Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2023

Interface web pour l'analyse des images médicales par modèle d'apprentissage profond pour la médecine personnalisée

Etudiant : Matthieu Roux

Professeur : Adrien Depeursinge

Déposé, le 06 février 2023

HES-SO Valais-Wallis • rue de la Plaine 2 • 3960 Sierre
+41 58 606 89 11 • info@hevs.ch • www.hevs.ch

RESUME

L'engouement pour l'intelligence artificielle en médecine, en particulier pour l'apprentissage profond ces 10 dernières années, concerne majoritairement le domaine de la radiologie pour lequel nombre d'outils sont actuellement utilisés. Néanmoins, les limites de cette technologie en imposent une utilisation très modeste dans le domaine de la médecine personnalisée.

Partant de ce constat, si un moyen d'apprentissage profond convaincant pour la médecine personnalisée existe, est-il possible de mettre à disposition des médecins un outil permettant d'utiliser ce type d'intelligence artificielle à travers une interface web ?

Devant cette problématique, nous avons dans un premier temps pesé les pour et les contres pour vérifier notre hypothèse concernant l'existence d'une technologie d'apprentissage profond utile pour la médecine personnalisée. Ensuite, nous avons sélectionné un outil d'inférence. Puis, nous y avons intégré un modèle d'apprentissage profond adapté aux besoins de la médecine personnalisée (i.e. réalisant l'extraction de caractéristiques profondes (deep features)). Et enfin, nous avons développé une petite application web permettant l'analyse d'examens d'imagerie médicale (CT, PT (et IRM)) par le modèle personnalisé présent dans l'outil d'inférence.

Au terme de ce travail, nous avons pu présenter une solution expérimentale permettant à la recherche en médecine personnalisée, et plus spécifiquement en radiomique, d'utiliser une technologie d'apprentissage profond pour l'analyse d'images médicales, et accessible par Internet via une interface visuelle.

Mots -clefs : intelligence artificielle, apprentissage profond, inférence, médecine personnalisée, radiomique, imagerie médicale, application web, modèle d'apprentissage profond, deep features

AVANT-PROPOS

Ce travail a été réalisé à la suite de ma dernière année de bachelor en informatique de gestion.

Il a été proposé par Monsieur Adrien Depeursinge.

Messieurs Roger Schaer et Vincent Andrearczyk ont activement participé, en particulier et respectivement, pour appréhender le fonctionnement des outils déjà en place et pour tout ce qui concerne le domaine de l'apprentissage profond.

La rédaction de ce document a été réalisée de septembre 2022 à février 2023. En parallèle à ce travail de bachelor, je suis en stage dans une entreprise fournissant des services SAP.

Ce sujet, ayant trait au domaine médical en relation avec mon ancienne activité de médecin-dentiste, m'a particulièrement intéressé. La technologie d'apprentissage profond, aussi, est un élément qui a orienté mon choix.

Pour faciliter la rédaction de ce travail, si un acteur doit être décrit, il sera considéré comme masculin mais cette désignation concerne aussi bien les hommes que les femmes.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont soutenu durant la durée de mes études et durant la réalisation de ce travail de bachelor.

Je remercie plus particulièrement :

- M. Adrien Depeursinge, professeur à la HES-SO Valais-Wallis pour m'avoir suivi, guidé
- et conseillé durant tout ce projet.
- M. Roger Schaer, collaborateur scientifique à l'Institut Informatique de Gestion HES-SO Valais-Wallis pour ses précieux conseils et son excellent support technique tout au long de ce projet.
- M. Vincent Andrearczyk, adjoint scientifique à l'Institut Informatique de Gestion HES-SO Valais-Wallis pour m'avoir guidé et fait profiter de son expertise dans le domaine du deep learning.

Et je remercie plus personnellement :

- M. Jean-Paul Margelisch, coordinateur REA de l'office AI de Martigny, pour son soutien et la confiance accordée me permettant ce reclassement
- Mme Andereggen Christine, ma compagne, pour son précieux soutien et ses encouragements durant ce parcours estudiantin

Ainsi que toute ma famille, ami.es et proches qui m'ont encouragé et soutenu durant toute ma formation.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

DÉFINITIONS

agile : en gestion de projet, représente un principe de gestion souple se centrant sur les risques et les besoins dans les projets et mettant en avant la relation entre les exécutants et les st parties prenantes.

API : (Application Programming Interface) interface de programmation d'application mettant en jeu un système permettant la communication entre des outils informatiques.

data scientist : spécialiste des données, il recueille, traite et analyse les données.

déployer : Mise à disposition des utilisateurs d'un outil informatique.

epoch : cycle complet d'une itération au cours de la phase d'entraînement d'un modèle d'apprentissage automatique profond.

framework : Ensemble de composants et d'outils logiciels offrant une architecture de base pour développement.

GitHub : service web d'hébergement et de gestion d'implémentation de logiciels informatiques.

handcrafting : (ang.) signifie fait manuellement, artisanalement

inférence : processus logique de prédictions efficaces par un modèle d'apprentissage automatique entraîné.

kernel : c'est le noyau de l'ordinateur qui permet la communication entre la partie matérielle et la partie logicielle.

librairie : Collection de composants informatiques pouvant être intégré au développement.

Open source : logiciels ou données informatiques en accès libre.

pip : Python Package Installer. Utilitaire python permettant d'installer des paquets (outils) dans un environnement python.

poids : concernant l'apprentissage profond, le poids fait référence à la quantité de données et d'informations contenues dans le modèle.

Product Backlog : élément central dans la gestion de projet agile, représentant sous forme de liste les fonctionnalités prioritaires

REST : c'est un style d'architecture logicielle qui décrit un ensemble de recommandations à utiliser permettant la transmission de données pour créer des services internet (web).

SCRUM : cadre de développement de projet par méthode agile

SOAP : protocole officiel édité par W3C (World Wide Web Consortium) permettant de transmettre des données pour créer des services internet.

Sprint (méthodologie SCRUM) : période brève dans le temps donnée à une équipe de développement pour qu'elle effectue une quantité de travail déterminée.

SSH : protocole de communication sécurité en technologie de l'information.

token : (jeton) désigne un élément identificateur de session d'authentification.

User story : c'est une description simple, non-formelle d'un besoin dans le domaine du développement logiciel.

Web Application : application informatique manipulable directement sur le réseau internet sans installation sur la machine de l'utilisateur.

web : système reliant un ensemble de pages, entre elles, par des liens hypertextes.

ABRÉVIATIONS

csv : coma-separated value (format de fichier informatique).

CT : Computerized Tomography (technique d'imagerie médicale).

DF : Deep Features (i.e. caractéristiques profondes).

dicom : digital imaging and communication in medecine (format de fichier informatique).

DL : Deep Learning (i.e. apprentissage automatique profond).

IA : intelligence artificielle.

IRM : Imagerie par Résonance Magnétique (technique d'imagerie médicale).

IT : Information Technologies (i.e. technologie de l'information).

JDK : Java Developpment Kit

JRE : Java RunTime Environnement

ML : Machine Learning (i.e. apprentissage automatique non-profond).

NIFTI : neuroimaging informatics technology initiative (format de fichier informatique).

pb : protobuf (format de fichier informatique).

PT : Position Emission Tomography (technique d'imagerie médicale).

VPN : Virtual Private Network (réseau privé virtuel).

WebApp : web Application.

1 Introduction

La place des outils doués d'intelligence artificielle n'a cessé de croître depuis ces dernières décennies et en particulier la technologie de l'apprentissage automatique profond (figure 1), jusqu'à faire croire à une certaine totipotence de son utilisation avec l'arrivée de ChatGPT.

En réalité, et plus particulièrement dans le domaine médical, il n'en est rien. Les outils doués de technologies d'apprentissage automatique sont très spécialisés car chacun a ses avantages et inconvénients à prendre en compte en fonction du travail à fournir mais aussi en fonction des éléments à analyser. Ainsi, dans le domaine particulier de la recherche médicale en médecine personnalisée, le secteur de la radiomique a une place particulière. Il a pour but d'analyser des caractéristiques radiomiques extraites d'examens d'imagerie médicale qui justifierait l'utilisation de DL mais les études se font sur des populations de patients limitées ce qui révoque une étude complète par DL.

Partant du constat mettant en évidence, d'une part, que deux domaines professionnels (i.e. les technologies de l'information (IT) et la médecine) distincts doivent travailler de concert pour exploiter au mieux et avec pertinences les outils informatiques modernes pour la recherche en médecine, et, d'autre part, que les recherches médicales en radiomique exploitent des données exploitables en DL mais inadaptée au développement de bout en bout d'un outil de DL dévolu à ces recherches médicales.

Ce travail de bachelor appartient à ce cadre de regroupement de compétences en ayant pour objectif d'intégrer une technologie d'apprentissage profond dans un domaine de recherche médicale spécifique qui est la radiomique et dont un outil basé sur l'apprentissage automatique conventionnel est déjà mis en place par l'équipe du Pr Depeursinge.

Afin de mener à bien ce projet, en tant que spécialistes, après quelques définitions/rappels utiles, nous replacerons le domaine de l'intelligence artificielle englobant les technologies d'apprentissage automatique dans le contexte médical, puis nous nous concentrerons sur la technologie d'apprentissage profond et sa place dans le domaine médical.

Ensuite, nous analyserons les besoins d'intégration par un accès à distance à des outils d'apprentissage profond pour les chercheurs en médecine et nous illustrerons ce type d'analyse en prenant comme exemple la plateforme QuantImage-V2, opérationnelle.

Une fois, cette analyse effectuée, nous présenterons notre cahier des charges exposant la synthèse de l'ensemble des fonctions souhaitées pour notre projet.

Ensuite, nous traiterons des phases de gestion de ce projet, du choix des technologies et des moyens utilisés, puis on détaillera leur mise en place avant de discuter des résultats, d'en présenter les ouvertures possibles puis de conclure.

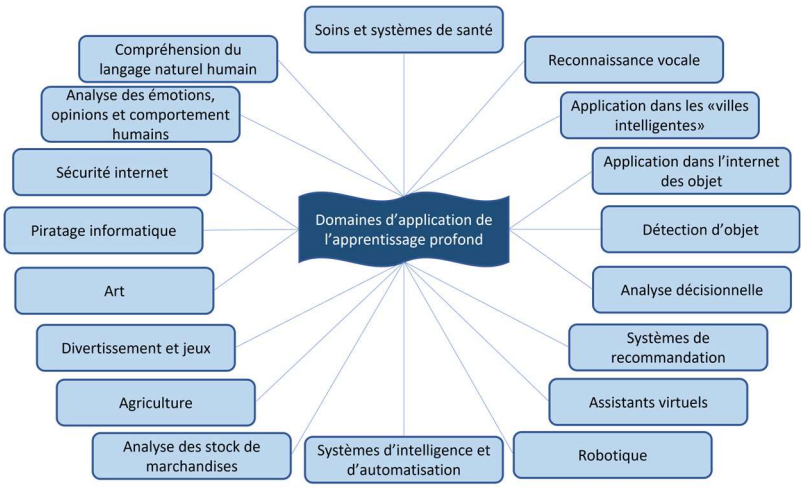


Figure 1 Place de l'apprentissage profond aujourd'hui

Adapté de Sarke (2021)

2 Contexte

2.1 Médecine personnalisée

La médecine personnalisée est une pratique basée sur les principes de la médecine moderne qui s'appuie sur des données cliniques et scientifiques pour fournir des soins centrés sur le patient. Elle se focalise sur les besoins spécifiques de chaque patient et offre un plan de traitement individualisé en fonction de ses antécédents médicaux, de son mode de vie et de ses préférences. En utilisant les technologies médicales les plus récentes, les médecins peuvent obtenir des informations précises mais leurs

analyses et leurs interprétations restent limitées par leurs propres capacités cognitives ne leur permettant de faire une prédiction « formelle » basée, seulement, sur la comparaison de parcours de quelques dizaines de patients (Galanopoulo, 2018).

L'utilisation de l'intelligence artificielle et de l'apprentissage automatique dans le domaine de la médecine personnalisée permet aux médecins d'offrir des soins plus complets et plus précis au patient. Ils peuvent également être utilisés pour créer des modèles prédictifs qui aideront les médecins à identifier les patients à risque et à planifier de meilleures stratégies de traitement grâce à des prédictions basées sur plusieurs années de suivi de parcours de centaines de patients (Galanopoulo, 2018).

2.2 Intelligence artificielle (IA) en médecine

2.2.1 Généralités

Comme nous l'avons déjà définie en introduction, l'IA a la « prétention » de pouvoir simuler le fonctionnement du cerveau humain.

Dès lors, cette capacité permet d'analyser des données d'examens complémentaires (examens biologiques, imagerie médicale, …) et l'histoire médicale (anamnèse, compte-rendu, …) d'un ou de plusieurs patients à des fins d'investigation diagnostic et pronostic, de planification et de prise en charge thérapeutique.

Les outils d'IA à destination médicale sont extrêmement spécialisés dans un domaine particulier et permettent d'offrir un support et/ou une aide pour le médecin dans sa prise en charge médicale. Par exemple, un outil pourra diagnostiquer et pourquoi pas localiser une lésion cérébrale et un autre outil pourra analyser la meilleure approche ou alternative pour son traitement.

Aujourd'hui, l'IA est déjà présente dans bon nombre d'outils utilisés en médecine, depuis 2015 (figure 2) le nombre d'autorisations délivrées par l'administration des Etats Unis d'Amérique (USA) compétente, la Food and Drug Administration (FDA), a bondi en passant de moins de 10 autorisations par an en 2015, à plus de 110 en 2021 (115 autorisations) concernant les outils employant de l'intelligence artificielle et de l'apprentissage automatique.

En regardant un peu plus en détails, on remarque très vite que ces outils prennent place dans le domaine de la radiologie où la FDA a validé 392 outils à cet effet (figure 3).

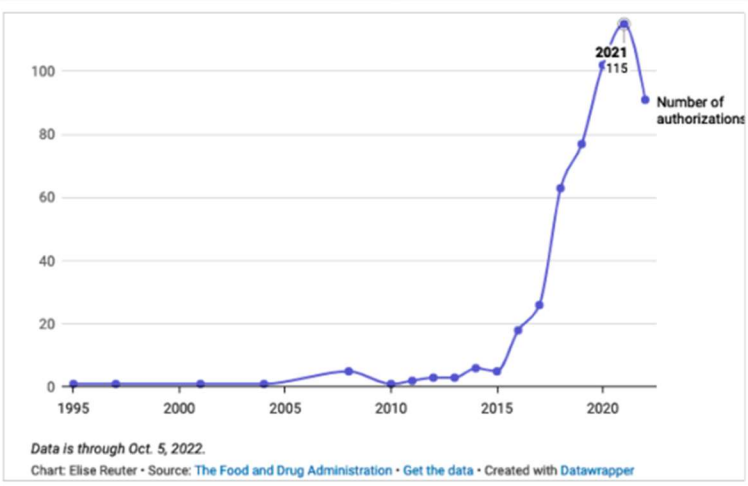


Figure 2 Nombre d'autorisations d'outil IA par la FDA
Source : AblerD.& al., The QuantImage V2 Cloud Platform : A One-Stop Tool For Clinical Radiomics Research, 25.11.2022

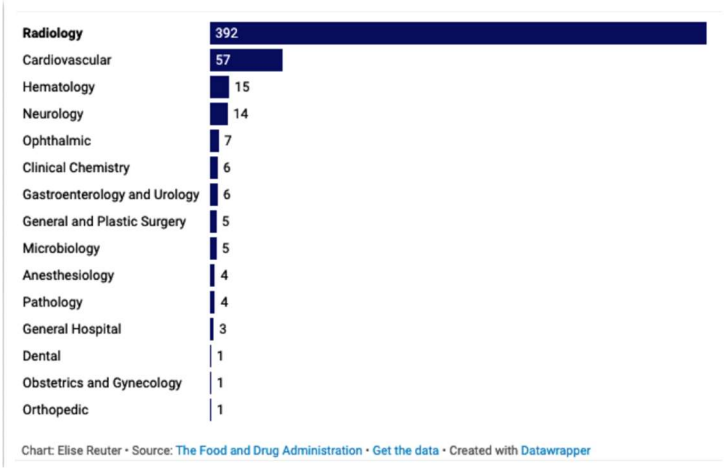


Figure 3 Nombre de dispositifs médicaux autorisés par la FDA selon les spécialités médicales
Source : AblerD.& al., The QuantImage V2 Cloud Platform : A One-Stop Tool For Clinical Radiomics Research, 25.11.2022

2.2.2 Apprentissage automatique (ML) versus apprentissage profond (DL)

L'apprentissage automatique et l'apprentissage profond sont deux sous-ensembles du domaine de l'intelligence artificielle (figure 4).

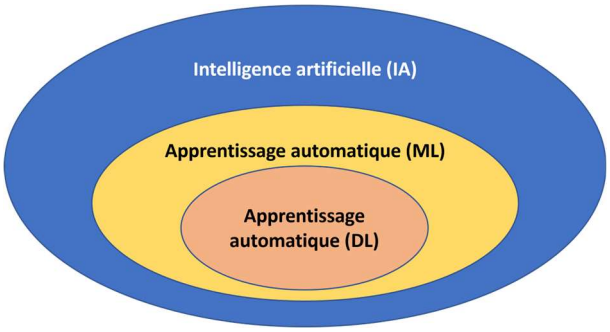


Figure 4 Relations AI, ML et DL

Le premier programme d'apprentissage automatique non deep learning (ML) date des années 1950 (Samuel, 1959). Aujourd'hui, grâce aux algorithmes mathématiques performants sur lesquels elle s'appuie, on peut la considérer comme un domaine efficace qui consiste à développer des modèles informatiques capables d'apprendre à partir de données plus ou moins nombreuses.

Ces algorithmes peuvent être classés en trois grandes catégories (selon nos cours de 7ème semestre de bachelor eninformatique de gestion (Genoud D., 2022)) (stable, s.d.) qui sont l'apprentissage supervisé, l'apprentissage non supervisé et apprentissage par renforcement. Les méthodes les plus couramment utilisées sont le k-plus proches voisins, l'arbre de décision, le régression logistique et la classification ascendante hiérarchique pour la première catégorie, la méthode k-means, le clustering hiérarchique et la factorisation matricielle pour la deuxième, et le Q-learning pour la troisième.

Parmi les outils d'apprentissage ML, il existe aussi des outils s'appuyant sur l'utilisation de neurones artificiels, moins nombreux que dans l'apprentissage profond, que l'on appelle apprentissage superficiel.

Ces différentes méthodes ML sont très efficaces et s'avèrent même plus efficace que les méthodes de deep learning lorsque le volume de données à traiter est en faible quantité (figure 5)

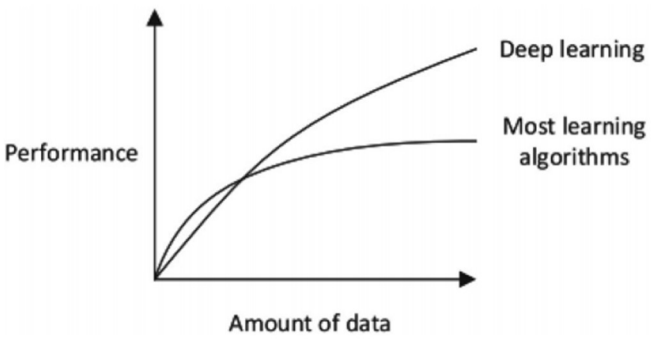


Figure 5 Performances des technologies IA en fonction du volume de données
Source : Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN COMPUT. SCI.* **2**, 420 (2021). https://doi.org/10.1007/s42979-021-00815-1

En ce qui concerne l'apprentissage profond (DL), il s'appuie aussi sur l'analyse par des algorithmes mathématiques mais ces derniers sont effectués uniquement par un ensemble de neurone. L'augmentation de puissance de calculs des ordinateurs a permis leur développement et l'amélioration depuis leur apparition en 2012 (LeCun Y, 2015).

La différence entre ML avec neurone et DL se fait sur le nombre de couches de neurones artificiels le constituant. On considère qu'un réseau de plus de 3 couches de neurones artificiels (figure 6) appartient au DL (Janiesch, 2021).

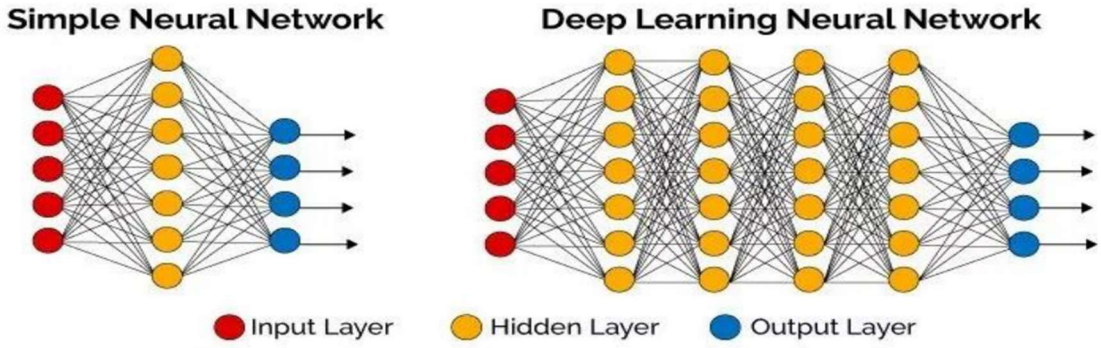


Figure 6 Différences neuronales entre ML et DL
Source : Ayushi Chahal, Preeti Gulia. Machine Learning and Deep Learning. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

Dès lors, on peut considérer que le machine learning regroupe 3 catégories d'apprentissage et que chaque catégorie à ces spécificités, qualités et inconvénient. Si la figure 5 nous montrait la relation entre le volume de données et la performance des apprentissages ML et profond, la figure 7 nous montre les relations entre la complexité des modèles DL et ML, leur

performance d'apprentissage et la difficulté de comprendre leur fonctionnement. Ainsi, les modèles de DL sont plus « opaques » dans la compréhension de leur fonctionnement mais ils se révèlent extrêmement performants (si les conditions vues précédemment sont réunies).

