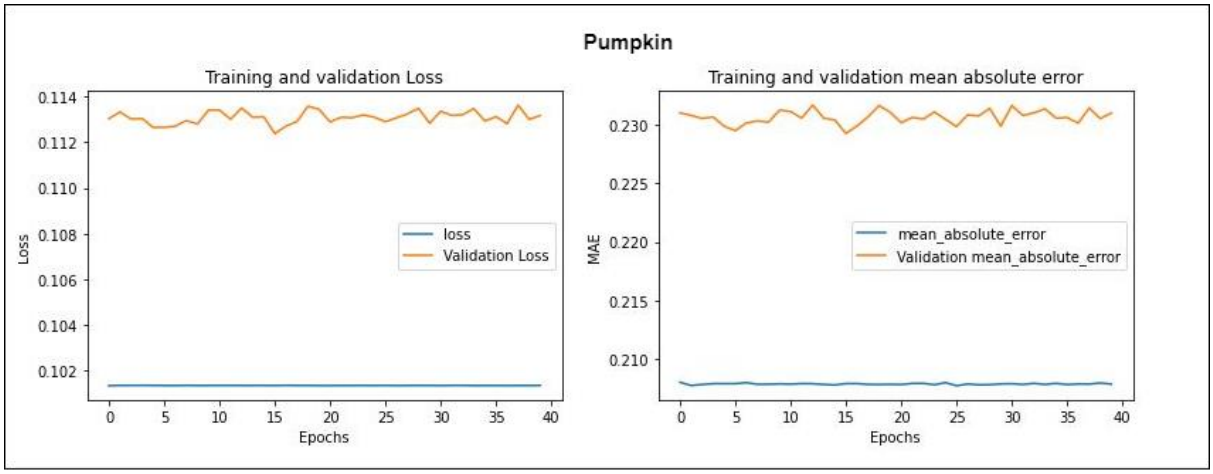

Figure 4-4 : Graphe de (MAE) et perte (Loss) pour Pumpkin.

La valeur de la perte d'entraînement est légèrement supérieure à celle de la perte de validation pour les premières époques, ce qui peut être surprenant. En effet, en apprentissage automatique, il est courant de voir des pertes de validation supérieures aux pertes d'apprentissage. Dans ce cas, cela est simplement dû à la présence du dropout, qui n'est appliqué que pendant la phase de l'entraînement et non pendant la phase de validation.

Nous pouvons voir que la perte d'entraînement devient beaucoup plus petite que la perte de validation. Cela signifie que notre modèle commence à s'adapter à notre ensemble de données d'entraînement après plusieurs d'époques. Cela explique pourquoi la perte de validation ne diminue pas beaucoup par la suite.

4.4 Prédire sur les données de test

Pour évaluer la performance de l'estimation de pose, nous avons besoin d'un ensemble d'images et des poses réelles de la caméra qui les ont capturées.

La fonction **predict ()** permet d'effectuer des prédictions à partir des modèles sur des données nouvelles. Les nouvelles données doivent contenir toutes les colonnes des données d'apprentissage, mais elles peuvent être dans un ordre différent et avoir des valeurs différentes. Dans le cas de nos modèles, au lieu d'effectuer une nouvelle prédiction sur l'ensemble d'apprentissage, il est possible de prédire sur l'ensemble de test, que nous n'avons pas utilisé pour l'apprentissage des modèles. Cela permet de déterminer l'erreur en dehors de l'échantillon pour les modèles.

```
a=vgg16_model.predict(test_featuresvgg16_flatten)
b=vgg19_model.predict(test_featuresVgg19_flatten)
c=res50_model.predict(test_featuresres50_flatten)
d=cnn_model.predict(test_images)
```

Figure 4-5 Code de source de prédiction sur l'ensemble de test

Sachant que la pose réelle $p = (x, q)$ et la pose estimée $\hat{p} = (\hat{x}, \hat{q})$, l'erreur de localisation de $\hat{p}$ se mesure par les écarts entre la translation (localisation) et la rotation (orientation) de p et $\hat{p}$.

L'erreur de translation est généralement mesurée en mètres et définie comme la distance euclidienne entre les positions réelles et celles qui ont été estimées.