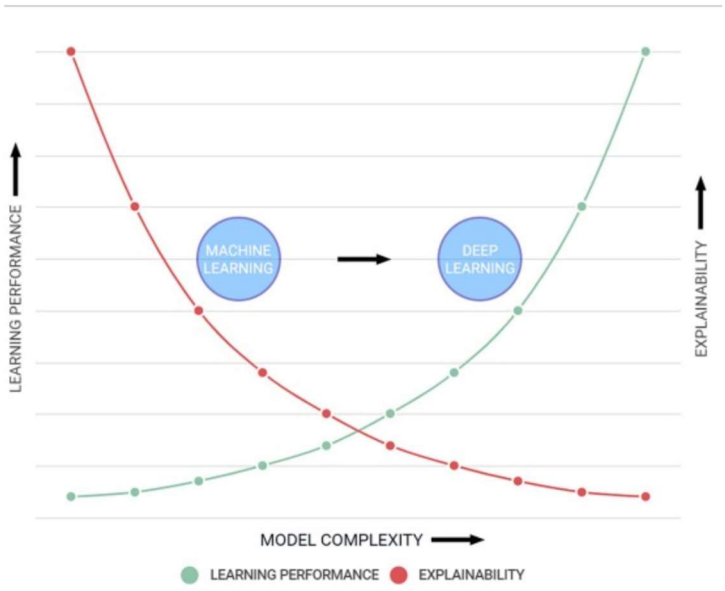


Figure 7 Relations entre performance, compréhension et complexité des modèles DL et ML
Source : Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. *Electron Markets* **31**, 685–695 (2021)

Un autre point très important, en particulier par sa relation directe avec notre travail, c'est la délégation et l'automatisation de certaines tâches. Comme nous le montre la figure 8 en fonction du type d'apprentissage, certaines opérations devant se faire de manière artisanale (handcrafted) avec les modèles simples d'apprentissage, peuvent être déléguées aux modèles constitués de neurones artificiels. Le deep learning est même représenté comme une technologie ne demandant aucune intervention humaine pour effectuer entièrement le traitement des données (figure 9). Cette faculté, du DL, de traiter les données brutes est particulièrement utilisée dans le domaine de l'imagerie médicale (F. Pesapane, 2018).

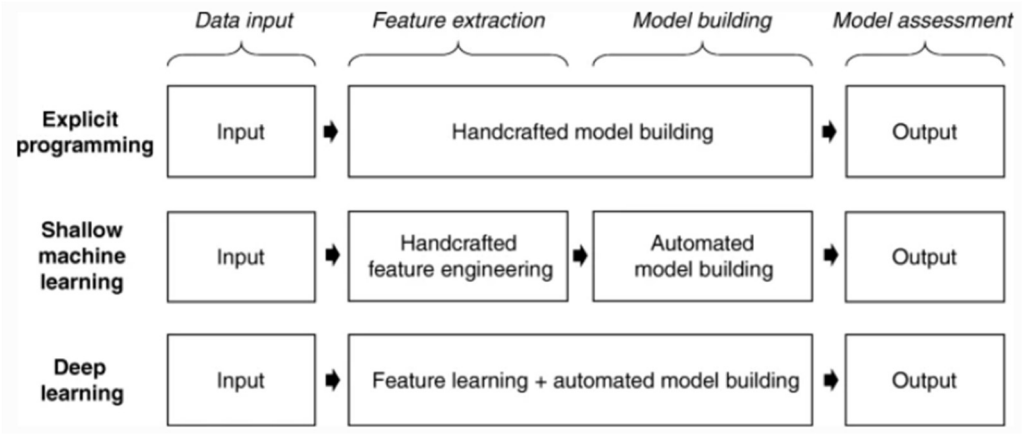


Figure 8 Etapes manuelles vs automatiques en apprentissage automatique
Source : Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. *Electron Markets* **31**, 685–695 (2021)

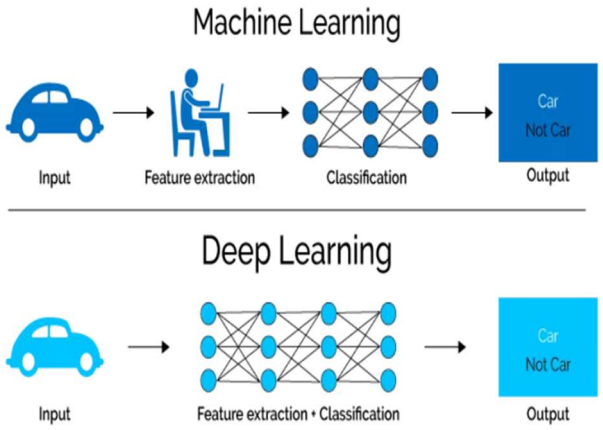


Figure 9 Place de l'être-humain dans le traitement des données en ML et en DL
Source : Ayushi Chahal, Preeti Gulia. Machine Learning and Deep Learning. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

2.2.3 Intégration de l'apprentissage automatique en médecine

Comme nous pouvons le voir sur la figure 10, l'intégration de technologie d'apprentissage automatique implique une équipe constituée de plusieurs disciplines d'expertise ayant chacune un rôle spécifique, sont partie prenante dans l'utilisation de l'apprentissage

automatique et de l'intelligence artificielle en médecine. On peut distinguer deux groupes, tels que :

- les agents experts techniques (de profession médicale ou non) tels que :
    - les spécialistes des données, les scientifiques de l'information au niveau technique dont le rôle sera de :
        - travailler sur les ensembles de données afin d'obtenir 1 sous-ensemble de données pour l'entrainement du modèle et un autre sous-ensemble pour sa validation ;
        - définir des modèles d'apprentissage en choisissant l'environnement et la technologie, en créant des modèles et faire tout le processus permettant de déployer et mettre à disposition des utilisateurs les modèles fonctionnel ;
- et les experts en médecine regroupés en 2 catégories :
    - Tout d'abord, les spécialistes de l'imagerie médicale qui :
        - fournissent les examens d'imagerie médicale après avoir créé et anonymisé des collections d'examens ;
        - et qui détectent, matérialisent et segmentent les lésions présentes sur les examens.
    - Et ensuite, il y a les cliniciens qui posent le problème que doit permettre de résoudre l'intelligence artificielle. Pour cela, il leur faut :
        - définir la tâche à accomplir et ses spécifications ;
        - fixer des critères d'éligibilité ;
        - les cohortes de patients nécessaires ;
        - spécifier un standard de référence.

Toute cette équipe travaille en synergie et intervient à des niveaux de finalités (besoins) différentes lors de l'utilisation de l'outil d'apprentissage.

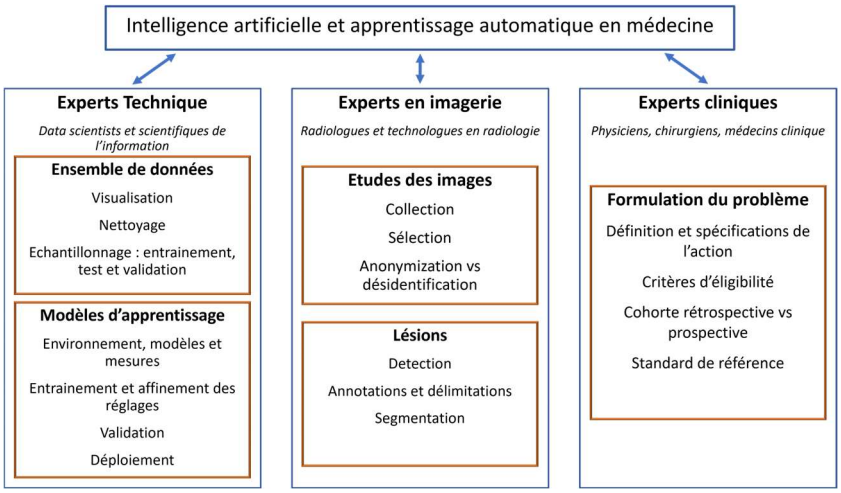


Figure 10 Equipes et rôles pour l'intégration d'IA et ML en recherches médicales
Adapté de Montagnon (2020)

2.3 Conclusion

Les technologies d'IA ont toutes leurs avantages et inconvénients ce qui peut engendrer une inefficacité si leur emploi est inadapté.

Ainsi, l'utilisation de telles technologies par des personnes non-expertes dans ce domaine nécessite l'implication d'experts en intelligence artificielle.

Dès lors, on peut dire que l'apprentissage profond est une technologie plus avancée qui peut traiter des données brutes de manière plus complexe et fournir des résultats plus précis que les technologies non DL. Cependant, la technologie DL nécessite un volume de donnée suffisant pour développer des modèles d'apprentissage performant.

3 Etat de l'art : apprentissage profond et médecine

3.1 Apprentissage profond

3.1.1.1. Architecture des réseaux de neurones artificiels

Comme nous l'avons vue précédemment, l'apprentissage profond s'appuie sur la technologie des neurones artificiels.

Ces derniers ont été définis pour la première fois à la fin des années 1950 (Rosenblatt, 1958) et sont représentés par un objet informatique programmé, capable de recevoir un élément donné à analyser à l'aide d'algorithmes mathématiques lui permettant de faire une prédiction pour un résultat souhaité (figure 11).

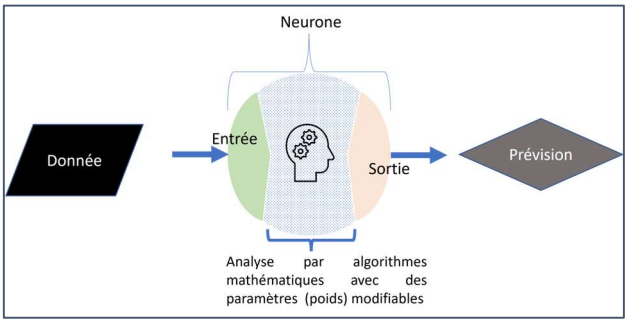


Figure 11 Neurone artificiel (schématisé)

Le réseau de neurones artificiels constitue, dans les faits, un ensemble de ces neurones artificiels mis en relation. Chaque neurone en aval étant dépendant des résultats (prévisions) du neurone le précédent. Un ensemble de neurones ayant un même type de donnée d'entrée constitue une « couche ». Et l'ensemble de toutes les couches liées constitue le modèle d'apprentissage (figure 12).

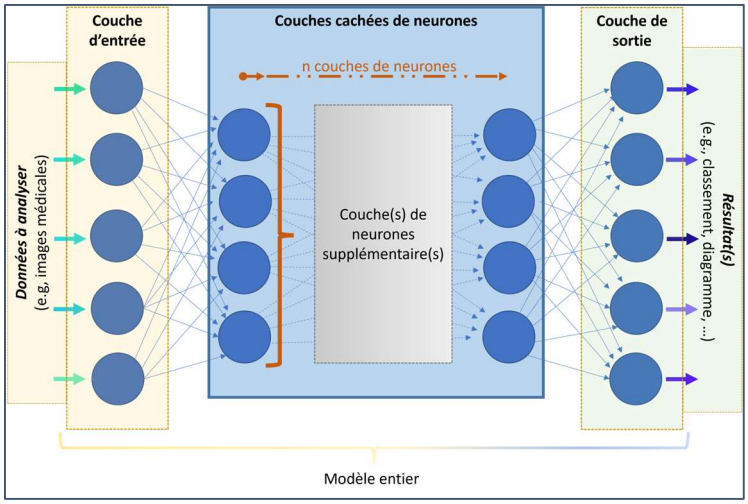


Figure 12 Modèle d'apprentissage profond schématisé

Les modèles ont la caractéristique de pouvoir réévaluer et modifier leurs paramètres de calculs, appelés « poids », au cours de leur apprentissage afin de réaliser des prédictions

fiables et optimales. D'autre part, il existe différentes technologies d'apprentissage que nous allons rapidement présenter.

3.1.1.2. Technologies

Les algorithmes et les fonctions implémentés au niveau des neurones, associés aux différentes combinaisons possibles d'interactions neuronales, ont permis le développement de différentes catégories de modèle d'apprentissage profond.

Actuellement, on distingue :

- les réseaux de neurones à convolution (CNN ou ConvNet)

Leur fonctionnement est basé sur celui de la vision animal et leur nom est dû au fait qu'ils utilisent une matrice dite de convolution pour exécuter leurs opérations d'analyse.

Leur efficacité dans le traitement des images est sans conteste et a été mise en avant lors du défi ImageNet de reconnaissance visuelle à grande échelle en 2012 (Krizhevsky A., 2012).

- Les réseaux de neurones récurrents (RNN)

Ce sont des réseaux constitués de neurones ayant des connexions récurrentes, grâces auxquelles il y a au moins un cycle d'analyse. C'est-à-dire que l'analyse d'un paramètre est faite au moins deux fois et comparée par et dans le neurone, à l'aide d'une mémoire à court terme à l'intérieur même neurone avant de « sortir » leur prédiction.

Ils sont utilisés pour modéliser des comportements à base de séquences, tels que le traitement du langage naturel et la prédiction des séries temporelles.

- Apprentissage par transfert (Transfer Learning)

Cette technique, comme son nom le suggère, se base sur la réutilisation d'un réseau de neurones pré-entraîné sur un ensemble de données d'une problématique particulière (Y. Wei, 2014). Afin de conserver la connaissance acquise par ce dernier et la réutiliser pour traiter une autre problématique.

Son principe est simple, on crée un modèle de DL que l'on valide. Puis on enlève une ou plusieurs couches de neurones que l'on remplace pour tout ou partie (figure 13).

La couche conservée est dite « gelée » et le nouveau modèle sera entrainé uniquement sur la nouvelle portion de neurones.

Le transfert d'apprentissage est largement utilisé dans le domaine médical car les données sont dans la plupart des cas insuffisantes pour réaliser un apprentissage performant.

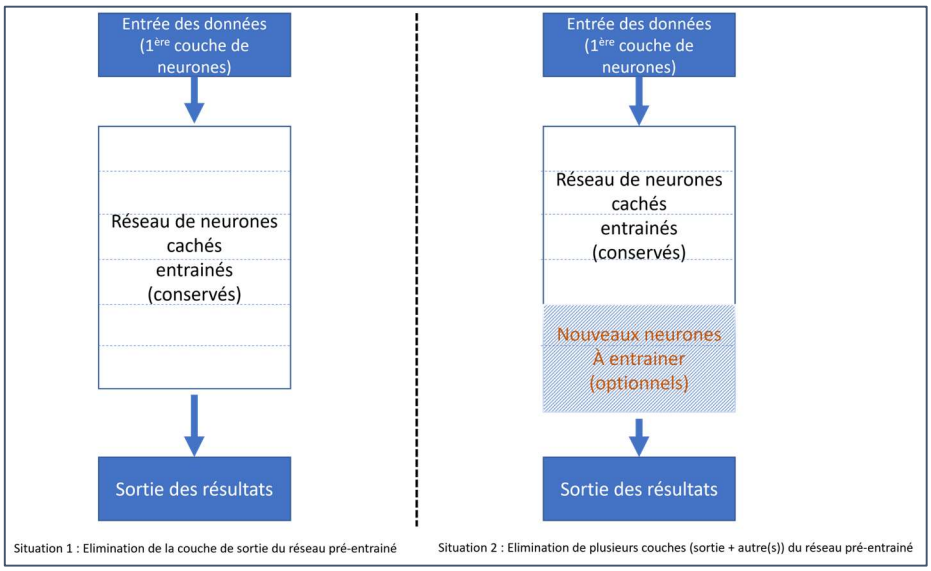


Figure 13 Apprentissage par transfert

Cette possibilité d'ablation des couches de neurones terminales d'un modèle déjà validé, permis de développer l'idée de ne pas remplacer les couches retirées pour ne conserver que les analyses intermédiaires afin de servir un autre une autre technologie d'apprentissage automatique. Cette technique d'extraction par DL de caractéristiques profondes, dénommée technique d'extraction de deep features (DF), est particulièrement intéressante dans le cadre de la radiomique.

C'est, d'ailleurs, ce dernier principe qui sera exploité au cours de notre travail pour l'obtention d'un modèle cohérent en permettant l'extraction automatisée par DL de ces caractéristiques (DF) utiles, devant habituellement être réaliser « artisanalement », dans le but d'être traitées par des modèles d'apprentissage non DL.

3.2 Apprentissage profond en médecine

3.2.1 Publications scientifiques

Une recherche rapide, sur le moteur de recherche de données bibliographiques spécialisée dans la biologie et la médecine PurMed (PubMed, s.d.) avec les termes « deep learning » et « image » comme mots-clefs, montre l'intérêt grandissant ces dix dernières années sur l'apprentissage en profondeur et son intérêt en médecine (figure 14).

Sur les 27'276 études publiées, 82.7% ont été publiées c'est 3 dernières années.

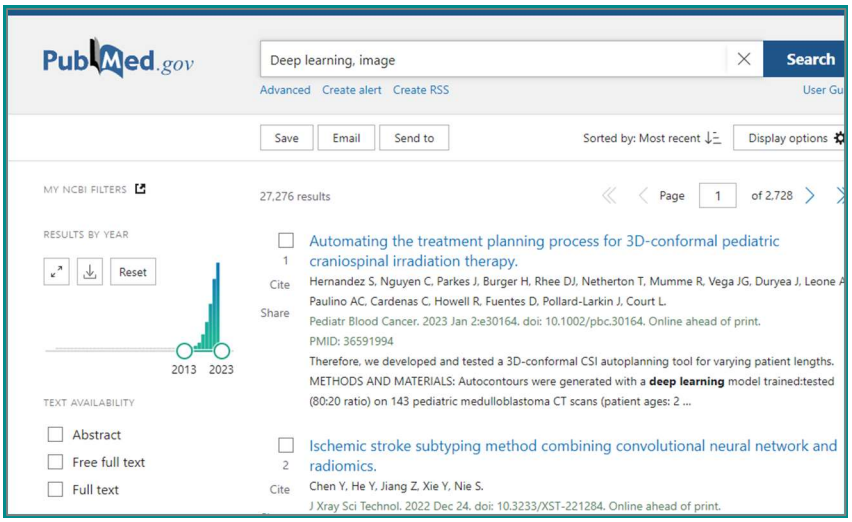


Figure 14 Nombre de publications PubMed sur "deep learning" & "image"

Une autre recherche, sur PubMed, avec « deep learning » et « radiomics » (figure 15) montre que l'imagerie computationnelle (i.e., la radiomique) est un secteur de niche avec une représentation de seulement 3.5% des publications sur le DL en 2022.

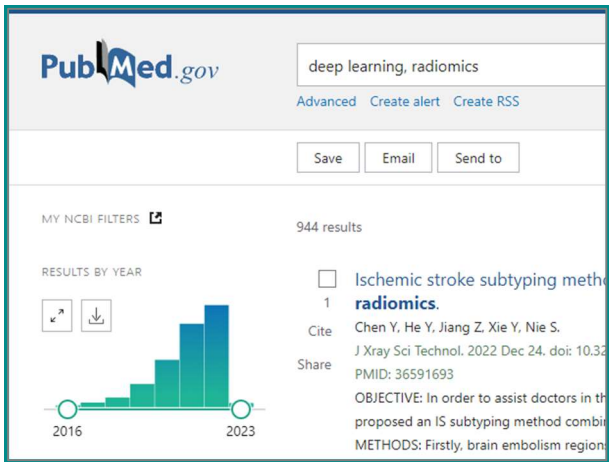


Figure 15 Nombre de publications PubMed sur "deep learning" & "radiomics"

Et cela malgré une apparition en 2012 (Lambin P, 2012) concomitante avec les outils de DL performants.

Cette extraction à haut débit de caractéristiques d'imagerie quantitatives ou de texture, comme le défini Vishwa & Michael A (2016), pourrait tout à fait bénéficier des modèles de DL à neurones à convolution permettant de palier les limites de l'extraction "artisanale" habituelle. D'autant plus que ses extractions concernent la distribution spatiale des intensités de signal, des relations entre les pixels et des informations texturales (van Timmeren, 2020).

Ainsi, l'apport que pourrait avoir la radiomique en recherche clinique et médecine personnalisée est reconnue (Vishwa, 2016) mais elle a des limites qui sont l'extraction des caractéristiques radiomiques (van Timmeren, 2020) et la faible population de patients dans les études de recherche en radiomique.

3.2.2 Applications en analyse d'images médicales

L'analyse conventionnelle des images médicales fait appel à tout un ensemble de compétences pour un clinicien en médecine (visualisation, discrimination, analyse, synthèse et connaissances), nous avons vu précédemment que la très grande majorité des outils embarquant de l'IA validés en médecine sont des outils radiologiques et que les performances des modèles de DL à CNN ont un intérêt majeur.

Ainsi, la technologie d'apprentissage profond en imagerie médicale permet aujourd'hui de réaliser plusieurs tâches complémentaires voir plus performantes qu'un spécialiste dans ce domaine.

La figure 16 nous montre que le DL est utilisé pour un grand nombre de tâches utiles en imagerie médicale, qui sont :

- le pré-traitement : qui consiste en l'application de transformation, de filtre, de la modification de la taille et/ou d'une normalisation de l'échelle de l'image, afin de faciliter et d'améliorer la qualité et la précision des résultats futurs ;
- la reconstruction d'images d'examens médicaux : qui consiste qui « assembler » des images en deux dimensions (2D) (provenant de tomodensitométrie (scanner), d'échographie, de radiographie ou d'imagerie par rayonnement magnétique (IRM)) pour obtenir une image en trois dimensions (3D)
- le débruitage : terme utilisé aussi en photographie qui consiste à éliminer des pixels indésirables qui perturbe la lisibilité de l'image. Par exemple, éliminer un artefact ou un halo présent sur l'image ;
- l'amélioration de la qualité des images : pour ce faire, un mécanisme d'ajout de pixel ou de modification de pixel pour par exemple déflouter une image ;
- la segmentation d'organe et de lésions : qui consiste en une opération de délimitation et matérialisation des contours (par une ligne artificielle dessinée sur l'image) de l'organe ou de la lésion ;
- la détection de lésion : qui vise à déterminer si un élément anormal est présent sur le cliché et cibler sa localisation ;
- la classification : qui, comme son nom l'indique, a pour but de classer une information dans une catégorie prédéterminée (e.g., malade / sain). Elle permet, en autres, de déterminer le diagnostic d'une pathologie ;
- l'observation médicale : appelée aussi suivi médical, il consiste en l'observation et la comparaison de constantes médicales (paramètres biologiques) afin de déterminer si une aggravation ou une progression d'une pathologie est en train de se produire ;

- la prédiction : cette tâche, particulièrement délicate, consiste à faire un (ou des) pronostique(s) sur le futur d'une maladie traitée ou non.

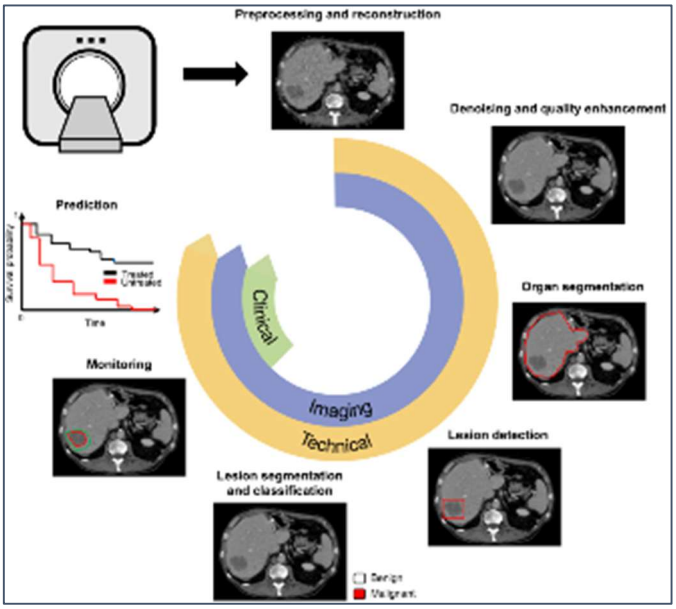


Figure 16 Utilisations potentielles du DL dans le traitement des images médicales
Source : Montagnon, E et al. (2020) "Deep learning workflow in radiology: a primer." Insights into imaging,11(1).