L'erreur de rotation est typiquement mesurée en degrés

Le tableau suivant présente les erreurs obtenues pour chaque architecture (L'erreur de translation et l'erreur de rotation)

Erreur	Modèles	Chess	Fire	Heads	Pumpkin	Stairs
L'erreur de translation (m)	Notre CNN	0.3480	0.2862	0.3467	0.6718	0.5858
	VGG16	0.3365	0.1425	0.2898	0.5029	0.4723
	VGG19	0.3358	0.1425	0.2865	0.4998	0.4721
	ResNet50	0.3326	0.1421	0.2829	0.4893	0.4712
L'erreur de rotation (°)	Notre CNN	09.6624	14.1987	13.4778	11.8265	14.2875
	VGG16	07.9911	11.5006	12.4778	10.9276	12.9984
	VGG19	07.9804	11.4657	12.4531	10.8167	12.9978
	ResNet50	07.0426	11.3876	13.2080	10.6175	12.9934

Tableau 4-2: L'erreur de translation et l'erreur de rotation.

4.5 Performances de localisation

Les erreurs de translation et de rotation sont généralement rapportées sous forme de statistique (dans notre cas on utilise le médiane).

La figure présente les performances médianes des modèles de toutes les scènes, évaluées sur la totalité de l'ensemble des tests.

Les graphes de la figures IV-7 ci-dessous montrent les histogrammes cumulatifs de l'erreur de localisation pour toutes les scènes. Nous remarquons que ResNet50 est généralement plus précis.

Le jeu de données en intérieur contient de nombreuses caractéristiques ambiguës et sans texture et des changements significatifs de point de vue et de trajectoire entre les ensembles de formation et de test qui rendent la localisation plus difficile.

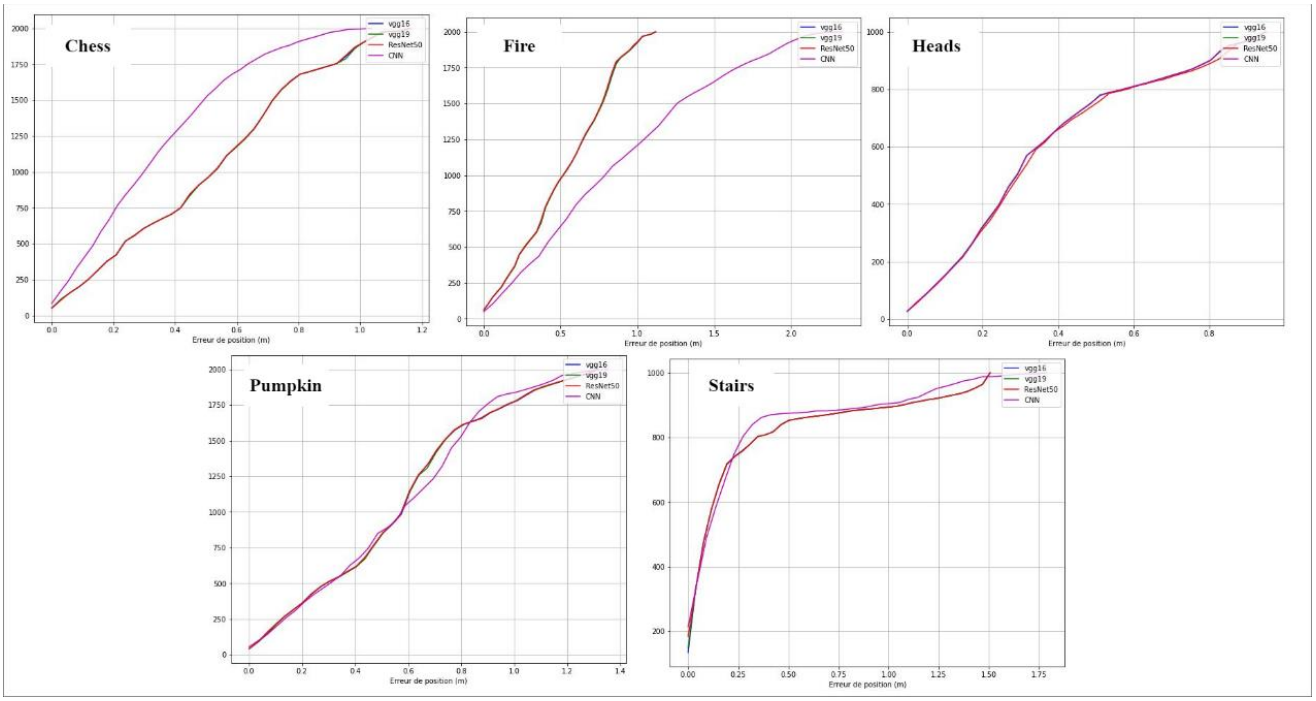

Figure 4-6 : la précision de la localisation pour la position.

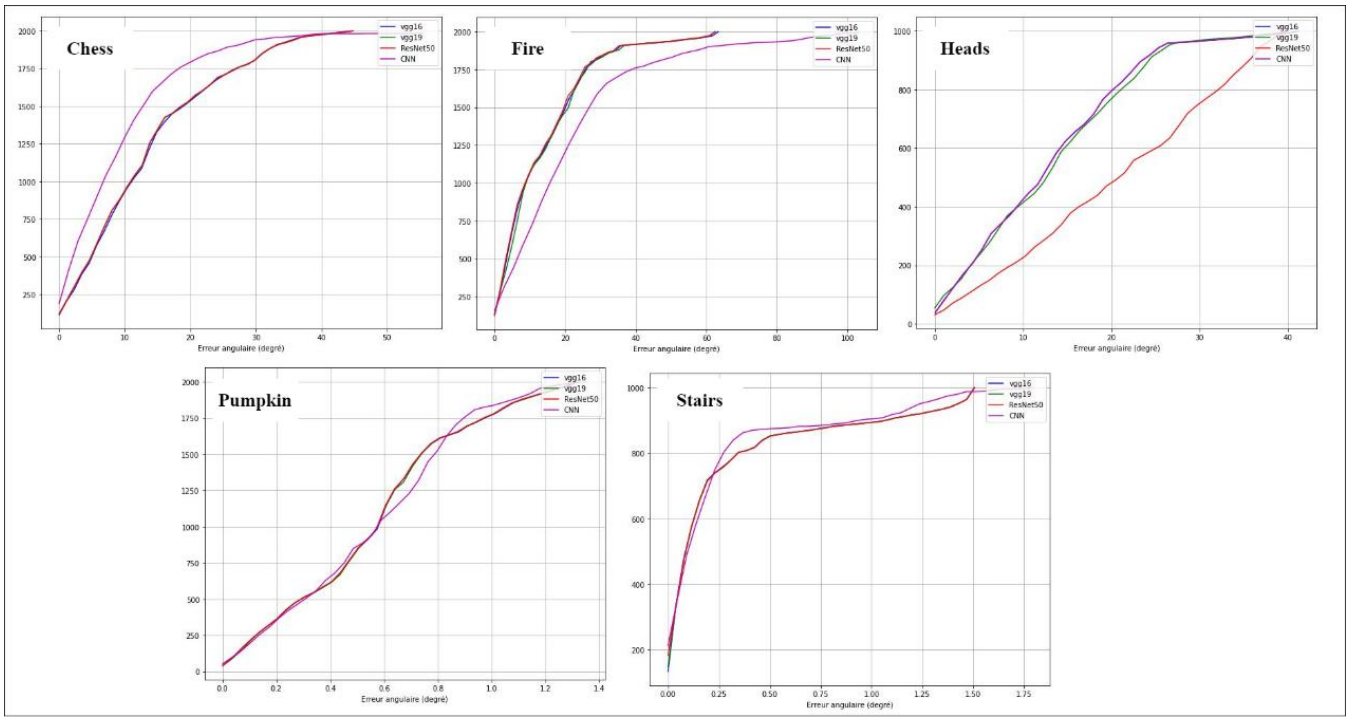

Figure 4-7 : la précision de la localisation pour l'orientation.

Le tableau suivant présente les erreurs obtenues pour chaque cas de manière générale selon les axes (x,y,z). L'erreur angulaire et l'erreur de position sont utilisées pour évaluer les performances de chaque architecture. L'erreur angulaire est utilisée pour calculer les erreurs entre la rotation estimée et les valeurs réelles, qui sont représentées par un quaternion (vecteur à 4 dimensions). D'autre part, l'erreur de position est utilisée pour mesurer les erreurs entre la translation estimée et les valeurs réelles, qui sont représentées par un vecteur à 3 paramètres.

Erreur	Modèle	Chess	Fire	Heads	Pumpkin	Stairs
Erreur selon x(m)	CNN	0.6434	0.6543	0.4536	0.6685	0.1543
	Vgg16	0.5546	0.5456	0.3147	0.6309	0.1318
	V19	0.5547	0.5446	0.3142	0.6296	0.1317
	ResNet50	0.5536	0.5413	0.3149	0.6095	0.1310
Erreur selon y(m)	CNN	0.1685	0.2568	0.0908	0.2643	0.2589
	Vgg16	0.0999	0.1582	0.0750	0.20	0.2260
	Vgg19	0.0978	0.1576	0.0747	0.2006	0.2248
	ResNet50	0.0955	0.1568	0.0738	0.1996	0.2241
Erreur selon z(m)	CNN	0.5134	0.9490	0.3687	0.7469	0.4767
	Vgg16	0.4798	0.8883	0.3331	0.6089	0.4306
	Vgg19	0.4794	0.8875	0.3320	0.6070	0.4338
	ResNet50	0.4791	0.8860	0.3316	0.6065	0.4301

Tableau 4-3: les erreurs de translation obtenues pour chaque cas selon les différents axes.