3.3 Conclusion

Les outils d'IA de DL, dont la précision repousse les limites physiologiques humaines, permettent d'avoir des résultats plus précis et plus fiables que l'analyse conventionnelle par un professionnel de santé ce qui promeut leurs utilisations en radiologie dans diverses tâches de manipulation et d'analyse d'images médicales.

Mais comme nous l'avons vu, de manière générale, les modèles de DL sont extrêmement gourmands en données pour leur phase d'apprentissage.

Pour pallier cet inconvénient et afin de pouvoir tirer le meilleur parti des modèles de DL, nous avons à disposition la méthode d'apprentissage par transfert dont les modèles sources permettent de traiter des images médicales et d'en extraire des DF utilisables par les outils « simples », déjà éprouvés, de ML.

Ainsi, la méthodologie d'extraction de DF permettrait d'ouvrir une porte pour intégrer de manière sensée le DL dans les études en radiomiques qui s'appuie sur de faibles populations de patients.

4 Analyse des besoins

4.1 Modèle d'apprentissage profond en accès internet

La radiomique se basant sur l'analyse d'image médicale et l'extraction de caractéristiques, liées aux pixels impliquant des notions telles que la texture, à analyser pour en tirer des conclusions, l'automatisation de ce travail d'extraction par une technologie de DL paraît assez pertinente.

Cependant, l'expertise nécessaire pour développer et mettre en place un tel outil, dépasse bien souvent les compétences des experts en médecine.

A contrario, le développement d'un outil DL efficace en recherche médicale par les spécialistes en technologie de l'information, nécessite des compétences leur faisant trop souvent défaut.

De surcroît, les modèles DL performants doivent pouvoir être accessible aisément et exploitable facilement par moult équipes de recherche et de chercheurs parfois disséminés sur plusieurs sites. D'autre part, ces modèles doivent pouvoir être géré le plus aisément possible par nos spécialistes en technologie de l'information impliquant une notion de centralisation.

D'autre part, la radiomique analysant des caractéristiques particulières, l'intégration avec un outil déjà existant, tel que QuantImage-V2, permettra de valider les caractéristiques radiomiques mises à disposition par le modèle de DL et pourquoi pas de redéfinir ces caractéristiques avec une granularité plus fine des DF.

En conséquence, l'étude de la faisabilité d'une mise à disposition de modèles d'apprentissage profond, avec accès et utilisation facilités grâce à une interface web, ainsi que

l'étude de la possibilité de l'intégrer judicieusement dans le flux de traitement d'une plateforme déjà existante (i.e. QuantImage-V2), nous paraît pertinente.

4.2 QuantImage-V2 (QI2)

Bien que n'étant pas le sujet principal de notre travail, il n'en reste pas moins le point de départ de ce dernier et une ligne directrice dans la volonté de fournir un outil performant pour les recherches en médecine personnalisée et en radiomique.

La mise en place d'une nouvelle technologie intégrable dans des plateformes fonctionnelle impose toute une réflexion qui passe par la compréhension du processus métier afin d'avoir un outil utile et exploitable.

Ainsi, l'étude de l'exemple QI2 nous permet de nous placer dans ce contexte.

4.2.1 Origine et fonctionnement

Sur la page internet d'information de QuantImage-V2 (quantimage-v2-info, s.d.), QI2 est décrite comme un projet développé par la HES-SO en collaboration avec le Fond National Suisse (SNF) et le Swiss Personalized Health Network (SPHN) et soutenu par la fondation Hasler afin de mettre à disposition un outil unique pour la recherche en radiomique permettant de vérifier la pertinence de la radiomique pour ce qui est du diagnostic, du pronostic et de la prédiction (survie) dans des contextes d'oncologie bien définis (e.g., patients atteints de cancer pulmonaire).

Afin de satisfaire à cet objectif, QI2 prend en charge le travail de bout en bout d'une étude radiomique type.

Selon Abler Daniel & al. (2023), il offre un accès en ligne à la gestion des cohortes de patients, à l'extraction de caractéristiques radiomiques (texture, intensité, contour (figure 17)) à partir de scanner médicaux (CT), PET-Scan (PET) et IRM. Il permet aussi l'exploration de ses caractéristiques à l'aide d'un outil de visualisation (figure 18), et permet aussi le développement et l'évaluation "sans code" de modèles d'apprentissage automatique non deep learning afin de définir statistiquement, sur la base des résultats spécifiques aux patients, le niveau de survie.

Tout ceci se fait en s'appuyant sur des outils déjà éprouvés et sur des bibliothèques open-source (quantimage-v2-info, s.d.) dont on citera pour l'intérêt de notre travail :

- OKAPI qui permet de transformer les séries d'images DICOM (bidimensionnelles) en images NIFTI (tridimensionnelles) afin d'intégrer la notion de volume dans l'analyse des images ;
- Pyradiomics qui permet l'extraction de toute une liste de caractéristiques et sous-caractéristiques radiomiques (120 en totalité) (figure 19)

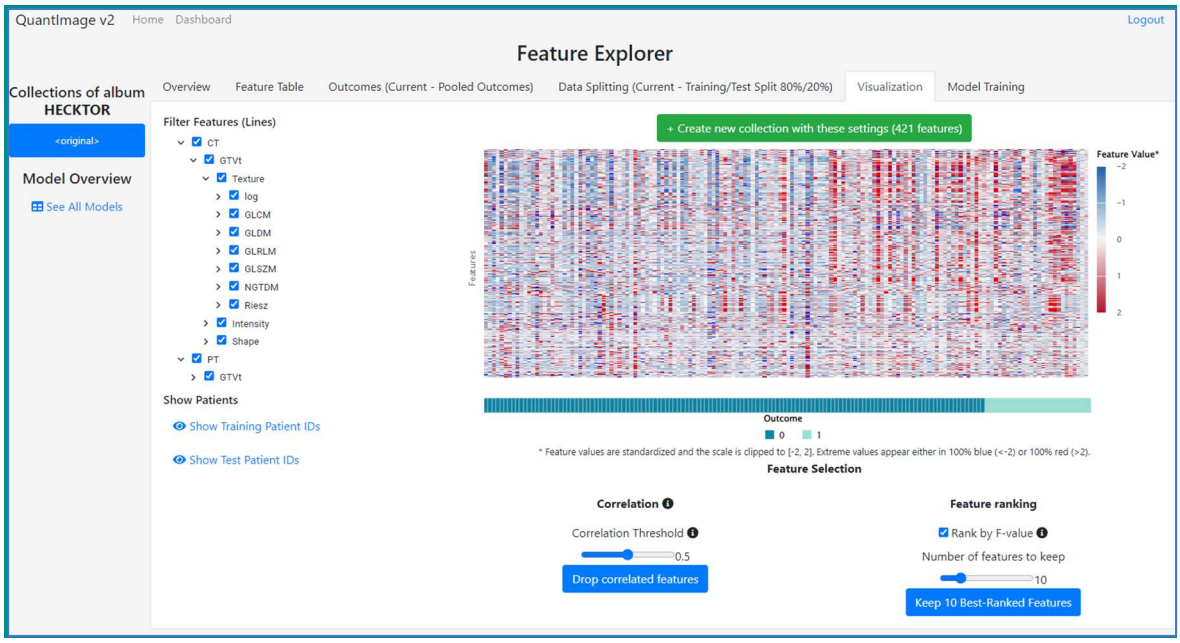


Figure 17 QuantImage-V2 - Gestion caractéristiques radiomiques extraites

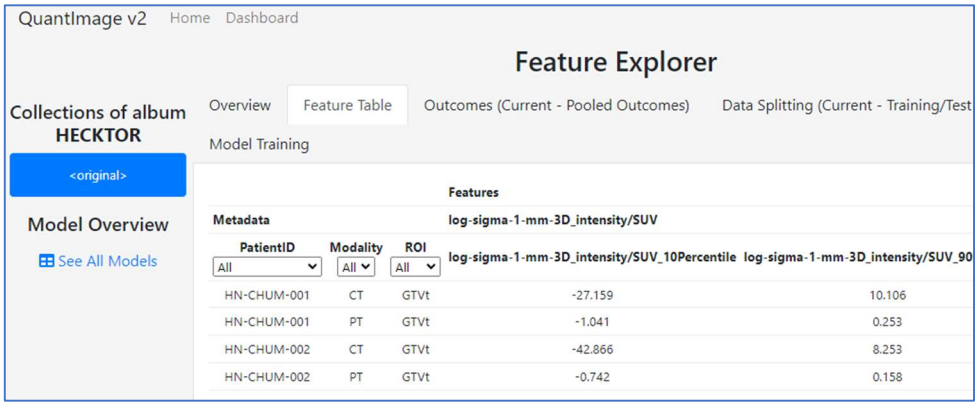


Figure 18 QuantImage-V2 - Visualisation des caractéristiques radiomiques extraites

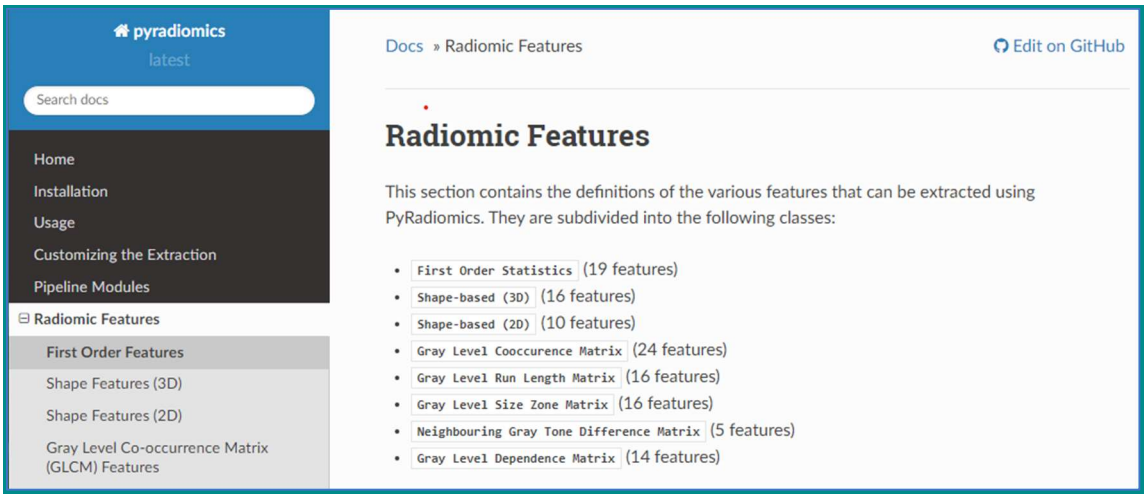


Figure 19 Caractéristiques radiomiques extraites avec Pyradiomics

4.2.2 Intérêts et limites

La plateforme QuantImage-V2 étant développée par une équipe de spécialistes en technologie de l'information en coordination avec des spécialistes en recherche médicale en radiomique permet de mettre en commun les compétences et l'expertise de ces deux corps de métier totalement distincts afin d'avoir, aujourd'hui, un outil performant répondant aux besoins de la recherche clinique en radiomique.

Cependant, les outils de machine learning mis à disposition utilisent une technologie dite « simple » non deep learning pour faire les analyses des caractéristiques radiomiques. Ce qui est tout à fait adapté à la situation de ce type de recherches qui exploitent peu de données (peu de patients).

De plus, l'extraction des caractéristiques radiomiques pouvant être gérée par l'intermédiaire de nombreux paramètres (figure 20), la justesse de leur réglage semble difficile à obtenir et paraît fastidieuse, voire hasardeuse.

D'autre part, et toujours dans un souci de recherche scientifique, il pourrait être intéressant de faire des analyses de deep features (DF) à un niveau de détails plus fin ou différent (défini par les chercheurs en médecine) des caractéristiques radiomiques standards.

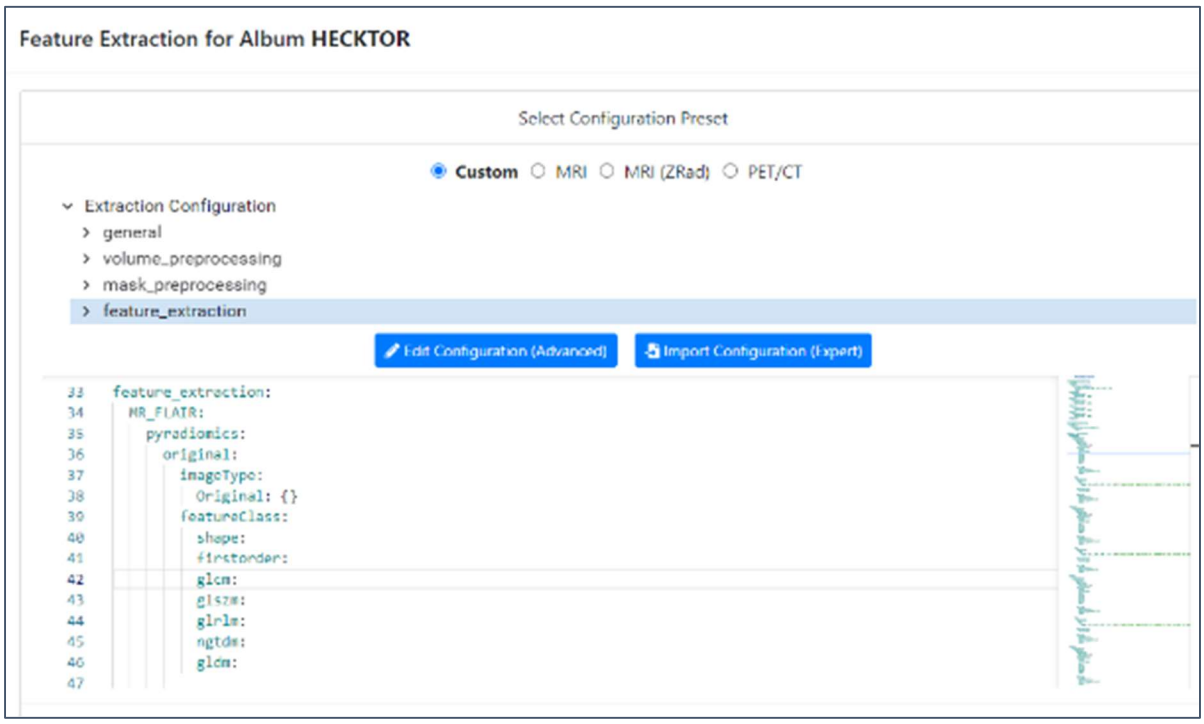


Figure 20 QuantImage-V2 - Paramètres d'extraction des caractéristiques radiomiques

4.2.3 Ouverture vers l'utilisation d'un modèle d'apprentissage profond

A partir de ce constat concernant le côté fastidieux et approximatif d'extraction des paramètres utiles, et avec nos connaissances sur l'apprentissage profond, il paraît intéressant de se servir de cette technologie pour extraire de manière plus performante les caractéristiques radiomiques ou d'autres caractéristiques pertinentes (deep features).

Le développement de tels modèles d'apprentissage demandant des compétences particulières, comme pour la technologie d'analyse actuellement à disposition, ces modèles seraient déterminés en amont par des experts dans le domaine, en tenant compte des avis experts des médecins pour la définition de la couche de restitution.

Leur utilisation par les chercheurs en médecine permettrait de valider le niveau et la valeur d'apprentissage du (des) modèle(s) DL mis à disposition.

Dès lors, l'introduction de la technologie d'apprentissage profond et, en particulier de deep features dans le cadre de notre travail, trouve tout son intérêt sur la plateforme QuantImage-V2 afin de valider la faisabilité d'une telle intégration à cet outil existant.

5 Cahier des charges

5.1 Qualités fonctionnelles

L'outil développé devra posséder diverses capacités afin de réaliser certaines actions techniques obligatoires :

- une relation client-serveur

Notre service d'inférence devra permettre aux utilisateurs de pouvoir interroger un modèle d'apprentissage profond à distance à l'aide d'une connexion internet ;

- la mise à disposition aisée des modèles entraînés

L'utilisateur devra pouvoir sélectionner le modèle qu'il souhaite utiliser parmi un nombre de modèles disponible qui lui seront présentés ;

- l'inférence fonctionnelle et des résultats accessibles

De manière assez évidente, nous devons faire en sorte que le modèle, auquel il est possible d'accéder, réalise l'analyse pour laquelle il a été conçu et que ces résultats soient rendus accessible pour l'utilisateur qui l'a interrogé ;

- la prise en charge des images médicales NIFTI (type et format)

Le dispositif doit pouvoir bénéficier du serveur d'images de l'hôpital (PACS) par l'intermédiaire de la plateforme Kheops, ainsi que de la fonctionnalité, offerte par OKAPI, de transformation des images bidimensionnelles des examens médicaux sources (format DICOM) en images tridimensionnelles (format NIFTI) afin d'en extraire les caractéristiques (deep features) à l'aide du modèle DL ;

- une Interface web à disposition

Dans l'optique de déporter la contrainte d'apprentissage de nouvelles technologies de l'information, nous souhaitons mettre en place une interface visuelle et coutumière pour les utilisateurs non-experts dans l'IT ;

Dès lors, il nous faudra mettre en place un service d'inférence avec une interface graphique ou alors ajouter cette interface si elle n'est pas disponible nativement sur l'outil choisi ;

- la possibilité d'être intégrée dans une plateforme existante (QuantImage-V2)

Ici, ayant choisi la plateforme existante QI2 comme plateforme de référence, il faut que l'intégration reste pertinente.

Ainsi, les limites de la technologie DL ne permettant pas de faire une analyse complète aboutissant à un résultat final d'analyse radiomique, nous avons comme option et volonté de pouvoir intercaler notre outil entre le traitement des images 3D et la présentation des caractéristiques servant à l'analyse conventionnelle déjà en place. Ainsi, comme le représente la figure 21 les utilisateurs pourront choisir d'utiliser ou non les caractéristiques extraites par DL.

N.B : la figure 21 est présentée uniquement titre d'exemple de développement future possible sur la plateforme QuantImage-V2.

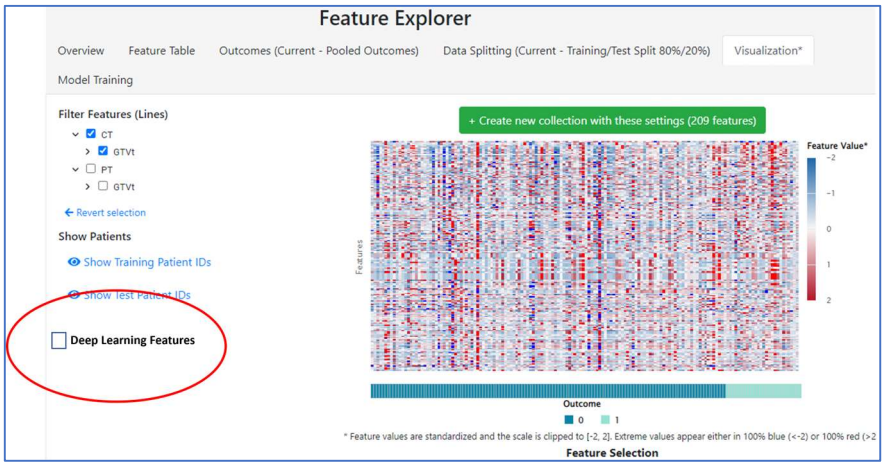


Figure 21 Hypothèse d'intégration visuelle future des DF dans QunatImage-V2

5.2 Qualités non fonctionnelles

Les premières qualités que nous souhaitons avoir seraient une installation, une maintenance et une optimisation facilitée. D'autre part, nous aimerions avoir un moyen d'analyse des performances d'utilisations des modèles et donc d'avoir un moyen de surveillance des performances de l'outil.

Et enfin, le dispositif mis en place doit être sécure, fiable et durable avec une certaine assurance dans le suivi de son développement et de ses mises à jour.

5.3 Adaptation du dispositif

5.3.1 Cas d'usage

Pour notre cas d'usage, nous définissons le contexte suivant :

"Nous sommes une équipe de spécialistes en technologie de l'information travaillant déjà avec des équipes de spécialistes en recherches médicales. Ces derniers utilisent une technologies d'apprentissage automatique non-profond et souhaitent mettre en place une technologie d'apprentissage profond pour leur recherche."

Ainsi, nous définissons les action s suivantes :

- Action 1 : mettre en place un environnement permettant d'utiliser (déployer) la technologie DL ;
- Action 2 : définir des modèles DL pertinent pour les travaux de recherche ;
- Action 3 : mettre à disposition ces modèles DL sur l'environnement de déploiement DL ;
- Action 4 : les chercheurs devront accéder à l'inférence de ces modèles ;
- Action 5 : les chercheurs devront pouvoir exploiter les résultats de l'inférence des modèles ;

5.3.2 Diagramme de séquence

La figure 22 détaillant la séquence reflétant notre cas d'usage, nous montre qu'une première étape, de préparation de l'environnement de DL avec la mise en place d'un outil d'inférence et d'un modèle de deep learning entraîné et recevant en entrée un format de données identique au format mis disposition par les experts médicaux, a été réalisée. Cette étape regroupe : la mise à disposition des images par les experts en médecine, l'installation de l'outil d'inférence, l'implémentation (adaptation) du modèle de DL, la phase d'analyses avec tests et validation du modèle avec la visualisation des résultats obtenus et la confirmation de la validité du modèle.

La seconde étape concerne, l'accès au service d'inférence par un utilisateur, qui doit se connecter, puis sélectionner les examens à analyser, puis choisir le modèle souhaité, puis consulter les résultats et faire un retour aux experts en informatique pour améliorer l'ensemble du dispositif si besoin.