Les résultats obtenus par l'architecture ResNet50, montrent une légère amélioration par rapport aux résultats obtenus avec les architectures traditionnelles.

De manière générale, l'utilisation de l'apprentissage par transfert pour la régression de la pose présente des avantages : entraînement effectué en continu, la nécessité d'espace mémoire est réduit. L'obtention de la pose est en temps réel (moins de 50 ms pour l'estimation de la pose par image).

4.6 Conclusion

Dans ce dernier chapitre, nous avons présenté en détail l'évaluation et les tests effectués sur nos approches proposes pour les différentes scènes.

Nous avons d'abord présenté les résultats de la prédiction de la position et de l'orientation qui ont été analysés à l'aide des mesures de performance.

Conclusion Générale

Dans ce projet, nous sommes intéressés par la problématique de l'estimation de la pose de la caméra utilisant les images acquises par ce dernier. L'estimation de la pose de la caméra est utile dans de nombreux domaines d'application, tels que la réalité augmentée, la navigation des robots, les véhicules autonomes, etc.

Le domaine de l'estimation de pose avec l'apprentissage profond a rapidement évolué, grâce à la progression de l'apprentissage profond et de la vision par ordinateur.

Dans la plupart des approches d'estimation de la pose d'une caméra, les performances de l'algorithme sont également influencées par des facteurs externes comme l'éclairage et le bruit des capteurs. Ces facteurs externes provoquent souvent des erreurs dans les champs de la caméra donc l'estimation de la pose échoue ou diverge.

Les approches basées sur le réseau neuronal convolutifs permettent au réseau d'être robuste face aux facteurs externes.

Le principal objectif de ce mémoire était d'estimer la pose de la caméra directement en traitant les images capturées par caméra. Le modèle que nous proposons est basé sur un réseau de neurones convolutif qui prédit le vecteur de pose de la caméra.

L'apprentissage par transfert permet d'améliorer de manière significative la vitesse d'apprentissage, ainsi que les performances d'estimation.

Les performances obtenues montrent que l'apprentissage par transfert peut être utilisé efficacement pour l'estimation de pose de caméra.

Perspectives

Ses travaux permettent d'ouvrir des perspectives de recherche scientifique sur le court et le long terme.

De plus,

- Il serait très utile de travailler avec des bases de données de grande taille ou bien de créer notre propre jeu de données.

-Créer des modèles fine tuning.

- Nous pouvons implanter ou implémenter l'algorithme d'estimation de la pose à sous un processeur de traitement de signaux (comme FPGA ou sous Raspberry), qu'il peut acquérir les images d'une caméra.

Bibliographie

[1] A. K. Solutions, Robotics, Infinity Science Press, April 30, 2007.

[2] O. A. A. Mohammad , H. M. Mohammad , S. M. Iqbal et B. I. Napsiah , «Review of visual odometry: types,approaches, challenges, and applications,» *SpringerPlus,* p. 5:1897, 2016.

[3] P. Henry, M. Krainin, E. Herbs, X. Ren et D. Fox, «RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments,» *Springer, Berlin, Heidelberg,* pp. 477-491 , 2014.

[4] M. Marie-Anne , «Caméras 3D pour la localisation d'un système mobile en environnement urbain,» *HAL open science,* 2015.

[5] J. Klas, B. Martin, K. Fredik et A. Kalle, «Image-Based Localization Using Hybrid Feature Correspondences,» *Centre for Mathematical Sciences,Lund University, Lund, Sweden.*

[6] G. Puneet , R. Stergios I et S. Gaurav S, «Robot Localization Using Relative and Absolute Position Estimates,» *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients* , vol. 2, pp. 1134-1140, 1999.

[7] F. BOUZIANE , «HTL : Une approche pour l'hybridation des techniques de localisation.».

[8] A. Neil , «Relativity in the Global Positioning System,» *Living Rev. Relativity,* 2003.

[9] D.-W. Hugh, Fellow, IEEE et B. Tim , «Simultaneous Localisation and Mapping (SLAM):Part I The Essential Algorithms».

[10] W. Yihong, T. Fullin et L. Heping, «Image-based camera localization: an overview,» *Visual Computing for Industry,Biomedicine,and Art,* 2018.

[11] S. Noah , M. S. Steven et S. Richard , «Modeling the world from internet photo collections,» *International Journal of Computer Vision* , p. 189–210, 2008.