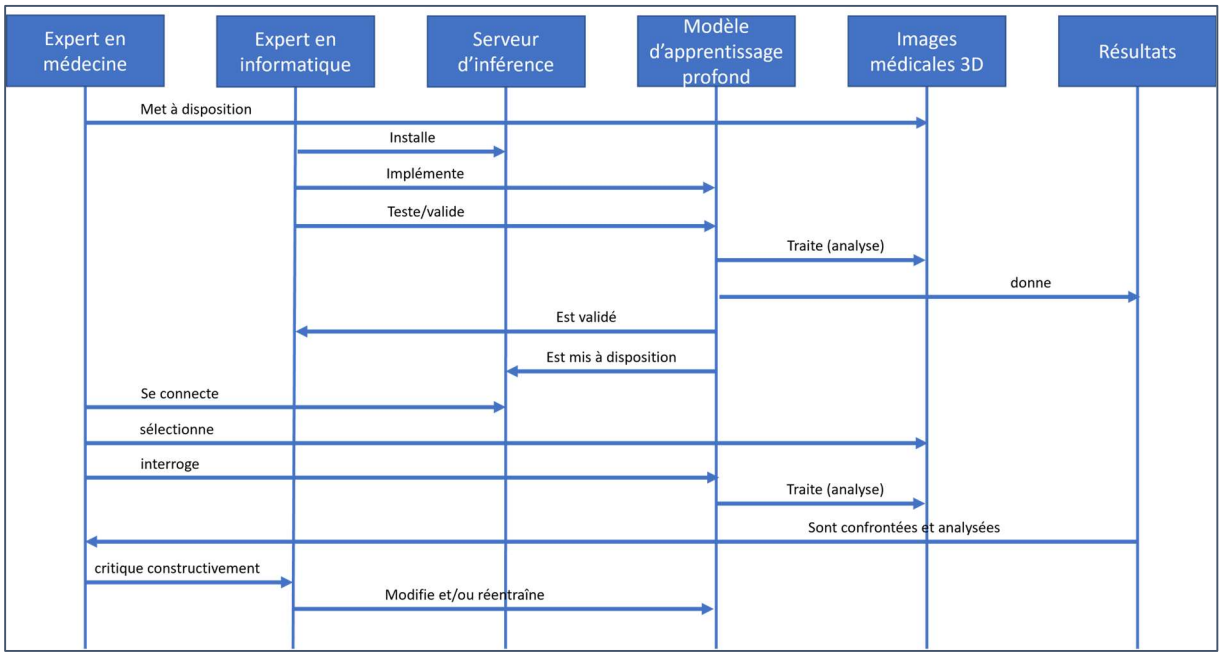


Figure 22 Diagramme de séquence

5.3.3 Schéma de l'environnement

N.B : l'environnement de travail représentant les composants utilisés, ce schéma (figure 23) est construit avec la dénomination de l'outil d'inférence (Triton Nvidia®) dont on détaillera les raisons de son choix dans le chapitre suivant.

Le détail de cet environnement de travail sera développé plus tard dans ce document mais on peut relever qu'il est composé de deux entités ; soit, un ordinateur local hébergeant l'outil de développement individuel (IDE) et un navigateur internet relié (par une connexion sécurisée SSH) à un environnement distant (un serveur) hébergeant 2 environnements virtuels. L'accès et l'interaction avec cet environnement distant se font par ligne de commande.

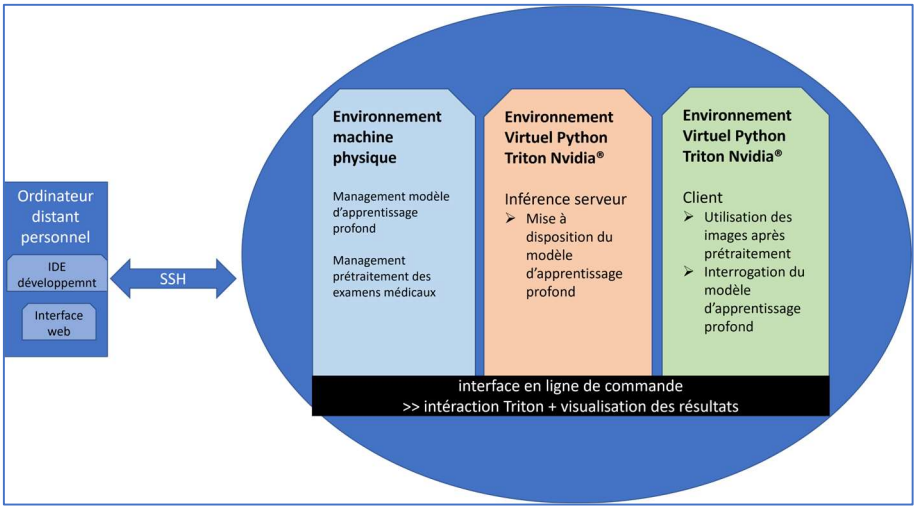


Figure 23 Environnement de travail (schématisation)

5.3.4 Chaine de traitement

Pour ce travail, nous pouvons définir une chaîne de traitement (pipeline) décrivant la place de l'outil d'inférence dans le flux de son utilisation et ses relations avec les différents intervenants (figure 24).

Tout d'abord, on peut distinguer deux groupes ayant des interactions avec le serveur d'inférence. Le groupe de professionnel en IT qui va élaborer et mettre en place un modèle

de DL validé, d'un côté. Et de l'autre côté, le groupe d'utilisateurs (i.e. les spécialistes en recherche médicale) qui va bénéficier l'inférence DL par l'intermédiaire d'une plateforme.

On peut aussi distinguer un autre élément important pour une réussite d'un tel projet, c'est la relation entre les informaticiens et utilisateurs du modèle DL.

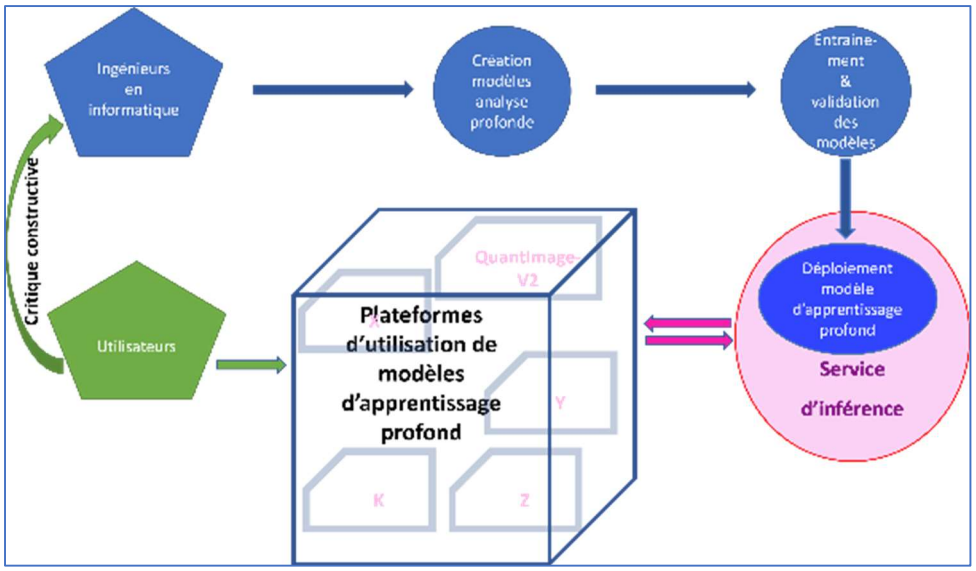


Figure 24 Chaine de traitement du service d'inférence

6 Gestion du projet

Notre projet s'est déroulé de manière agile en appliquant au tant que possible la méthodologie SCRUM.

Dès lors, nous avons organisé les itérations de ce projet en 7 sprints que nous avons répartis entre les mois de d'octobre et de janvier- Durant ces sprints nous nous sommes fixés des objectifs à réaliser matérialisés par des user stories définies dans le Product Backlog de notre projet (annexe 1).

- Sprint 1 :

La première étape de notre projet fut de définir notre cadre de développement en définissant le cas d'usage, les technologies et les outils qui seront utilisés.

- Sprint 2

Durant cette itération, nous avons installé notre environnement de travail et de développement et nous avons déterminé le modèle DL que nous allions utiliser.

Pour ce dernier point, nous avons fait appel à un spécialiste du domaine DL.

- Sprint 3

Ici, nous avons eu une phase d'appropriation et adaptation du modèle DL ainsi que l'installation de l'outil d'inférence que nous avions défini durant le sprint 1.

Afin de préparer la future utilisation de cet outil, nous avons constitué un ensemble de données d'examens d'imagerie médicale test 2D, utilisable par le modèle DL sélectionné.

- Sprint 4

Ce sprint fut dédié à l'intégration du modèle DL dans l'outil d'inférence, au développement d'un programme informatique permettant de sélectionner des images médicales de notre ensemble test, de les transformer en images 3D et d'exécuter l'inférence en utilisant le modèle DL.

- Sprint 5

Durant cette cinquième itération, nous avons implémenté une fonction d'affichage des résultats de l'analyse par le modèle DL.

Toutes les itérations précédentes concernant l'utilisation de l'outil d'inférence n'ayant pu se faire qu'avec une interface texte en ligne de commande, nous avons défini les éléments pour la création d'une interface web et nous avons débuter son implémentation jusqu'à avoir accès à notre ensemble de données test.

- Sprint 6

Au cours de ce sprint, le développement de l'interface web a suivi son cours pour permettre la sélection des images 2D, leur transformation en image 3D, l'exécution de l'inférence de notre modèle DL et la visualisation des résultats dans notre navigateur Internet.

- Sprint 7

Ce dernier sprint était dédié au développement de la WebApp pouvant interagir avec Kheops, par l'intermédiaire de la connexion sécurisée utilisée pour QuantImage-V2 (fourni par une API nommée Keycloak) (figure 55).

Le but de ce développement était de simuler l'intégration du serveur d'inférence Triton dans le flux de QuantImage -V2.

7 Choix technologiques et techniques

7.1 Choix du service d'inférence Triton NVidia® (Triton)

La mise à disposition d'un modèle de deep learning par l'intermédiaire d'une interface internet, comme pour tout élément « consommable » sur Internet, peut se faire par de multiples moyens ; certains plus judicieux ou pertinents que d'autres.

Ainsi, pour mettre à disposition des modèles DL, il existe des outils dédiés exploitants diverses méthodologies et offrants des options de paramétrages et d'ajout de fonctionnalités différentes.

Ainsi, nous avons dû faire un tour d'horizon et analyser les différents moyens existants afin de choisir un outil (i.e. Triton) en adéquation avec les objectifs et qualités que nous lui souhaitions.

7.2 Démarche

L'utilisation d'un modèle d'apprentissage profond peut se faire de manière locale et indépendante, c'est-à-dire que le modèle est installé sur un ordinateur et accessible uniquement sur cet ordinateur, ou alors le modèle peut être partagé. Dans l'optique de notre travail, nous n'étudierons que cette deuxième possibilité.

Notre choix pour la technologie de déploiement de modèles DL a été conditionné en fonction de plusieurs critères (annexe 2).

L'élaboration de tels modèles pouvant s'effectuer dans différents langages de programmation, une notion d'interopérabilité apparaissait importante afin que l'outil puisse accepter divers modèles construits avec les langages de programmation et les environnements les plus couramment utilisés actuellement, que sont : PyTorch, TensorFlow/Keras, TensorRT, ONNX et XGBoost.

Par ailleurs, autre point crucial pour se prononcer, en faveur d'un outil ou d'un autre, fut son accessibilité ainsi que celle offerte aux modèles. Plus un outil sera aisé d'accès, plus facile seront les conditions d'utilisation.

Ainsi, parmi les possibilités d'interagir avec le dispositif, nous avions comme impératif, un échange d'information avec une API REST car c'est ce qui est actuellement en place avec QuantImage-V2 ; et aussi parce que c'est une technologie très usitée.
Une autre technologie appelée gRPC, développé par Google® et permettant des échanges plus rapides, était aussi prise en compte dans l'idée d'un développement futur mais en moindre mesure.

Outre ces deux caractéristiques précédentes, il fallait aussi prendre en considération le système opérationnel du serveur qui pouvait accueillir cet instrument de déploiement des modèles d'apprentissage. La HES-SO ayant à disposition des serveurs exploitant un système Ubuntu, cet environnement devait en premier lieu être considéré.
Bien entendu, que tout autre environnement additionnel accepté par l'outil permettrait d'avoir une souplesse d'hébergement appréciable.

Dans ce même ordre d'idée, de largesse d'utilisation, la possibilité d'installer cet instrument sur une machine (serveur) physique ou virtuel, voir dans un environnement Cloud (Azure, AWS, …), était pris en compte.

En plus de cet environnement d'exploitation, les modèles de deep learning consommant beaucoup de ressources du système (e.g., processeur central (CPU), processeur graphique (GPU)), l'outil devait pouvoir utiliser et si possible gérer la charge de travail imposée sur les CPUs et les GPUs.

En plus de cet environnement d'exploitation, les modèles de deep learning consommant beaucoup de ressources du système (e.g., processeur central (CPU), processeur graphique (GPU)), l'outil devait pouvoir utiliser et si possible gérer la charge de travail imposée sur les CPUs et les GPUs.

Tous ces éléments faisaient partie de nos critères de base et toute fonctionnalité additionnelle était un plus et a conditionné notre décision.

Ainsi, nous avons défini, en tout, 19 critères répartis en 7 catégories qui sont :

- la catégorie « apprentissage profond », avec 2 sous-catégories :
  - déploiement ;
  - gestion d'inférence ;
- la catégorie « environnement », avec 3 sous-catégories :
  - machine locale ;
  - machine virtuelle ;
  - Cloud ;
- la catégorie » voies d'accès », avec 3 sous-catégories :
  - HTTP (REST) ;
  - gRCP ;
  - Offline (utile pour de la maintenance par exemple) ;
- la catégorie « conteneur » (qui se suffit à elle-même) ;
- la catégorie « langage/plateforme d'apprentissage profond supportée », avec 5 sous-catégories (dans l'idée de qui peut le plus, peu le moins) :
  - langage 1 ;
  - langage 2 ;
  - langage 3 ;
  - langage 4 ;
  - langage 5 ;
- la catégorie « système d'exploitation » (supportant l'installation de l'outil), avec 4 sous-catégories :

- Ubuntu (condition sine qua non) ;
- Windows
- Mac OS
- Linux

- Et la catégorie « composants additionnels »

7.2.1 Etat des lieux des moyens actuels

Après avoir déterminé les différentes caractéristiques souhaitées pour notre instrument de déploiement des modèles de DL, nous avons répertorié les différentes technologies pertinentes en vue de notre utilisation, permettant une mise à disposition de modèles de DL sur le web. Ainsi, nous avons retenu les serveurs d'inférence dédiés aux modèles d'apprentissage, les serveurs informatiques web pour les nombreuses possibilités de développement d'outils permettant d'utiliser les modèles et de les rendre accessibles sur le web, et le cloud dont certains offrent des services spécialisés dans l'inférence.

7.2.1.1. Les serveurs d'inférence

Un serveur d'inférence est un type de serveur qui est conçu pour exécuter des modèles d'apprentissage automatique déjà entraînés afin de produire des résultats prédits. Ces serveurs sont conçus pour permettre un accès rapide et facile aux algorithmes d'apprentissage automatique et sont généralement utilisés pour la surveillance et le traitement des données en temps réel.

7.2.1.2. Les serveurs informatiques

On peut aussi déployer les modèles sur Internet par le biais d'un serveur informatique.

Ici, les modèles sont hébergés sur le serveur internet et les requêtes des utilisateurs sont transmises par le biais d'une interface de programmation d'application (API) REST ou SOAP développée à cet effet.

Une autre alternative de déploiement sur un serveur web serait d'utiliser des outils tels que Flask, Django ou Node.js ou à l'aide d'un service de déploiement de modèles, comme TensorFlow Serving, Algorithmia.

Ce type de développement a été pris en considération pour notre analyse en y intégrant le service TensorFlow Serving mais nous l'avons écarté, pour notre étude, l'analyse impliquant le développement d'une API spécifique pour l'inférence des modèles.

7.2.1.3. Le Cloud

Les services d'hébergement, appelés services de cloud computing, offrent, pour certains, une variété d'outils et de services pour le déploiement des modèles de deep learning.

Ils proposent en particulier, des outils de développement, des modèles pré-entraînés, des bibliothèques de modèles, des services de surveillance,...

Parmi les fournisseurs de tels services, on retrouve Amazon Web Services (AWS) qui met à disposition Amazon SageMaker, Google Cloud Platform (GCP) avec Google AI Platform ou alors Microsoft Azure et son Microsoft Azure Machine Learning.

Ces services d'hébergement pourraient avoir un intérêt mais pas dans l'immédiat du fait des frais imposés pour leur utilisation qui seraient à répercuter sur les utilisateurs.

Ainsi, étant écarté en tant que choix d'étude, nous avons tout de même décidé de prendre en considération la possibilité de tel ou tel autre outil pouvant être hébergé sur ses plateformes.

7.2.2 Justification du choix de Triton

En fonction de toutes les caractéristiques définies et des outils sélectionnés, afin de faire notre choix, nous avons comparé les outils en fonction du nombre de caractéristiques qu'ils incluaient (figure 25) . Pour ce faire, nous avons donné un score de « 1 » si la caractéristique était dans l'outil et de « 0 » sinon (annexe 2). Et concernant la sous-classe Ubuntu, nous avons exclu l'outil s'il ne l'honorait pas.

Il en est ressorti que le serveur d'inférence Triton NVidia® (nommé Triton, ci-après) est sorti grand gagnant de ce classement en cochant toutes les cases (19/19) (figure 25).

De plus, cet outil peut héberger d'autres outils tels que KServe (KuberFlow) qui était deuxième avec 16 points sur 19 selon notre analyse.

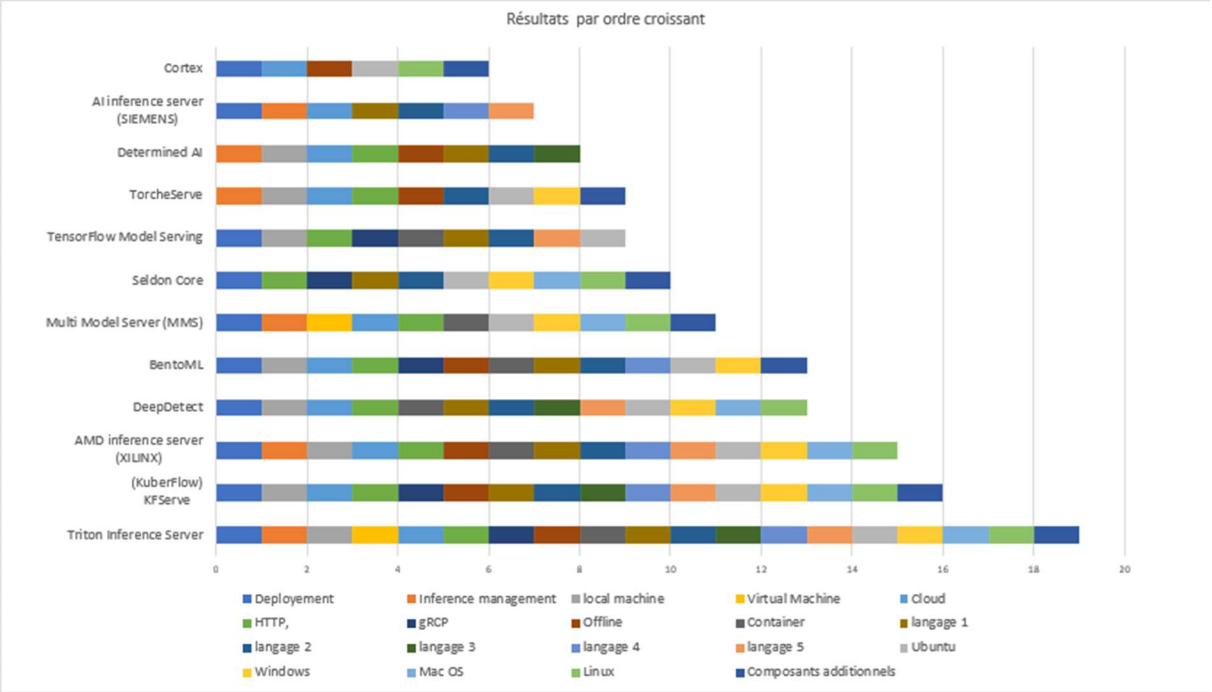


Figure 25 Comparaison des outils d'inférence par ordre croissant d'intérêt

Ainsi, Triton Nvidia® fut sélectionné pour ce travail car il offre :

- une prise en charge d'un large éventail de modèles qui pourra offrir une interopérabilité entre des modèles développés dans divers langages et divers environnements ;
- des possibilités de connexions à Triton sont multiples et de nombreux protocoles sont supportés. En premier lieu, le protocole REST mais aussi gRPC ;
- la flexibilité grâce au nombre de systèmes d'exploitation, dont Ubuntu, pouvant le supporter mais aussi la flexibilité au niveau matériel. L'étude nous démontrant qu'en fonction du type de ressources (CPU et/ou GPU) de la machine hôte, les outils ne sont pas toujours compatibles, ici, Triton offre une compatibilité avec les CPUs intel et ARM en plus des GPUs Nvidia ;

- une capacité d'optimisation des performances des ressources du système telles que l'accélération matérielle des GPU NVidia et la gestion de l'utilisation CPUs/GPUs si les deux entités sont présentes. Ceci doit permettrait d'accélérer et d'augmenter l'efficacité d'exécution des modèles ;
- c'est un service serveur/client qui permet une centralisation de tous les modèles et donc une facilité de maintenance et de gestion d'accès en cas d'utilisation par plusieurs équipes de professionnels différentes ;
- des caractéristiques supplémentaires intéressantes, telles que :
    - la prise en charge de différentes versions d'un même modèle ;
    - une capacité annoncée de prendre en charge des centaines de milliers de requêtes d'inférences à la fois et sa capacité d'effectuer des inférences sur des environnements distribués ;
    - la possibilité d'ajouter des outils additionnels de surveillance de performances des modèles ;
    - la possibilité de mettre en place des fonctionnalités de sécurité par chiffrement, contrôle d'accès et l'isolation des modèles offertes par Triton ;
    - la confiance sur la durabilité de Triton puisque c'est une entité de référence mondiale (i.e., NVidia) qui met à disposition des ingénieurs pour développer et entretenir l'outil ;
    - et enfin, un support technique avec une documentation foisonnante et une assistance technique à disposition.

7.3 Détermination du (des) modèle(s) d'apprentissage profond

7.3.1 Contexte

L'implémentation et l'entraînement d'un modèle complet d'apprentissage profond n'étant pas l'objet de notre travail, nous avons effectué la recherche d'un modèle éprouvé en accès libre.

La spécificité du domaine médicale et des formats d'images tridimensionnelles utilisées nous a obligé à trouver un modèle spécifique adapté.

Dès lors, ce travail de recherche demandant une expertise dans le domaine du machine learning et en particulier en deep learning, afin de trouver un modèle pertinent, de le comprendre et de valider la possibilité de l'intégrer dans notre travail, c'est M Andrearczik Vincent qui l'a effectué.

Une fois le modèle déterminé et « validé » comme compatible avec notre travail, la spécificité de préparation des images nous a imposé de réadapter l'outil de transformation OKAPI en vue « d'alimenter » le modèle de DL avec des données comparables aux données initialement prévues pour ce modèle.

Ainsi, en aparté, nous avons pu vérifier un premier élément important dans nos décisions de choix de l'outil d'inférence : la flexibilité offerte par le large choix de format et de plateforme de développement des modèles acceptés.

Par ailleurs, afin de préciser les différents formats de modèles possibles pour notre choix, nous rappelons ici, que le serveur d'inférence Triton NVidia accepte les modèles développés en :

- TensorFlow : qui est une plateforme open source (développé par Google en C++) de développement de modèle deep learning pour l'apprentissage automatique et le traitement du langage naturel. Cette plateforme intègre une bibliothèque informatique Keras qui permet de construire des modèles de DL complexes ;
- PyTorch : qui est une plateforme aussi open source (développé par Meta en Python et C++) pour l'apprentissage automatique profond ;
- TensorRT est un ensemble logiciel (framework) open source (développée par NVidia) pour l'apprentissage automatique ;
- ONNX : qui veut dire Open Neural Network eXchange et est donc un environnement de développement d'intelligence artificielle open source (développé par Facebook mais repris par la communauté Linux en langage Python et C++).
- …

7.3.2 Modèle DL choisi

Ainsi, en tenant compte de toutes les spécificités de notre contexte citées précédemment, il a été trouvé un modèle adapté, mis à disposition sur le site internet (Keras, s.d.) de la bibliothèque informatique Keras (et donc la plateforme TensorFlow) à titre de code d'exemple pour la classification en pneumopathologie à partir d'examens d'imagerie médicale.

C'est un modèle à réseau de neurones à convolution (CNN) 3D, constitué de 17 couches de neurones, implémenté afin de prédire la présence d'une pneumonie virale (d'en estimer la probabilité) à partir d'image CT 3D.

Ce qui, concernant le type d'images à analyser, correspond tout à fait à notre situation pour les images CT mais qui devra être réévalué concernant les examens PT et IRM. Pour notre étude, nous nous satisferons de la pleine capacité à analyser les examens médicaux CT.

Cependant, concernant sa finalité de classification, cela ne nous concerne pas.

Au contraire, aspirant à une utilisation possible en recherche médicale en radiomique, nous souhaiterions seulement extraire des caractéristiques (deep features) pertinentes pour qu'elles soient analysées par un modèle d'apprentissage non-profond (par exemple, comme le modèle déjà en activité sur la plateforme QuantImage-V2).

7.3.3 Conclusion

Ainsi, le modèle sélectionné, nous satisfera pour mettre à disposition ce modèle de deep learning via un service web et l'intégrer dans le flux d'une plateforme existante telle que QuantImage-V2.

Cependant, afin de se mettre dans des conditions réalistes, nous devrons nous servir de ce modèle comme base afin d'en dégager un modèle d'apprentissage par transfert, nous permettant d'extraire des deep features qui pourront être utilisées par un modèle de ML traditionnel.

7.4 Environnement de développement

7.4.1 Choix du langage de développement

7.4.1.1. Rapide tour d'horizon des langages à disposition

Comme nous avoir pu le constater dans l'analyse des différents moyens de déploiement des modèles de deep learning, Triton accepte des outils développés en C++, Java et Python.

Ces trois langages de programmation sont des langages très populaires (figure 29)

Le langage C++ (figure 28) est un langage de programmation générique, procédural et orienté objet qui inclue des aspects d'un langage de bas niveau (proche du langage machine binaire) permettant de personnaliser et d'optimiser la compilation du programme développé mais aussi des aspect d'un langage de haut niveau (plus proche de l'humain) avec des fonctions complexes. Le framework TensorFlow a été codé dans ce langage.

Le langage Java (figure 26) est le langage d'apprentissage du bachelor en informatique de gestion à l'HES de Sierre. C'est un langage orienté objet de plus haut niveau que le langage C++ mais incluant quand même des aspects de bas niveau. Il nécessite un kit de développement (JDK) et un environnement d'exécution (JRE). Pour le kit de développement il existe 2 fournisseurs : Oracle™ qui nécessite une licence et OpenJDK qui est openSource et en libre accès.

Et enfin, le langage Python (figure 27) qui est un langage de haut niveau orienté objet. C'est un langage ayant pour réputation d'être facile à appréhender. C'est aussi l'un des langages les plus utilisés dans le domaine du machine learning et le plus populaire en 2022 (figure 29). Il est bon de savoir qu'il co-existe toujours deux versions de Python mais que nous nous concentrerons sur la version Python3 puisque la version Python2 n'est plus maintenu depuis 2020.

Figure 28 Logo C++

Figure 26 Logo Java

Figure 27 Logo Python

TIOBE

Schedule a demo

Dec 2022	Dec 2021	Change	Programming Language	Ratings	Change
1	1		Python	16.66%	+3.76%
2	2		C	16.56%	+4.77%
3	4	▲	C++	11.94%	+4.21%
4	3	▼	Java	11.82%	+1.70%
5	5		C#	4.92%	-1.48%

Figure 29 Classement Tiobe 2022
Source : https://www.tiobe.com/tiobe-index/

7.4.1.2. Décision

L'environnement système exploité étant Ubuntu, le modèle de deep learning choisi étant développé en langage Python3 fournissant d'emblée toutes les librairies nécessaires à l'exploitation de ce langage, nous nous sommes orientés tout naturellement vers le langage de programmation Python.

La version 3.6.9 de Python sera donc utilisé pour le développement de code spécifique à l'utilisation du modèle pour et par le serveur d'inférence Triton Nvidia®.

Cependant, concernant l'implémentation de l'interface web, le langage python a pu être maintenu en grande partie mais l'environnement web nous à imposer d'y joindre des portions de code html et javascript.

7.4.2 Choix de l'environnement de développement intégré (IDE)

7.4.2.1. Rapide tour d'horizon des IDE à disposition

Le choix du serveur d'inférence ayant orienté le choix du langage de programmation vers Python, notre IDE devra lui aussi prendre en charge ce langage.

Une recherche succincte sur Internet nous présentant une liste de meilleurs IDE pour développer en Python, nous donne 3 IDE familiers (PyDev, PyCharm et Visual Studio Code) et 1 apparemment très simple et performant (Jupiter) (tableau 1).

Le choix de l'IDE a été fortement conditionné par le langage de programmation principal (Python3) choisi mais aussi par la possibilité de cumuler plusieurs autres langages de programmation (e.g., html et javascript) ainsi que par l'expérience et l'aisance d'utilisation.

7.4.2.1. Décision

VSCode étant un IDE facile d'utilisation prenant en charge de multiples langages de programmation et offrant une interconnectivité simple à mettre en place (avec le serveur de la HS-SO), il sera choisi en tant qu'environnement de développement intégré.

Tableau 1 Environnements de développement intégré évalués
Source : https://www.commentcoder.com/ide-python/

Logos	Nom de l'IDE	Fournisseur	Pan tarifaire	Environnement de développement
	PyDev	Eclipse Foundation	Gratuit	Fenêtre d'application identique à Eclipse
	PyCharm	JetBrain	- Gratuit pour étudiant - 249.00€ /utilisateur/an	Fenêtre d'application
	Visual Studio Code (VSCode)	Microsoft	Gratuit	- Fenêtre d'application - navigateur internet
	Jupyter	Jupyter	Gratuit	- navigateur internet

7.4.3 Choix de l'interface web

Un bref aperçu lors de notre recherche d'information sur les outils d'inférence, nous a montré que la majorité des interactions avec ces outils se faisait en mode textuel par ligne de commande (cli) à travers le terminal dédié.

Afin d'avoir une interface plus conviviale et plus facile à utiliser, nous souhaitons développer une interface web pour interagir avec un service tel que le serveur Triton.

D'autre part, l'intégration d'une telle interaction permettra de faire évoluer notre travail vers le développement d'un outil avec une architecture 3 tiers intégrant le concept Modèle Vue Contrôleur (MVC) (figure 30). Ce qui permettrait d'ajouter une couche de sécurité et de facilité de maintenance en dissociant les données du moteur (travail) de l'application, le tout disjoint de l'affichage des résultats.

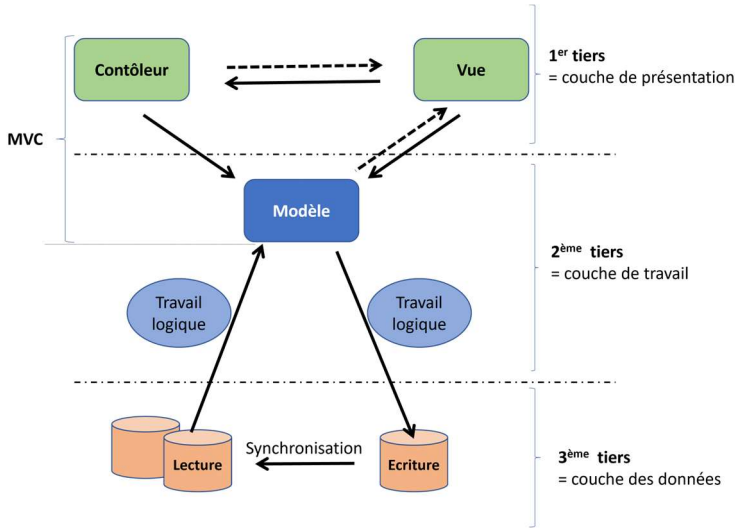


Figure 30 Concept MCV dans une architecture 3 tiers

Pour ce faire, nous avons la possibilité d'implémenter une interface de programmation d'application (API) qui permet de connecter un logiciel ou un service tel que Python avec un autre logiciel afin d'échanger des données et des fonctionnalités.

Ce type d'outil très performant serait très judicieux dans le cadre d'intégration unique dans une plateforme déjà existante comme QuantImage-V2 mais ce n'est pas le but principal de notre travail. En conséquence, même si cette interface pourrait être intéressante, en particulier pour mettre en place une couche de sécurité et de facilité de maintenance, nous avons écarté ce moyen.

L'autre dispositif nous permettant de réaliser cette interface web est de développer une petite application web (WebApp) qui permettrait de proposer un ensemble de services offrant une interaction complète avec le serveur Triton.

Ainsi, nous nous sommes dirigés vers une solution plus simple et plus rapide à mettre en place, grâce à un outil spécialement dédié à cet effet et adapté à notre choix de langage de développement.

Cet outil de développement web en Python3 appelé « Flask » est open source, léger et permet de réaliser des pages Internet.

Cependant, le seul langage Python ne permettant pas de développer intégralement des interfaces web, nous devrons utiliser d'autres langages dédiés que cet environnement Flask accepte. Ainsi, au niveau des pages d'affichage Internet, on utilisera du code HTML et pour les besoins d'intégration de Triton dans le flux de QuantImage-V2, nous devrons utiliser du code javascript pour interagir avec l'API dédié à Kheops.

8 Développements réalisés

Pour le lecteur soucieux d'avoir plus d'éléments concernant les implémentations de fonctionnalités effectuées pour notre travail, l'intégralité du code développé est disponible sur un répertoire GitHub dont l'accès est public (annexe 3).

8.1 Mise en place du serveur d'inférence (Triton NVidia®)

En premier lieu, il est important de noter que toutes les interactions avec l'outil se font en ligne de commande (de manière textuelle) sans interaction graphique. Ce qui peut en limiter le nombre d'utilisateurs (voir en décourager certain) qui doivent être suffisamment à l'aise avec ce type d'interaction Homme-machine.

A toutes fins utiles, nous nous permettons de rappeler que le fonctionnement d'un serveur impose la présence d'un service client pouvant se connecter à ce serveur.

Ainsi, nous avons procédé à installation sur notre réseau local (« localhost »), du service « serveur » puis du service « client » de Triton.

8.1.1 Mise en place du service (serveur)

Nous avons installé le serveur Triton dans deux environnements de travail durant notre étude mais pour des raisons de cohérence avec l'objet de ce travail, nous ne détaillerons que la mise en place sur notre environnement définitif (serveur Ubuntu de l'HES-SO). Dans ce cadre, l'installation du serveur d'inférence a suivi un processus simple mais rigoureux composé de 2 étapes.

1) la création d'un environnement virtuel Python :

Cette étape est nécessaire afin de se mettre dans des conditions où le serveur Triton pourrait être installé sur une entité Ubuntu indépendante de l'environnement actuel. Cela aura aussi pour avantage de dissocier aisément la partie serveur de la partie client. La seule relation entre ces deux entités se fera par la connexion sur le réseau local. Cet environnement virtuel sera nommé TritonServer (figure 31).

Figure 31 Activation de l'environnement virtuel Python du serveur Triton NVidia

2) L'installation du serveur d'inférence Triton

Pour ce faire, nous avons suivi les recommandations de NVidia (figure 32) et donc utiliser l'image du container Docker Triton mis à disposition par NVidia en suivant les instructions décrites sur la page de démarrage rapide pour l'utilisation (mise à disposition sur (NVidia Triton Infernce Server, 2023))

L'instruction CLI d'installation du serveur (figure 33) contient des options qui nous permettent de préciser le nombre de GPUs utilisés pour les inférences, l'adresse du réseau sur lequel la communication avec le client se fera, le dossier qui stockera les modèles à disposition pour Triton.

Après avoir contrôlé que le serveur était bien actif sur notre réseau local (figure 34), nous avons vérifié la disponibilité des modèles sur le serveur (figure 35). L'instruction dédiée à cette

vérification permet de vérifier le type d'environnement (Backend) mis en place automatiquement en même temps que les modèles prêts à être utilisés (figure 36).

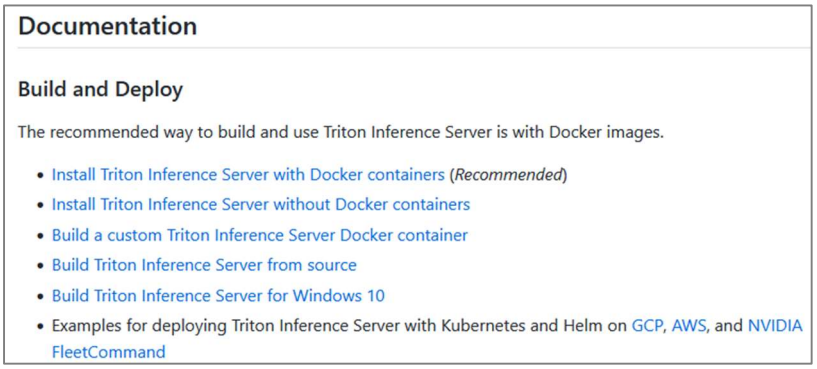


Figure 32 Recommandations NVidia pour l'installation du serveur Triton

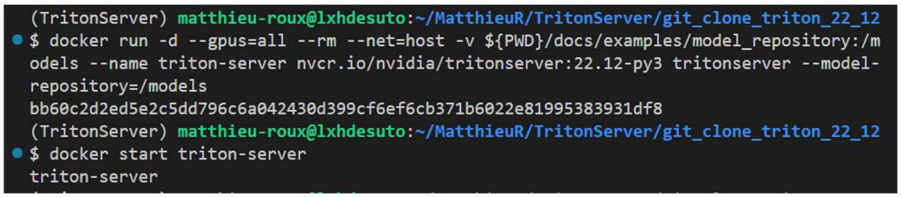


Figure 33 Construction du serveur Triton NVidia

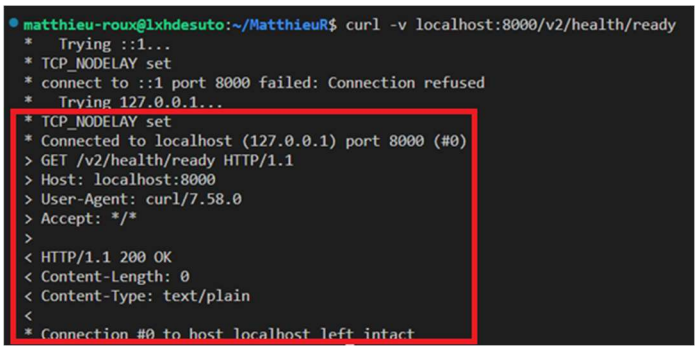


Figure 34 Validation de l'écoute du serveur Triton NVidia

Figure 35 Vérification des modèles disponibles dans le serveur Triton

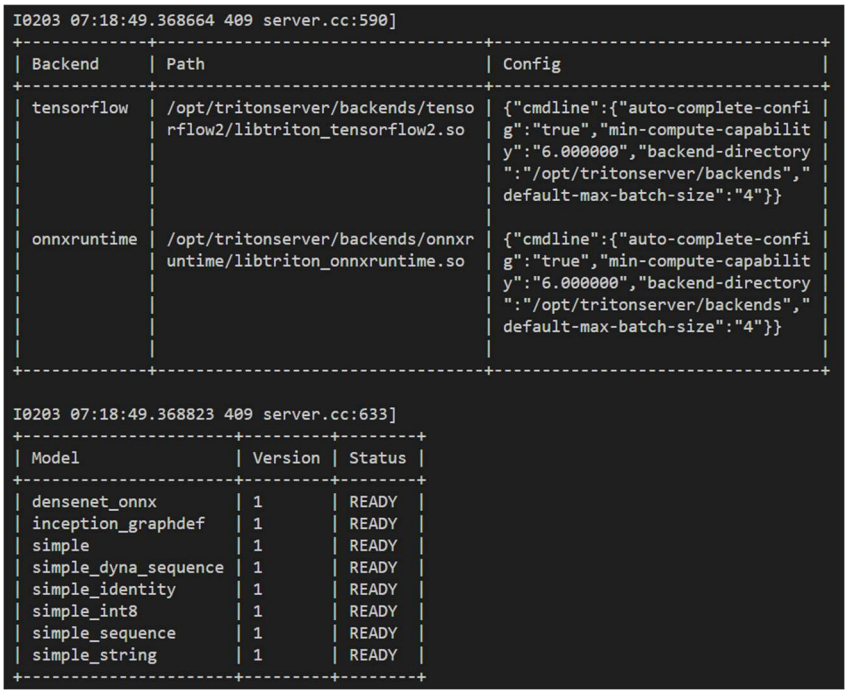


Figure 36 Résultats de consultation des modèles d'inférence sur Triton serveur

Ainsi, après toutes les vérifications faites, nous pouvions être satisfait d'avoir installé un serveur d'inférence Triton opérationnel.

Cela étant fait, nous avions des modèles d'exemples fournis par NVidia afin d'effectuer des essais de prise en main mais nous n'avions pas encore fait la démarche d'intégrer un modèle construit en-dehors de ce serveur.

D'autre part, nous n'avions pas de client Triton pour interroger ces modèles, donc nous en avons créé un.

8.1.2 Mise en place du client

Notre dessein de faire un service distant, et afin de simuler cette « distance » entre le serveur d'inférence et le client, nous avons utilisé un nouvel environnement virtuel Python (figure 37).

Dès lors, afin d'utiliser le service client Triton NVidia® et de le tester, nous avons suivi les indications fournies sur la page GitHub de ce service (GitHub/Triton inference Server/client, s.d.).

Cette dernière propose quatre possibilités de création du client mais nous en avons appliqué deux afin de les comparer et de décider du procédé le plus adapté à notre situation. Le premier moyen pour créer le client Triton est, tout comme pour le serveur Triton, d'utiliser un conteneur spécialement conçu et mis à disposition par NVidia ; la deuxième technique est d'utiliser la bibliothèque « tritonclient » pouvant être mis en place avec l'installateur de paquets Python (pip).

Pour des raisons de commodités d'installation et d'utilisation, nous avons retenu la deuxième solution pour la suite de ce travail.

Ainsi, après avoir installé ces bibliothèques permettant de disposer les éléments pour communiquer avec le serveur Triton, son test fut effectué à l'aide des exemples de code, fournis par NVidia, permettant d'interroger le serveur d'inférence.

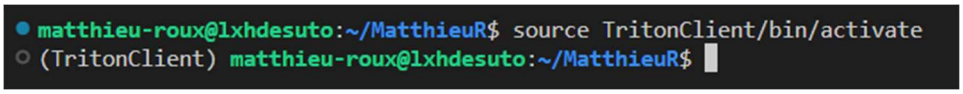


Figure 37 Activation de l'environnement virtuel Python du client Triton NVidia

Ce test validant le bon fonctionnement client-serveur du service Triton NVidia, il nous restait à intégrer un modèle compatible avec les besoins de notre travail pour la réalisation d'inférences dans le cadre de recherches médicales en radiomique.

Pour des raisons de sens logique, nous allons présenter le travail effectué pour déterminer et s'approprier ce modèle, avant de le mettre à disposition sur le serveur Triton, puis d'y accéder avec le client Triton.

8.2 Intégration du modèle d'extraction de caractéristiques profondes (DF) à Triton NVidia® (Triton)

8.2.1 Appropriation du modèle DL choisi

Le modèle est implémenté en langage Python en s'appuyant sur l'environnement TensorFlow et la bibliothèque Keras, comme nous l'avons vu précédemment.

Ce modèle étant un exemple documenté (Keras, s.d.), nous avions, dès lors, le privilège de disposer d'un manuel (Keras, s.d.) de présentation et d'explications de qualité, nous détaillant toutes les sections et actions de son implémentation.

Parmi les éléments notables, nous pouvons ressortir que le modèle fait une série de 100 cycles complets d'analyse des données (i.e. epochs). Que les données analysées proviennent d'images de scanner médicaux en 2 dimensions qui sont transformées en images 3D, que l'entraînement du modèle et sa validation se fait selon une répartition proportionnelle de 70% des images pour l'entraînement et 30% pour la validation.

Ainsi, nous avons pu nous familiariser assez rapidement avec le code et distinguer les éléments qui nous seraient utiles et ceux dont nous pouvions nous passer.

Après cette phase d'analyse et ajout d'une fonction d'enregistrement du modèle dans le code d'origine, une fois le modèle entraîné avec les données d'origine, nous avions à disposition le modèle validé enregistré dans un fichier de format HDF5 (.h5).

D'autre part, comme nous l'avons déjà signalé, les résultats restitués par la dernière couche de neurones de ce modèle ne nous convenaient pas et afin de satisfaire à notre besoin, nous l'avons rendu apte en éliminant les 3 dernières couches de neurones artificiels afin d'obtenir des deep features (256 DF) comparables aux 120 caractéristiques extraites avec Pyradiomics.

Pour ce faire, une fois la portion du modèle à conserver définie (figure 38), nous avons utilisé la bibliothèque Keras pour reconstruire le modèle afin d'isoler la partie du modèle qui nous intéressait, puis d'enregistrer ce nouveau modèle au même format que précédemment (HDF5).

```
Model: "3dcnn"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 64, 1)] 0
_________________________________________________________________
conv3d (Conv3D)              (None, 126, 126, 62, 64)  1792
_________________________________________________________________
max_pooling3d (MaxPooling3D) (None, 63, 63, 31, 64)    0
_________________________________________________________________
batch_normalization (BatchNo (None, 63, 63, 31, 64)    256
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 61, 61, 29, 64)    110656
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 30, 30, 14, 64)    0
_________________________________________________________________
batch_normalization_1 (Batch (None, 30, 30, 14, 64)    256
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 28, 28, 12, 128)   221312
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 14, 14, 6, 128)    0
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 6, 128)    512
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 12, 12, 4, 256)    884992
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 6, 6, 2, 256)      0
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 2, 256)      1024
_________________________________________________________________
global_average_pooling3d (Gl (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 512)               131584
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,352,897
Trainable params: 1,351,873
Non-trainable params: 1,024
_________________________________________________________________
```

Figure 38 Architecture du modèle DL Keras et couche de neurones utile

Figure 39 Fichier d'enregistrement HDF5 du modèle DF

Une fois ce modèle DF obtenu (figure 39), il nous restait à le mettre à disposition sur le serveur Triton.