[12] D. Nam-Duong , K. Amine , S. Catherine , R. Pierre-Yves et R. Jérôme , «Relocalisation Robuste de Caméra en Temps Réel pour la Réalité Augmentée par une Approche Hybride,» *HAL open science,* 2018.

[13] Z. Andrew , «Survey on Visual-Based Localization,» *Princeton University.*

[14] X. Xing, J. Jie et Z. Yin, «A review of Visual-Based Localization,» *Computer Science,* 2019.

[15] N. David, N. Oleg et B. James , «Visual odometry,» *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004,* pp. I-I, 2004.

[16] S. Davide et F. Friedrich , «Visual Odometry,» *IEEE ROBOTICS & AUTOMATION MAGAZINE,* pp. 1070-9932, 2011.

[17] S. Torsten , L.-T. Laura , P. Marc et Z. Qunjie , «Understanding the Limitations of CNN-Based Absolute Camera Pose Regression,» *IEEE,* 2019.

[18] C. Nouamane , «Les neurosciences ont inspiré l'intelligence artificielle : le modèle neuronal et les concepts,» Journal Du Net (JDN), 21 09 2020. [En ligne]. Available: https://www.journaldunet.com/solutions/dsi/1494055-les-neurosciences-ont-inspire-l-intelligence-artificielle-le-modele-neuronal-et-les-concepts-2-2/. [Accès le 28 04 2022].

[19] S. M. WARREN et P. WALTER , «A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY,» *Bulletin of Mathematical Biology,* vol. 52, n° %1l/2, 1943.

[20] H. Donald, The Organization of Behavior. A Neuropsychological Theory, New York: Wiley, 1949.

[21] F. ROSENBLATT, «The perceptron: a probabilistic model for information storage and organization in the brain,» *Psychological Review,* vol. 65, n° %16, 1958.

[22] A. Betts, «Going deep with deep learning,» Martech insights, action impact, 2 01 2018. [En ligne]. Available: martechtoday.com. [Accès le 23 05 2022].

[23] O. Keiron et N. Ryan , «An Introduction to Convolutional Neural Networks,» *Aberystwyth University, Ceredigion ,Lancaster University, Lancashire,* 2015.

[24] T. Claude , «LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME,» *HAL open source,* 1992.

[25] C. FERNANDEZ-MALOIGNE et . R. GUILLEVIN, «L'intelligence artificielle au service de l'imagerie et de la santé des femmes».

[26] P. NAKAMOTO , Neural networks and deep learning : deep learning explained to your granny a visual introduction for beginners who want to make their own deep learning neural, 2017.

[27] M. Sauget, «Parallélisation de problèmes d'apprentissage par des réseaux neuronaux artificiels. Application en radiothérapie externe,» 2007.

[28] M. Alp, «Introduction aux Réseaux de Neurones Artificiels Feed Forward,» 2008.

[29] H. S. MURAT , «A BRIEF REVIEW OF FEED-FORWARD NEURAL NETWORKS,» *Ankara University, Faculty of Engineering, Department of Electronics Engineering,* vol. 50, pp. 11-17, 2006.

[30] B. Avijeet , «Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More,» simplilearn, [En ligne]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/. [Accès le 02 Mai 2022].

[31] N. Aditta Das, «Convolution neural network(basic),» Kaggle, [En ligne]. Available: https://www.kaggle.com/discussions/getting-started/171212. [Accès le 22 05 2022].

[32] M. Yani, «Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail,» *Journal of Physics: Conference Series*, 2019.

[33] S. Pranshu , «Basic Introduction to Convolutional Neural Network in Deep Learning,» Analytics Vidhya, 01 Mars 2022. [En ligne]. Available: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/. [Accès le 02 Mai 2022].

[34] Z. Valentino , G. Spacagna, D. Slater et P. Roelants, Python Deep Learning, 2017.

[35] J. Sinno et Q. Yang, «A Survey on Transfer Learning,» *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 22, n° %110, OCTOBER 2010.

[36] A. Aqeel , «Difference between AlexNet, VGGNet, ResNet, and Inception,» 07 Juin 2019. [En ligne]. Available: https://towardsdatascience.com. [Accès le 27 Mai 2022].

[37] Y. Shavit et R. Ferens, «Introduction to Camera Pose Estimation with Deep Learning,» *Toga Networks a Huawei company.*

[38] S. Yoli et F. Ron , «Introduction to Camera Pose Estimation with Deep Learning,» *Toga Networks a Huawei company.*

[39] «RGB-D Dataset 7-Scenes,» 01 Janvier 2013. [En ligne]. Available: https://www.microsoft.com/. [Accès le 26 02 2022].