8.2.2 Mise à disposition du modèle DF sur le serveur Triton

Comme nous avons pu le voir dans notre section décrivant l'installation du serveur Triton, les modèles sont stockés dans un dossier (« models ») lié au serveur lors de sa création.

Le simple « collage » du modèle sauvegardé, tel quel, dans « models » ne suffit pas à le mettre à disposition sur le serveur Triton. Le guide utilisateur (présent dans le répertoire GitHub de Triton Serveur et consultable à l'adresse web *https://github.com/triton-inference-server/server/blob/main/docs/user_guide/model_repository.md*) explique qu'il y a une architecture et un formatage à respecter.

Une petite analyse dans le dossier GitHub du code source de Triton serveur hébergeant les exemples de modèle (figures 40. 41 et 42) nous montrent, également, qu'en fonction de chaque type de modèle des règles spécifiques sont à appliquer.

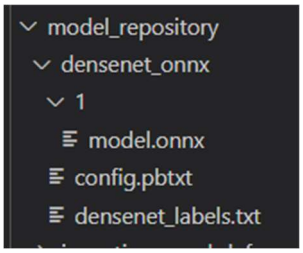


Figure 40 Exemple Triton de modèles ONNX

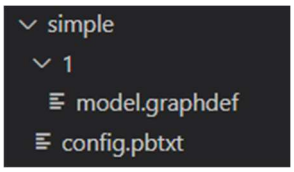


Figure 41 Exemple Triton de modèles TensoFlow "graphdef"

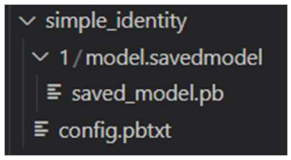
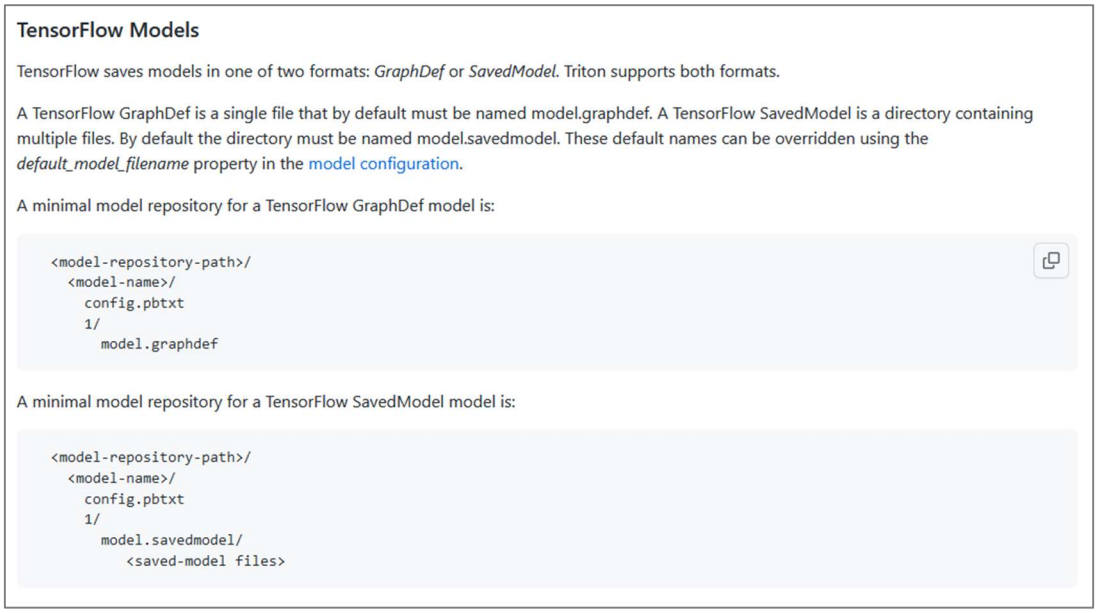


Figure 42 Exemple Triton de modèles TensorFlow "savedmodel"

En allant plus en avant dans nos connaissances pour intégrer notre modèle DL, nous nous sommes aperçus que le format de sauvegarde du modèle pour lequel nous avions opté (HDF5) n'était, actuellement, pas pris en charge par le serveur Triton et que pour les modèles développés sur l'environnement TensorFlow-Keras, seuls les formats « graphdef » et « savedmodel » (*.pb*) (figure 43) sont acceptés.



TensorFlow Models

TensorFlow saves models in one of two formats: *GraphDef* or *SavedModel*. Triton supports both formats.

A TensorFlow GraphDef is a single file that by default must be named model.graphdef. A TensorFlow SavedModel is a directory containing multiple files. By default the directory must be named model.savedmodel. These default names can be overridden using the *default_model_filename* property in the model configuration.

A minimal model repository for a TensorFlow GraphDef model is:

```
<model-repository-path>/
  <model-name>/
    config.pbtxt
    1/
      model.graphdef
```

A minimal model repository for a TensorFlow SavedModel model is:

```
<model-repository-path>/
  <model-name>/
    config.pbtxt
    1/
      model.savedmodel/
         <saved-model files>
```

Figure 43 Formatage des modèles TensorFlow pour Triton serveur

Source :https://github.com/triton-inference-server/server/blob/main/docs/user_guide/model_repository.md

D'autre part, chaque modèle doit être accompagné d'un fichier de configuration en format « protobuf text » (*.pdtxt*).

Dès lors, pour satisfaire à ces exigences, nous avons dû :

- dans un premier temps, faire en sorte de sauvegarder notre modèle DL entraîné sous un format adapté en suivant des étapes similaires à celles que nous avions déjà réalisées pour obtenir notre modèle DF. Ainsi, nous avons opté pour le format d'enregistrement *.pb* qui se présente sous la forme d'un dossier et non d'un fichier (figure 44)

- ensuite, nous avons dû créer le fichier de configuration *.pbtxt* à partir de notre modèle DF. Pour cela, il nous a fallu composer un petit programme en nous inspirant d'un code source mis à disposition sur GitHub par Lei Mao (Mao, s.d.).

  En analysant, le fichier obtenu nous avons constaté que ce dernier était constitué de la configuration du modèle mais aussi de l'ensemble des poids (figure 45), utiles aux neurones artificiels du modèle mais inutiles dans ce fichier. Nous les avons donc effacés.

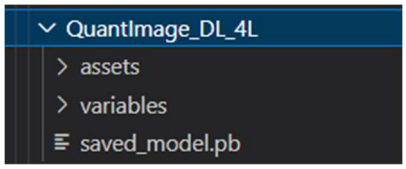


Figure 44 Dossier d'enregistrement PROTOBUF du modèle DF

```
node {
  name: "QuantImage_DL_4L/conv3d/BiasAdd/ReadVariableOp/resource"
  op: "Const"
  attr {
    key: "dtype"
    value {
      type: DT_FLOAT
    }
  }
  attr {
    key: "value"
    value {
      tensor {
        dtype: DT_FLOAT
        tensor_shape {
          dim {
            size: 64
          }
        }
        tensor_content: "\375\273\352:~\371\'\272\350\031\210\273\257N>\272\035\002V\273jQ\010;
        \275\300N\273F\357\243:c\377\311\271\351?\3059&
        \2526\272\233\004\2309\336\300::\215\360J:E\202:\270\314W\265\272\253\244\"\273[\000,
        \273sdz:\230\317\363\272{\312\024\274\202\nC\273s/
        *\272\004\375\206\273\231\212\341\272%\032\027:~\365\302:,\216\211\272\262$\321;
        \333\014T\273M{\002\273\320\267E:)\245\360\272\354\004\260\272\207\274\306;
        \362o\350\272\361=&\274\006\'\202\271\360p\371;(d\336\272\r\233\212;\213\007\337:p
        [\277:\334_?9a\177\240\271\201a
        \273\345\376\001\272\230\205\362\272\356\220!\273i\374\3569\335 \201\273\000\204\210;
        \t\255\016;\300-\207\273\0045\t;gzB:\021C_\273B\346/
        :\222R\005\273f\324\031\2737k-\273\351\320\263\270 \311\3649\231\344\010\273"
      }
    }
```

Figure 45 Extrait du fichier de configuration *.pbtxt* brut du modèle DF

- ayant réuni l'ensemble des éléments du modèle DF nécessaire à sa mise à disposition sur le serveur d'inférence, il nous restait à modifier le dossier du modèle selon la structure imposée par Triton (figure 46) afin de pouvoir l'intégrer de manière effective dans le dossier « models » du serveur Triton.

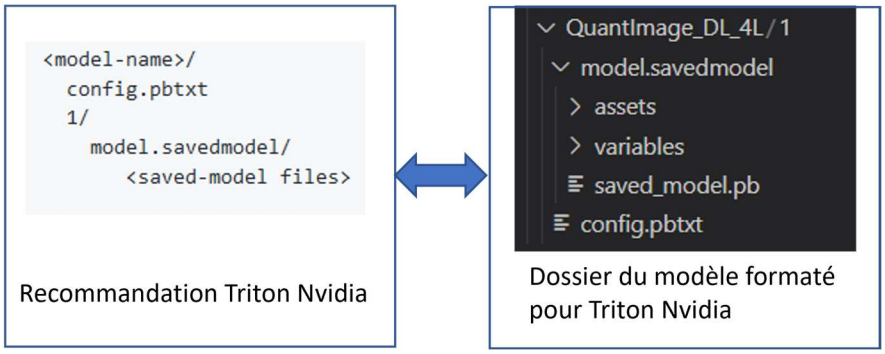


Figure 46 Adaptation du dossier d'enregistrement du modèle selon recommandation Triton NVidia°

Une fois le modèle DF répondant aux conditions imposées par Triton NVidia et disponible dans le répertoire « models » du serveur, nous avons réalisé les éléments du client Triton permettant de demander l'inférence de ce modèle.

8.2.3 Utilisation du modèle DF par le client Triton

Pour ce faire nous nous sommes inspirés d'exemples fournis par NVidia (présent dans le code source du répertoire GitHub fourni par l'équipe de NVidia (GitHub/Triton inference Server/client, s.d.), et en particulier sur le code Python présent dans *image_client.py* (figure 47).

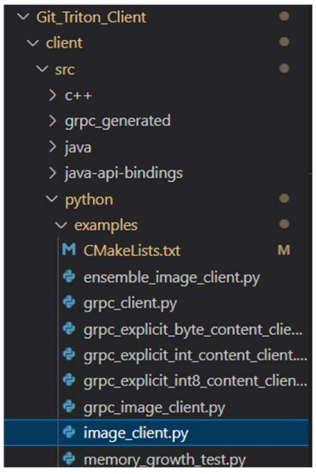


Figure 47 Exemples de code pour le client Triton fourni par NVidia

L'intérêt d'adapter le code fourni par NVidia est majeur. Il permet de conserver les bonnes pratiques de conception (par exemple pour la connexion du client au serveur et l'accès au

modèle (figure 48)) et, si jamais, les moyens sécurisés mis en place par les professionnels ayant conçu ces exemples.

```
try:
    if FLAGS.protocol.lower() == "grpc":
        # Create gRPC client for communicating with the server
        triton_client = grpcclient.InferenceServerClient(
            url=FLAGS.url, verbose=FLAGS.verbose)
    else:
        # Specify large enough concurrency to handle the
        # the number of requests.
        concurrency = 20 if FLAGS.async_set else 1
        triton_client = httpclient.InferenceServerClient(
            url=FLAGS.url, verbose=FLAGS.verbose, concurrency=concurrency)
except Exception as e:
    print("client creation failed: " + str(e))
    sys.exit(1)

# Make sure the model matches our requirements, and get some
# properties of the model that we need for preprocessing
try:
    model_metadata = triton_client.get_model_metadata(
        model_name=FLAGS.model_name, model_version=FLAGS.model_version)
except InferenceServerException as e:
    print("failed to retrieve the metadata: " + str(e))
    sys.exit(1)
```

Figure 48 Code de création de l'objet client de Triton

Cette phase de notre travail compose de 2 étapes :

1) le développement d'une application d'interaction avec le modèle de deep learning, adaptée à nos besoins :

(Pour des raisons de simplification de compréhension, nous appellerons le code original de référence « code 1 » et le code modifié « code 2 »).

Ainsi, le code 1 nous fournissant déjà tous les éléments pour nous connecter au serveur Triton, dans un premier temps, l'adaptation a été faite afin de permettre l'utilisation des images médicales 3D dans le but d'alimenter notre modèle DF, ensuite, afin de permettre le traitement de la réponse et l'affichage les résultats retournés.

Pour ce faire, nous avons intégré de nouvelles fonctionnalités et modifié, en conséquence, le module du code 1 nommé « preprocess » (figure 49) afin de permettre la prise en charge d'images 3D en format NIFTI nécessaires pour notre modèle DF.

Ensuite, nous avons adapté l'affichage des résultats dans le module « postprocess » (figure 50).

N.B : à ce stade de notre travail, l'affichage ne se fait que dans le terminal de ligne de commande, donc uniquement en format texte avec la commande python « print (…) ».

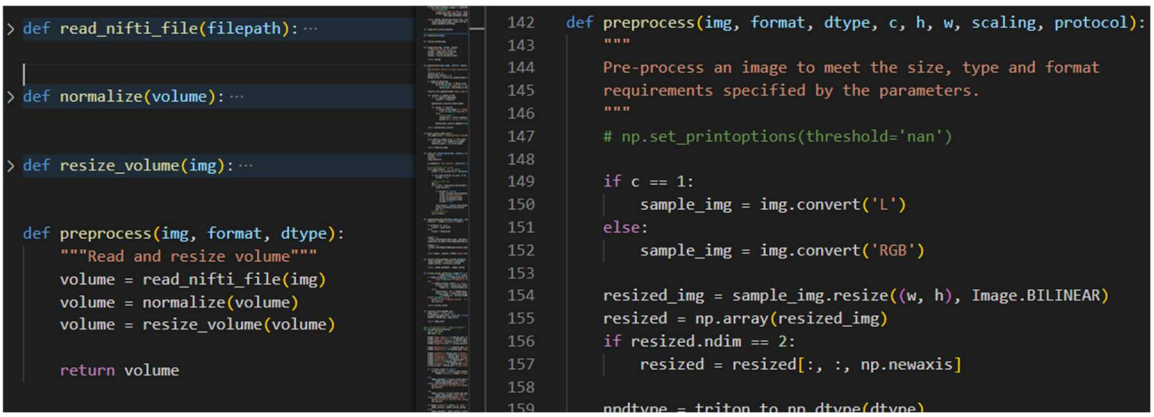


Figure 49 Adaptation du module "preprocess"

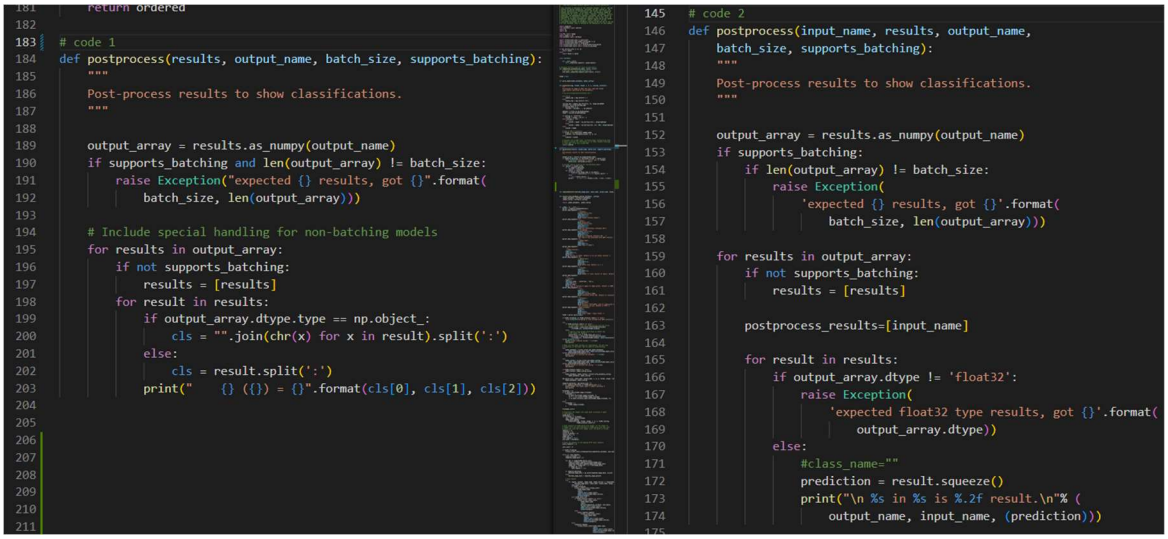


Figure 50 Comparaison code1 / code 2 : méthode postprocess

Le programme obtenu était opérationnel mais ne prenait pas en charge la transformation des image 2D en image 3D. Ainsi nous avons développé et intégré un module (figure 51) dédié à cet effet s'appuyant sur les fonctionnalités d'OKAPI.

```
def recovery_nifti(DICOM_zip):
    temp_nifti= "./temp_nifti/"
    os.makedirs(temp_nifti, exist_ok = True)
    toNifti (DICOM_zip, temp_nifti)

    return temp_nifti
```

Figure 51 Module de transformation des images 2D en images 3D

Ce programme réunissant les principales fonctionnalités nous permettant de tester le service d'inférence fourni par le serveur Triton et son test étant concluant, il nous restait à concevoir une interface graphique plus conviviale que l'interaction par ligne de commande.

8.3 Développement de l'interface web

Comme définis précédemment, nous avons choisi de développer cette couche web avec l'environnement Flask qui est dédié à Python.

Dès lors, l'implémentation pour le routage et pour la majorité des interactions avec le serveur d'inférence au sein des pages web se fait en Python.

Ce développement peut être détaillé en 2 grandes étapes.

La première fut la mise en place d'une web application (WebApp) simple avec un accès direct à nos données tests d'examens médicaux. Et la deuxième fut de développer un accès à l'outil Kheops permettant un accès aux examens médicaux utiles pour la recherche médicale clinique.

8.3.1 Développement WebApp 1 (simple)

Durant ce développement, nous avons défini un flux de pages web simples composé de 2 éléments.

La première page (figure 52) permet de faire la sélection du modèle disponible sur Triton serveur et de choisir les examens à analyse et valider ces choix ; et la deuxième (figure 53) permet de voir les résultats de l'inférence.

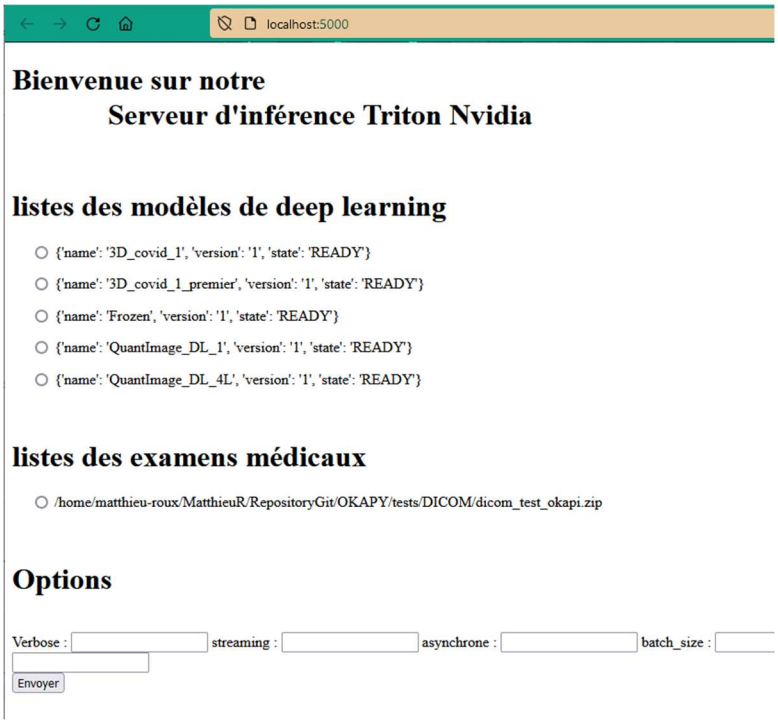


Figure 52 Première page WebApp simple

En arrière-plan, la connexion au serveur Triton se fait via un client, déclaré dans la WebApp, qui va interroger le serveur. Ainsi, les actions « cachées » à l'utilisateur, sont dans l'ordre chronologique :

- une requête du client Triton vers le serveur pour savoir s'il est disponible et quels sont les modèles pouvant être utilisés ;
- la réponse du serveur qui envoie la liste des modèles disponibles ;
- l'interrogation par la WebApp vers le répertoire de stockage des examens d'imagerie médicale pour avoir la liste des examens disponibles ;
- une requête d'inférence vers le serveur Triton, par le client Triton, avec le modèle et les images choisis ;
- au niveau du serveur Triton, la transformation des images 2D en images 3D, puis leur analyse par le modèle DL choisi ;
- l'inférence et le renvoi des résultats obtenus par le serveur Triton.

Une fois cette interface fonctionnelle et validée (figure 53), du fait de notre volonté de pouvoir intégrer le serveur d'inférence Triton dans un flux de recherche médicale déjà existant, nous avons poursuivi cette quête en faisant évoluer cette WebApp afin de l'interfacer avec l'outil Kheops (permettant l'accès aux examens médicaux utiles pour leurs recherches cliniques).

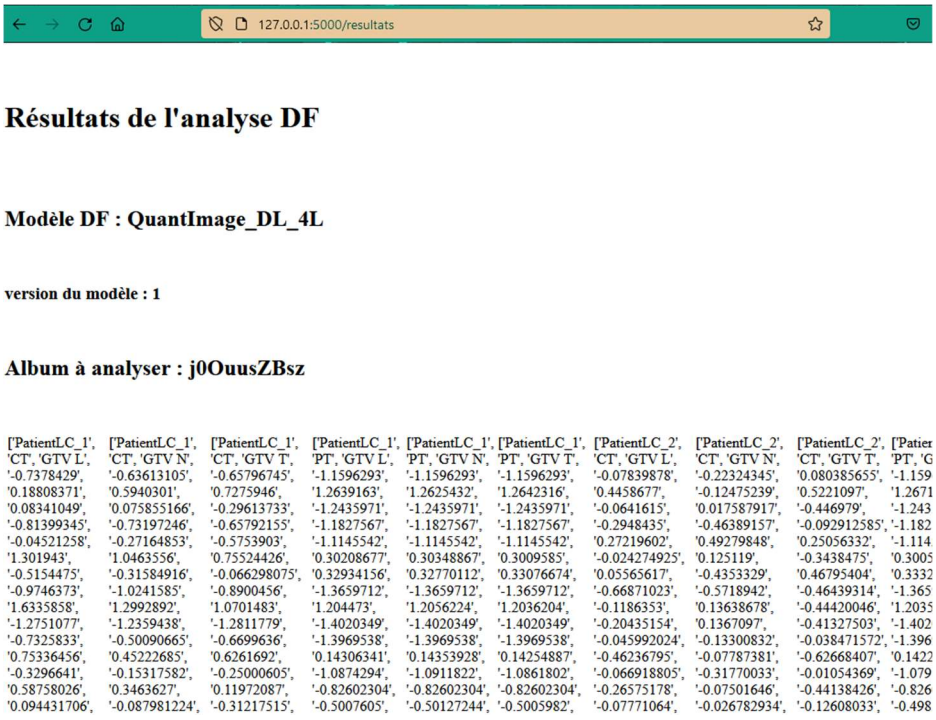


Figure 53 Affichage des résultats DF dans interface web

8.3.2 Développement WebApp 2 (interfacée à Kheops)

Cette deuxième WebApp est une évolution de la précédente où il a fallu intégrer une API existante (Keycloak) gérant l'accès sécurisé à Kheops dans la cascade d'événements créée dans la première interface web.

L'intégration de cette interface ne fut pas sans mal, et après de multiples tentatives, nous avons dû adapter notre nombre de pages Internet et notre séquence d'affichage.

Ainsi, nous avons développé (figure 55) trois pages web visibles par l'utilisateur permettant pour la première page de se connecter à Kheops (figure 56) si aucune connexion n'est active (et passant immédiatement à l'étape suivante si l'utilisateur est déjà connecté), puis suivre les

étapes (figure 58) de sélection d'un album contenant les examens convoités, de choix des examens d'imagerie médicale souhaités et de validation de son choix pour accéder à la page suivante.

La deuxième page web (figure 57), permet de choisir le modèle DF (ou DL) et de valider ce choix pour faire l'inférence.

La troisième page de visualiser les résultats(figure 58), les enregistrer sous un format *.csv* sur l'ordinateur de l'utilisateur (tous dans un fichier *.zip*) (figures 59 et 60) et de recommencer une nouvelle inférence.

En arrière-plan, l'application réalise dans l'ordre chronologique :

- l'interrogation du service d'autorisation d'accès à Kheops ;
- la demande et la réception de la liste des albums disponible sur Kheops ;
- la demande et la réception de la liste des examens d'imagerie médicale CT, PT (et IRM) disponible dans l'album choisi précédemment ;
- le téléchargement sur le serveur hébergeant la WebApp des examens choisis ;
- la transformation des images 2D en images 3D ;
- la création d'un client Triton ;
- une requête de ce client vers le serveur Triton pour savoir s'il est disponible et quels sont les modèles pouvant être utilisés ;
- la réponse du serveur qui envoie la liste des modèles disponibles ;
- une requête d'inférence, par le client Triton vers le serveur Triton, avec le modèle et les images choisis ;
- l'inférence et le renvoi des résultats obtenus par le serveur Triton.

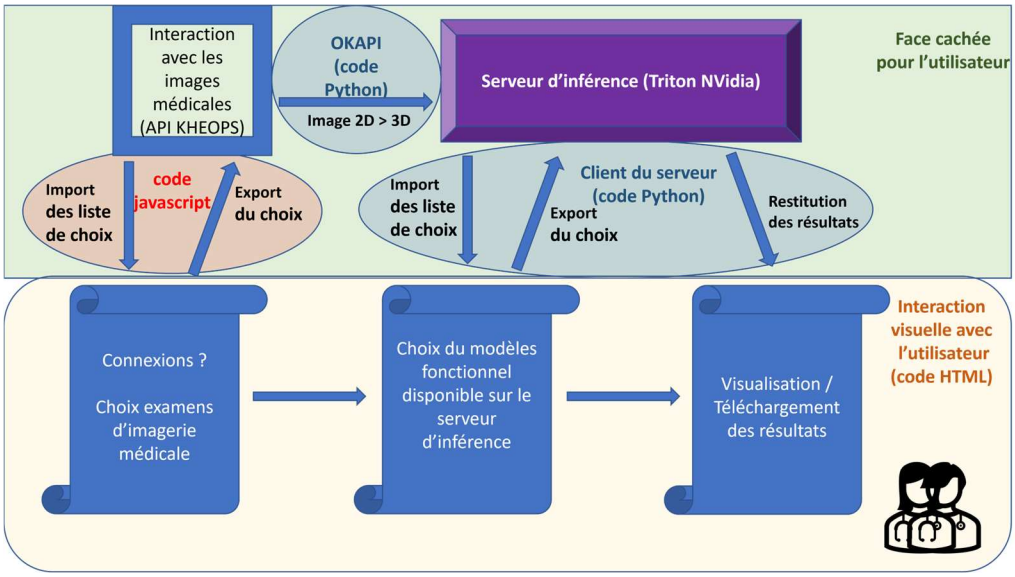


Figure 54 Architecture WebApp - API Kheops

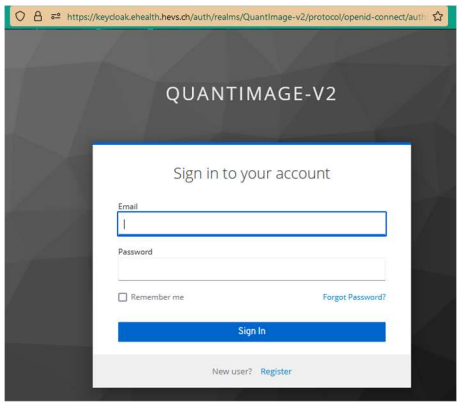


Figure 55 Fenêtre de connexion à Kheops avec son identifiant QuantIamge-V2

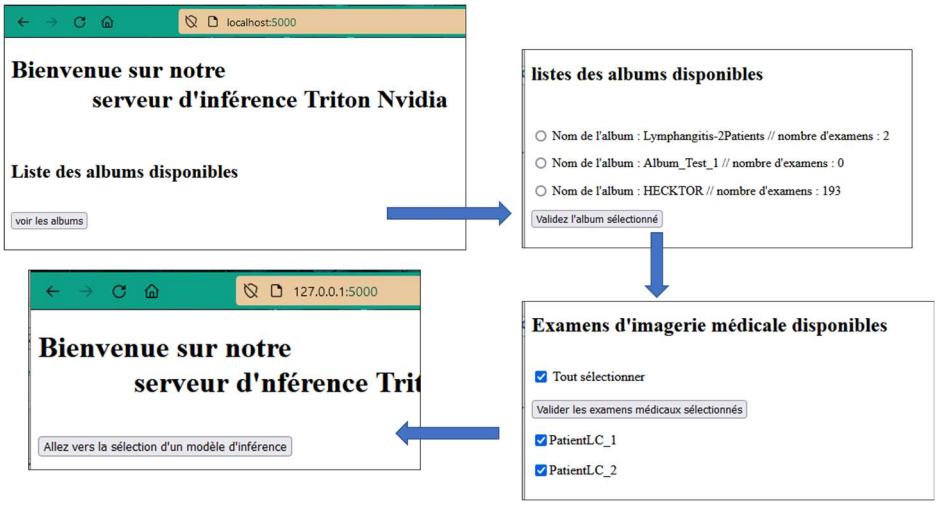


Figure 56 WebApp Kheops : Séquence d'affichage - page 1

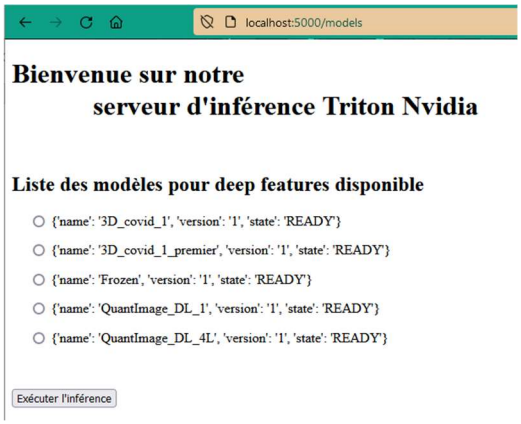


Figure 57 WebApp - page 2

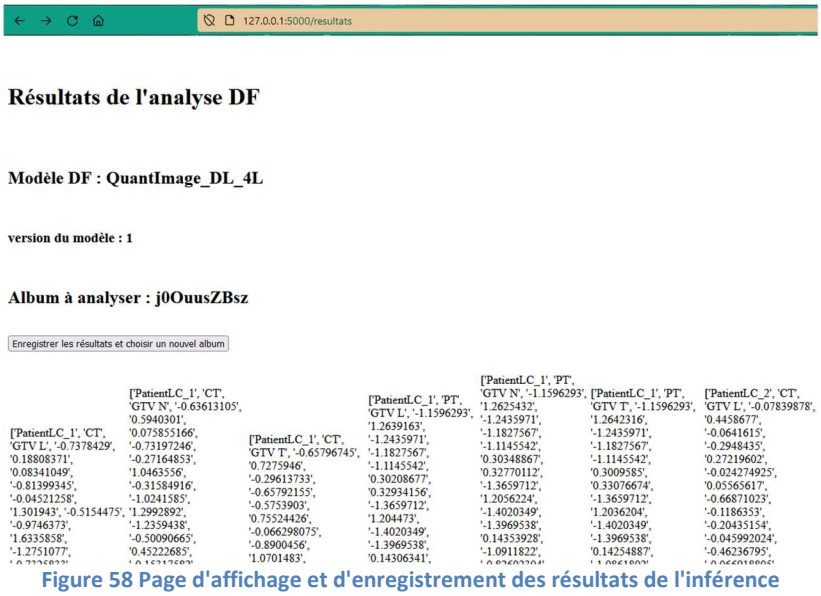


Figure 58 Page d'affichage et d'enregistrement des résultats de l'inférence

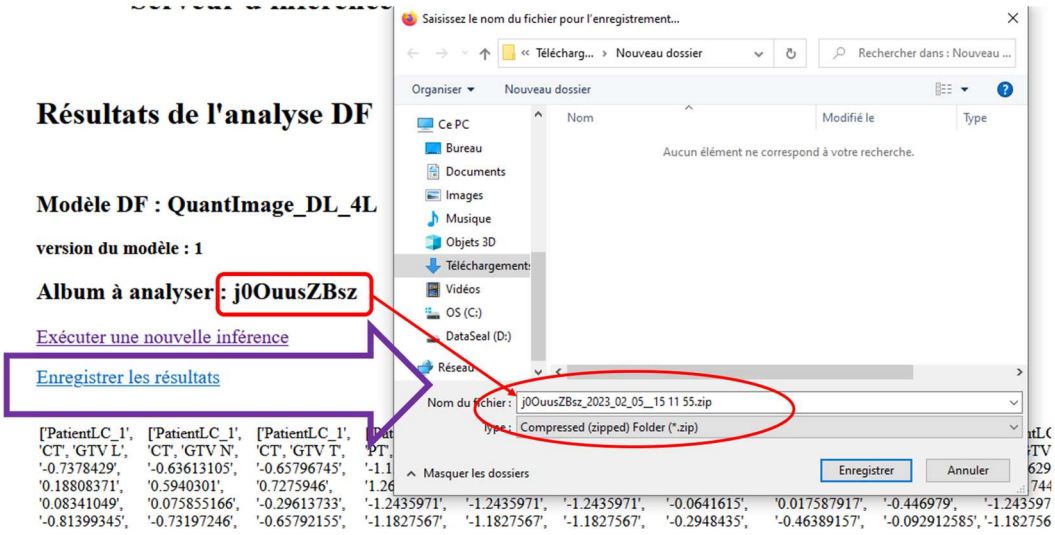


Figure 59 Enregistrement des résultats DF

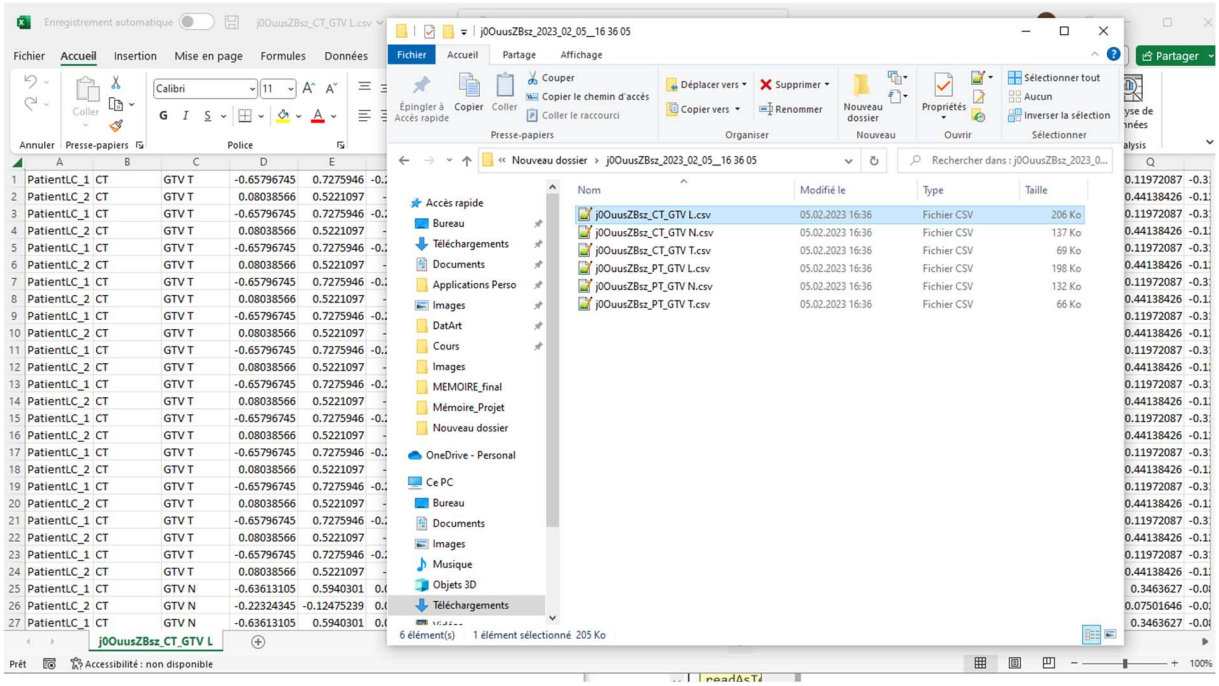


Figure 60 Fichiers csv d'enregistrement dans le zip

8.4 Test de performances et test de sécurité

Les tests de performance pour l'utilisation du modèle peuvent être faits avec des outils supplémentaires à installer sur le serveur d'inférence. Malheureusement, nous n'avons pas pu en mettre en place durant le lapse de temps qui nous était imparti.

Concernant les tests de sécurité, il existe des tests au sein même de Triton serveur qui évalue la sécurité du client. Ainsi, en se basant sur les exemples de programme client fournis et en respectant l'usage des librairies de Triton, la sécurité au niveau connexion entre le client Triton et le serveur Triton est gérée par les outils NVidia.

Cependant, même si la connexion avec l'API de KHEOPS est sécurisée et oblige l'usage d'un token pour accéder aux examens d'imagerie médicale, la sécurité au niveau de l'accès avec Flask n'a été ni évaluée ni testée. Là aussi, par manque de temps.

Ce projet en l'état ne pourra pas être mis en service sans faire ses tests d'acceptance obligatoire.

9 Discussion

Au cours de notre travail, nous avons évalué l'intérêt de l'intelligence artificielle en médecine, et en particulier la technologie de l'apprentissage profond en radiomique. Nous nous sommes aussi penchés sur les différents moyens de mise à disposition et d'intégration dans les outils de routine de cette technologie pour les professionnels non-spécialistes en technologie de l'information.

Tout ceci ayant été entrepris avec attention, la présentation de nos résultats en soulignera les intérêts et les limites en découlant.

- Concernant l'intérêt de la technologie de l'apprentissage profond pour recherche médicale clinique en radiomique :

Nous avons pu voir que le développement de bout en bout de modèle dédié à la recherche en radiomique n'était pas judicieux en vue d'obtention de résultats fiables puisque les études en radiomiques se basent sur des cohortes de patients peu fournies. Malgré tout, le DL serait adapté pour l'extraction à haut débit de caractéristiques radiomiques par analyse d'images médicales. Elle permettrait de s'affranchir de l'extraction par intervention humaine (handcrafting) de ces caractéristiques.

Les avancées des techniques d'utilisation et de manipulation des modèles DL confirmés, nous ouvrent une porte nous permettant d'envisager de pallier l'impossibilité de création de nouveaux modèles DL. En adaptant des modèles de traitement d'images médicales éprouvés, il est, aujourd'hui, possible d'extraire des caractéristiques profondes exploitables par les outils de ML (non-deep) déjà en place.

Dès lors, une alternative découlant du DL pouvant apporter un bénéfice aux recherches médicales en radiomique, nous avons pu déterminer un modèle DF utile.

- Concernant la technologie de DL utilisée :

Nous avons adapté un modèle DL existant en l'amputant des couches de neurones terminales inutiles pour la recherche radiomique afin de récupérer les caractéristiques fournies par la dernière couche de neurones conservée.

Cette démarche fut ponctuée par la rechercher d'un modèle DL valide adapté pour notre travail et par l'adaptation de ce modèle.

La première étape, bien que réalisée par un expert dans le domaine de l'IA, fut assez compliquée car il faut prendre en compte divers paramètres (tels que le type d'images médicales à analyser) qui limites drastiquement le choix des modèles DL adaptés.

La deuxième étape, nécessite aussi l'intervention d'un expert pour analyser le modèle DL choisi et définir qu'elle sera la couche de neurones pertinente au vu des données qu'elle renvoie.

- Concernant les moyens techniques de mise à disposition de cette technologie :

Les moyens techniques permettant de fournir l'inférence à, potentiellement, plusieurs groupes de chercheurs en médecine, implique une nécessité d'accès à distance. Nous avons donc axé nos recherches sur la possibilité d'y accéder via le web.

Nous pouvons relever qu'il est toujours possible de construire un service d'inférence de bout en bout mais qu'il existe de nombreux outils consacrés à cette tâche. Il y a entre autres les fournisseurs de service d'hébergement Cloud qui offre des plateformes dévouées au déploiement de modèles DL et qui ont comme inconvénient d'être payants. Et il existe des outils open source permettant le déploiement de services d'inférence. Parmi tous ces outils en libre accès, le choix peut être difficile et il a été nécessaire de définir les caractéristiques essentielles pour notre usage.

Selon nos conditions, nous avons retenu un outil assez généraliste, le serveur d'inférence Triton NVidia, offrant très bonne flexibilité nous ayant permis de nous adapter aisément au modèle que nous avons défini. La création aisée de la relation client-serveur est aussi un point fort de cet outil.

D'autre part, Triton bénéficie d'une équipe de développement performante qui réalise des mises à jour régulières permettant une compatibilité optimale avec l'infrastructure dans laquelle il est développé. Il bénéficie aussi d'une très grande communauté permettant de trouver des réponses en cas de questions ou de problèmes lors de son installation ou de sa maintenance. L'inconvénient, de ce développement prospère et d'une telle communauté active, réside dans le fait que beaucoup d'informations désuètes circulent et qu'il est nécessaire d'être très attentif en recherchant des informations.

Par ailleurs, la prise en main facilitée de Triton n'est pas dénuée d'inconvénient et demande un certain temps d'apprentissage. Par exemple, la mise à disposition d'un modèle personnalisé pour une inférence demande un formatage spécifique du modèle. Ce formatage n'est pas disponible spontanément avec les plateformes de développement (e.g., TensorFlow) des modèles DL et demande une intervention supplémentaire avec l'implémentation de fonctions spécifiques afin de fournir au serveur Triton le modèle dans le format approprié. D'autre part, l'interface unique en ligne de commande restreint son utilisation par des personnes coutumières des commandes CLI (ou volontaires pour se former).

On peut noter aussi qu'il existe une variété d'outils non-négligeable permettant d'évaluer les performances des inférences, d'améliorer les inférences, … que nous n'avons pas pu étudier au cours de notre travail.

Ainsi, nous pouvons conclure que le serveur d'inférence Triton est un outil performant permettant de s'adapter à beaucoup de situations et offrant un nombre substantiel d'outils intégrables pour son utilisation mais qu'il nécessite une phase d'apprentissage et des connaissances informatiques abouties afin de pouvoir être mis en production permettant l'accès aux modèles DF et/ou DL pour les inférences.

- Concernant l'accessibilité des outils d'inférence par les utilisateurs :

Comme nous l'avons souligné précédemment, le serveur d'inférence Triton se paramètre uniquement en ligne de commande et cela peut être un frein, pour des chercheurs en médecine clinique radiomique, pour installer cet outil mais aussi pour accéder à l'inférence.

Dès lors, l'implémentation d'une interface graphique plus conviviale semble nécessaire pour un usage élargi au non-professionnel des technologies de l'information.

Nous avons choisi d'implémenter cette interface de demande d'inférence et de visualisation des résultats en utilisant un outil dénommé Flask. Cet outil permettant de développer des pages web sur une base de langage Python, nous pouvons conclure que, dans ce cadre du langage Python, il est facile pour une personne ayant des compétences sommaires en programmation web de réaliser une interface utilisateur permettant d'utiliser le serveur d'inférence Triton.

- Concernant les moyens d'intégration des outils d'inférence dans un dispositif déjà existant

Dans le cadre de notre travail, nous avons pris l'exemple d'une plateforme existante dénommée QuantImage-V2 pour projeter notre réflexion d'une telle intégration sur une base objective et concrète.

Durant le travail d'analyse pour une telle incorporation, il a fallu étudier la fonctionnalité apportée par l'outil d'inférence, afin d'en déterminer son emplacement géographique et son insertion dans le flux d'utilisation de QI2 avant de définir le moyen d'intégration.

Ainsi, comme le montre la figure 61, il nous est apparu plus raisonnable d'externaliser le service d'inférence par rapport à la plateforme QI2. Ceci, parce que le serveur d'inférence n'a pas pour vocation d'être dédié à QI2, bien au contraire, il doit pouvoir profiter au plus grand nombre (i.e. équipes et sujets de recherche multiples).

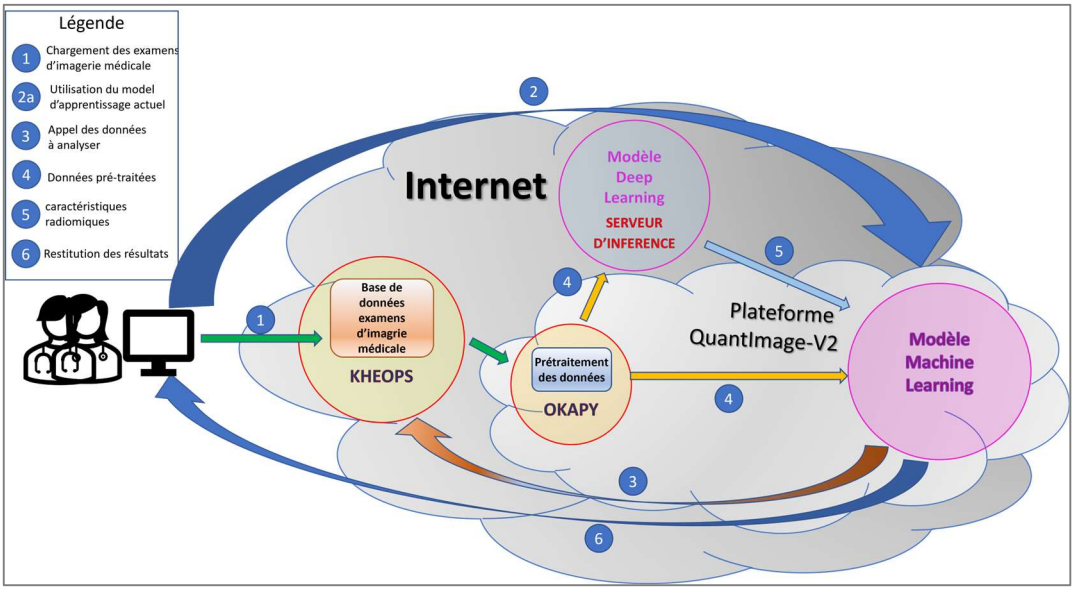


Figure 61 Intégration d'un outil d'inférence DF à QuantImage-V2

Après avoir, déterminer la place et le rôle du serveur d'inférence (Triton), nous avons décidé de réaliser la simulation de son intégration par l'intermédiaire une interface web.

Par cette interface, l'inférence devait se faire en toute fluidité pour l'utilisateur. Malheureusement, un problème dans la gestion des authentifications pour Kheops par une API (Keycloak) a perturbé la navigation web entre les différentes pages de notre WebApp et nous a imposé d'ajouter un élément cliquable nous obligeant à accéder explicitement au choix du modèle DF.

Ce souci de gestion de sécurité, qui nous avait déjà obligé à changer notre séquence d'affichage des pages, nous interpelle sur les compatibilités entre Triton et les outils existants en ce qui concerne la gestion des autorisations et des authentifications.

- Concernant les tests d'acceptance :

La mise en place de notre environnement de travail ayant été chaotique, la prise en main de Triton (en particulier, la mise à disposition du modèle DF) ayant demandé un temps non négligeable, nous n'avons pas trouvé le temps de mettre en place des tests de sécurités et de performances.

Ceci est particulièrement regrettable car on peut imaginer que si des tests de sécurité avaient été effectués, nous n'aurions peut-être pas eu le problème rencontré au niveau de notre WebApp avec connexion sur Kheops.

Quoiqu'il en soit, ces tests seront obligatoires en prévision d'une mise en production du serveur Triton.

10 Conclusion

Notre travail nous a permis d'apporter des éléments de réponse au problème de la place et de l'usage (via une interface web) de l'apprentissage profond en recherche médicale en radiomique.

Pour y répondre, nous avons pesé le pour et le contre par rapport aux autres technologies d'intelligence artificielle, évalué les avantages et inconvénients de la technologie DL, et replacé le tout dans le cadre spécifique de la radiomique.

L'issue de cette démarche, aboutissant vers l'extraction de caractéristiques profondes (DF) par modèle DL, nous a permis de poursuivre notre démarche en sélectionnant le serveur d'inférence Triton NVidia®, un outil d'inférence parmi les outils actuellement disponible, selon des critères précis.

Nous avons pu expérimenter sa mise en service et tester l'inférence d'un modèle DL que nous avons choisi, adapté et formaté (pour répondre à notre besoin et aux contraintes de Triton).

L'interface native du serveur Triton, ne nous permettant pas de satisfaire à notre exigence d'interface web, nous avons dû la développer.

Ce développement s'est fait en deux parties, l'une, représentant une interface simple en lien direct et unique avec le serveur Triton, a été réalisée de bout en bout. Et l'autre, souhaitant simulée l'intégration de Triton serveur dans le flux de la plateforme de recherche fonctionnelle QuantImage-V2, a pu être développée mais elle a dû être adaptée à la suite d'un

problème survenu avec l'outil de gestion de l'authentification de connexion à Kheops (permettant d'accéder aux examens d'imagerie médicale).

Ainsi, la technologie d'apprentissage profond peut avoir sa place dans les travaux de recherches médicales en radiomique et le serveur d'inférence Triton NVidia permet l'inférence des modèles d'extraction de caractéristiques profondes pour les chercheurs en médecine moyennant le développement d'une interface visuelle pour son utilisation.

La mise en place d'un tel service demande un regroupement de compétences entre experts en médecine et experts en technologies de l'information afin que les premiers transmettent leurs besoins aux seconds et que ces derniers mettent en place tous les moyens pour que le service soit opérationnel et efficace.

Partant de cette conclusion, on peut se demander si les DF ont une vraie valeur ajoutée dans les études radiomiques et si un déploiement concret du serveur d'inférence Triton pourrait permettre une centralisation du service d'inférence pour toutes les équipes de recherche travaillant avec la HES-SO.

11 Ouvertures possibles

Notre travail a permis de trouver une application de la technologie d'apprentissage profond en recherche médicale clinique en radiomique avec l'utilisation des caractéristiques profondes (DF) dans les modèles DL validés. Cependant, il reste à démontrer l'efficacité d'une telle application. Ainsi, un travail de recherche médicale pourrait être effectué en ce sens.

Par ailleurs, nous avons dû nous rendre à l'évidence que le temps imparti pour la réalisation de ce travail ne nous a pas permis d'effectuer tous les tests nécessaires pour une mise en production de Triton serveur. Dès lors, un travail complémentaire permettant d'évaluer la sécurité et les performances mais aussi les autres outils disponibles pour l'utilisation de Triton serait judicieux. En fonction des conclusions, ce service d'inférence pourrait servir à toutes les équipes de recherche en lien avec la HES-SO.

D'autre part, nous avons mis en service, localement, le serveur d'inférence Triton NVidia. Nous avons pu vérifier son efficacité pour l'inférence d'un seul type de modèle DL demandé par un seul utilisateur familiarisé avec les technologies de l'information (IT). Dès lors, en combinaison avec le travail complémentaire cité précédemment, il pourrait être judicieux de réaliser un laboratoire de test à plus grande échelle afin de vérifier la stabilité et l'utilisabilité en cas de charge de travail intense avec des non-professionnels IT.

Notre travail a porté à notre attention deux autres points méritant une étude plus approfondie. Le premier concerne le problème de connexion sécurisée et de navigation dans notre WebApp et le deuxième concerne l'absence d'interface graphique. Pour le premier point, les deux études précédentes permettraient d'y répondre. Concernant, le deuxième point, il faudrait d'abord faire une analyse des besoins auprès des chercheurs en médecine mais on pourrait envisager le développement d'une plateforme sécurisée avec une interface visuelle aboutie permettant aux chercheurs non-IT de mettre à disposition des modèles DL (transféré par une autre équipe de recherche, par exemple) sur le serveur Triton avec la gestion des connexions à différentes bases de données et la possibilité de demande d'inférence. Cette plateforme permettrait une certaine autonomisation des chercheurs non-IT.

Bibliographie

Daniel Abler, R. S. (2023). QuantImage v2: A Comprehensive and Integrated Physician-Centered Cloud Platform for Radiomics and Machine Learning Research. *European Radiology Experimental*.

F. Pesapane, M. C. (2018). Artificial intelligence in medical imaging: threat or opportunity? Radiologists again at the forefront of innovation in medicine. *Eur Radiol Exp, 2*(35). doi:10.1186/s41747-018-0061-6

Galanopoulo, L. (2018, 02 02). *Des logiciels experts en diagnostic médical*. Récupéré sur CNRs Le journal: https://lejournal.cnrs.fr/articles/des-logiciels-experts-en-diagnostic-medical

Geai. (2022, 02 01). *Top 9 des langages de programmation utilisés dans l'IA*. Récupéré sur hashdork: https://hashdork.com/fr/programming-languages-used-in-ai/

GitHub/Triton inference Server/client. (s.d.). Récupéré sur NVidia: https://github.com/triton-inference-server/client

Janiesch, C. Z. (2021). Machine learning and deep learning. *Electron Markets*, 685–695. Récupéré sur https://doi.org/10.1007/s12525-021-00475-2

Keras. (s.d.). *3D_image_classification*. Récupéré sur keras.io: https://keras.io/examples/vision/3D_image_classification/

Krizhevsky A., S. I. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Récupéré sur https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Lambin P, R.-V. E. (2012, March). Radiomics: extracting more information from medical images using advanced feature analysis. *Eur J Cancer, 4*(48), pp. 441-6. doi:10.1016/j.ejca.2011.11.036

LeCun Y, B. Y. (2015, May 28). Deep learning. *Nature*, pp. 436-44. doi:10.1038/nature14539

Mao, L. (s.d.). Récupéré sur GitHub: https://github.com/leimao/Frozen-Graph-TensorFlow/blob/master/TensorFlow_v2/example_2.py

Montagnon, E. C.-C. (2020). Deep learning workflow in radiology: a primer. *Insights Imaging, 11*(22). doi:10.1186/s13244-019-0832-5

NVidia Triton Infernce Server. (2023, 01 30). Récupéré sur NVidia: https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/getting_started/quickstart.html

PubMed. (s.d.). Récupéré sur PubMed: https://pubmed.ncbi.nlm.nih.gov/)

quantimage-v2-info. (s.d.). Récupéré sur https://medgift.github.io: https://medgift.github.io/quantimage-v2-info/

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 6*(65), pp. 386-408.

Samuel, A. L. (1959, July). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development, 3*(3), pp. 210-229. doi:10.1147/rd.33.0210

Sarker, I. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN COMPUT. SCI.*(2), p. 420. Récupéré sur https://doi.org/10.1007/s42979-021-00815-1

stable. (s.d.). Récupéré sur https://scikit-learn.org: https://scikit-learn.org/stable/

van Timmeren, J. C.-L. (2020). Radiomics in medical imaging—“how-to” guide and critical reflection. *Insights Imaging, 11*, p. 91. doi:https://doi.org/10.1186/s13244-020-00887-2

Vishwa, P. &. (2016, 03 31). A new application from established techniques. *Expert Rev Precis Med Drug Dev*, pp. 207-226. doi:10.1080/23808993.2016.1164013

Y. Wei, W. X. (2014, Jun 22). CNN: Single-label to Multi-label. Récupéré sur https://arxiv.org/abs/1406.5726

Annexes

Annexe 1 : Product Backlog

US Nr.	Thème	User Stories			Priority	Status	Size	Acceptance Criteria	Sprint	US accepted (done done)	MoSCoW
		En tant que ...	Je veux ...	car/pour ...							
1	Préparation /réflexion	Développeur	définir use cases	Avoir un cadre de développement	1250	3	1	Faisabilité du projet	1	ok	must
3	Préparation	Développeur	Définir les technologies utilisées	Architecture front-End, le type d'examen médical à analyser et l'architecture de Deep Learning, en accord avec la plateform existante (Quantimage)	1200	3	2	Outils existants réponds à nos critères	1	ok	MUST
4	Préparation	Développeur	Préparer l'environnement de travail	Mettre en place les outils de développement et de tests	1150	3	2	Environnement fonctionnel	2	ok	MUST
6	Préparation	Spécialiste en Intelligence artificielle	Définir un modèle de deep learning	Permettre l'analyse d'images médicale CT, PT (et IRM)	1050	3	2	Accepter les images 3D reconstruites depuis des images 2D CT, PT (et IRM) par l'outil OKAPI	2	ok	MUST
7	Préparation	Développeur	Prendre en main le modèle de deep learning	Permettre son intégration et son utilisation dans l'envireonnnemt de développement	1000	3	3	Comprendre la technologie et le langage informatique utilisé	3	ok	MUST
8	Deploiement	Développeur	Installation de l'outil d'inférence choisi	Avoir un dispositif d'inférence fonctionnel	950	3	8	Avoir une connexion client/serveur fonctionnelle vers l'outil d'inférence	3	ok	MUST
9	Préparation	Développeur	Obtenir des examens d'imagerie médicale	Constituer un dataset pour faire les tests	900	3	1	Création seeders	3	ok	MUST
10	Deploiement	Développeur	Mettre le modèle de DL à disposition dans l'outil d'inférence	Permettre l'inférence du modèle de DL	850	3	5	Charger un modèle de DL sur l'outil d'inférence et le rendre accessible	4	ok	MUST
11	Inference	Développeur	Exécuter l'analyse des données du dataset	Réaliser l'inférence de données du dataset	800	3	5	Développer l'outil permettant d'exécuter l'inférence	4	ok	MUST
12	inference	Développeur	Afficher les résultats de l'analyse par deep learning	Vérifier que l'inférence à bien été exécutée et constater les résultats	750	3	2	Développer l'outil permettant la restitution des résultas d'une inférence	5	ok	MUST
13	WebApp (simple)	Utilisateur	Définir le flux de navigation	Connaître les actions et affichages à prévoir	700	3	1	Définition de l'outil de développement de la web application	5	ok	MUST

14	WebApp (simple)	Utilisateur	Accéder à la collection d'examen d'imagerie (dataset)	Avoir la possibilité d'accéder aux images nécessaires pour son travail de recherche	650	3	8	Bouton d'accès + page de présentation des collections d'examen du dataset (liste des examens disponibles)	5	ok	MUST
15	WebApp (simple)	Utilisateur	Sélectionner un examen d'imagerie du dataset	Mettre à disposition un examen d'imagerie médicale pour une analyse par Deep Learning	600	3	5	Visualisation d'une liste de tous les examens disponibles avec des éléments clicables permettant leur sélection	6	ok	MUST
16	WebApp (simple)	Utilisateur	Exécuter une analyse de l'examen d'imagerie médicale choisi par deep Learning	Réaliser l'analyse de l'examen choisi par Deep Learning	550	3	3	Mise en inférence des images de l'examen médical au niveau du Deep Learning	6	ok	MUST
17	WebApp (simple)	Utilisateur	Afficher les résultats de l'analyse de la collection choisie par deep Learning	Visualiser les résultats de l'analyse par Deep Learning de la collection d'image choisie	500	3	2	affichage des résultats sur l'interface web	6	ok	MUST
18	WebApp (simple)	Utilisateur	conserver les résultats d'analyse	Exporter les résultats d'analyse de l'examen choisi soit par impression (papier, pdf), soit en format excel/word, ...	450	3	2	enregistrement des résultats sur un support informatique	6	ok	should
19	WebApp (upgrade to Quantimage-V2 integration)	Développeur	se connecter sur Kheops	Pouvoir accéder aux examens d'imagerie médicale	400	3	5	gestion de l'authentification	7	ok	should
20	WebApp (upgrade to Quantimage-V2 integration)	Utilisateur	Sélectionner un abulm et choisir une série examens disponible sur Kheops	Sélectionner les données de Kheops pour l'inférence	350	3	3	Série d'éléments clicables permettant les actions d'affichage et de sélection	7	ok	should
21	WebApp (upgrade to Quantimage-V2 integration)	Développeur	Importer et transfomrmer les images 2D en 3D	Mettre les images médicales à disposition dans le flux de l'inférence	300	1	3	Réutilisation des modules s'appuyant sur OKAPI développés précédemment && Passage automatiquement à l'étape suivante	7	Rejeté (passage automatique non implémenté)	should
23	WebApp	Développeur	Exécuter l'inférence	Réaliser l'analyse des examens médicaux sélectionnés	200	3	1	Exécussion automatique après l'étape de transformation 2D > 3D	7	ok	should
24	WebApp	Utilisateur	Visualiser/sauvegarder les résultats d'analyses	Utiliser, manipuler les depp features obtenues par l'analyses	150	3	2	Affichage dans un tableau et/ou sauvegarde des résulats	7	OK	should
25	WebApp	Développeur	Réaliser des tests de performances	Avoir connaissance des ressources utilisées, de la vitesse d'exécussion, ...	100			Fonction disponible ou à développer pour "surveiller" l'outil d'inférence	5		wish

Annexe 2 : Tableau d'évaluation des outils d'inférence

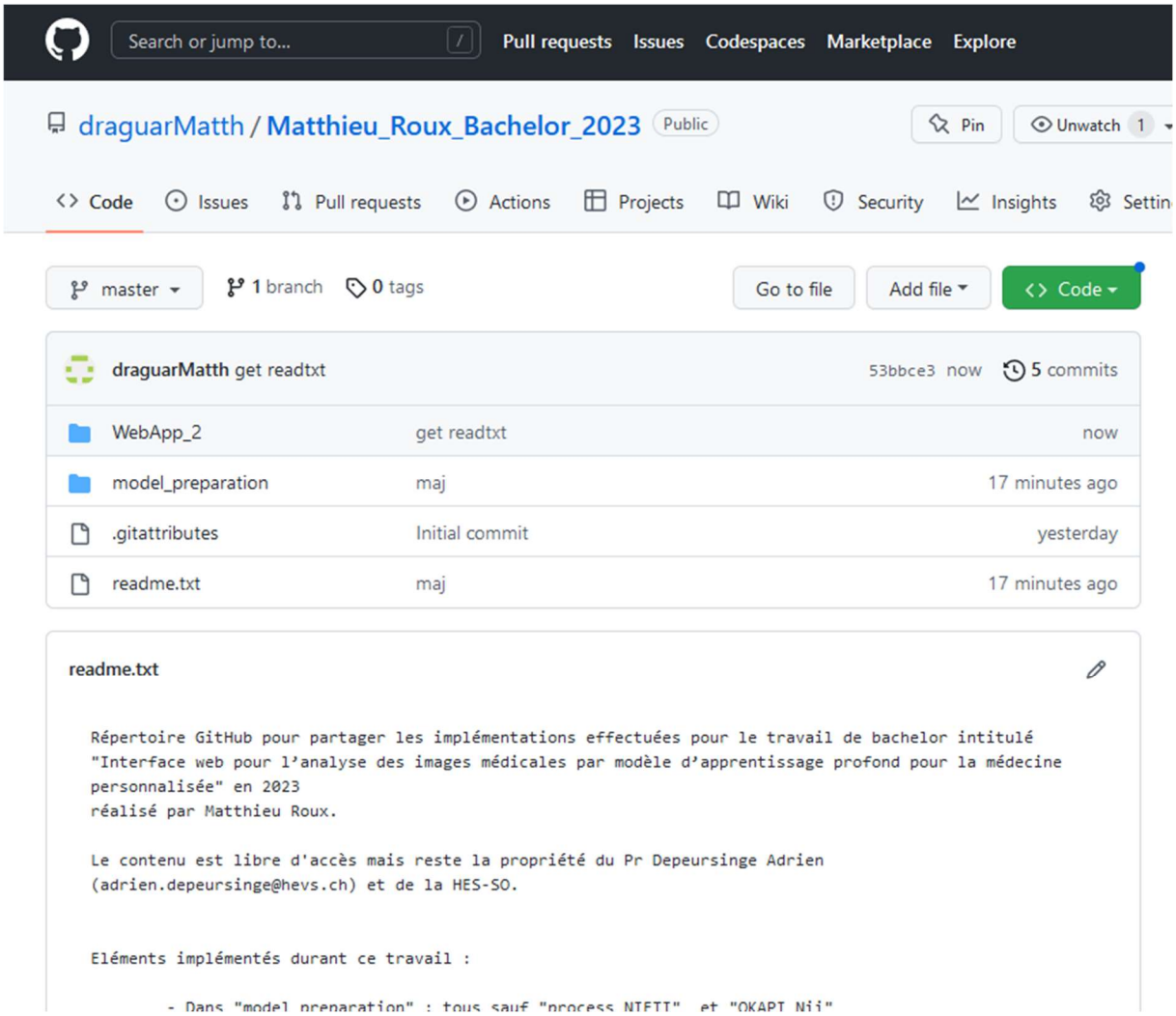
		Triton Inference Server	(KuberFlow) KFServe	AMD inference server (XILINX)	DeepDetect	BentoML	Multi Model Server (MMS)	Seldon Core	TensorFlow Model Serving	TorcheServe	Determined AI	AI inference server (SIEMENS)	Cortex
Gestion des modèles DL	Déploiement	1	1	1	1	1	1	1	1	0	0	1	1
	Gestion d'inférence	1	0	1	0	0	1	0	0	1	1	1	0
Plateformes d'hébergement	Machine locale	1	1	1	1	1	0	0	1	1	1	0	0
	Machine virtuelle	1	0	0	0	0	1	0	0	0	0	0	0
	Cloud	1	1	1	1	1	1	0	0	1	1	1	1
Types de connexion (protocoles)	HTTP	1	1	1	1	1	1	1	1	1	1	0	0
	gRCP	1	1	0	0	1	0	1	1	0	0	0	0
	Hors ligne	1	1	1	0	1	0	0	0	1	1		1
Conteneurisation	Docker, Kubernetes	1	0	1	1	1	1	0	1	0	0	0	0
Langages de développement de modèles DL	langage 1	1	1	1	1	1	0	1	1	0	1	1	0
	langage 2	1	1	1	1	1	0	1	1	1	1	1	0
	langage 3	1	1	0	1	0	0	0	0	0	1	0	0
	langage 4	1	1	1	0	1	0	0	0	0	0	1	0
	langage 5	1	1	1	1	0	0	0	1	0	0	1	0
Systèmes d'exploitation supportés	Ubuntu	1	1	1	1	1	1	1	1	1	0	0	1
	Windows	1	1	1	1	1	1	1	0	1	0	0	0
	Mac OS	1	1	1	1	0	1	1	0	0	0	0	0
	Linux	1	1	1	1	0	1	1	0	0	0	0	1
Autres	Composants additionnels	1	1	0	0	1	1	1	0	1	0	0	1

Sources annexe 2 :

- https://github.com/awslabs/multi-model-server
- https://developer.nvidia.com/nvidia-triton-inference-server
- https://github.com/triton-inference-server/server
- https://forestflow.ai/
- https://forestflow.github.io/ForestFlow/docs/inference.html
- https://www.deepdetect.com/
- https://github.com/jolibrain/deepdetect
- https://www.tensorflow.org/tfx/serving/
- https://www.bentoml.com/
- https://github.com/bentoml/BentoML
- https://github.com/Xilinx/inference-server
- https://xilinx.github.io/inference-server/main/index.html
- https://github.com/pytorch/serve
- https://pytorch.org/serve/
- https://kserve.github.io/website/0.9/
- https://github.com/kserve/kserve
- https://www.seldon.io/solutions/open-source-projects/core
- https://github.com/SeldonIO/seldon-core
- https://www.cortex.dev/
- https://www.dex.siemens.com/edge/manufacturing-process-industries/ai-inference-server?viewState=DetailView&cartID=&portalUser=&store=&cclcl=en_US&selected=edge

Annexe 3 : Répertoire GitHub du code développé durant ce travail

https://github.com/draguarMatth/Matthieu_Roux_Bachelor_2023.git



DECLARATION DE L'AUTREUR

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : M. Depeursinge Adrien et M. Schaer Roger.

Sierre, le 06 février 2023

Matthieu Roux