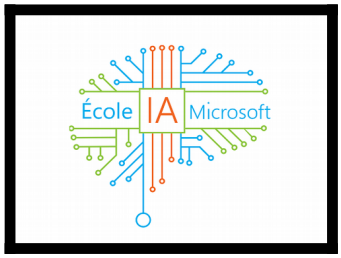


SIMPLON.CO

Projet Chef-d'œuvre: Conception d'une application de radiologie intelligente

École Microsoft Data/IA 2018-2020

Mathieu Lakshmanan

Sommaire

Introduction

Dans le cadre de la Formation Data/IA Microsoft/Simplon, il nous a été proposé de réaliser un projet "chef d'œuvre" permettant de mettre en pratique l'ensemble des connaissances accumulées lors de ces 19 derniers mois. J'ai pour ma part choisi de développer une application pour un client fictif mais une problématique bien réelle dans le domaine de la vision par ordinateur dans la santé, et plus spécifiquement dans la radiologie. L'e-santé était le domaine de mon entreprise d'alternance et c'est aussi le domaine dans lequel j'aimerais faire évoluer ma carrière, d'abord par affinité mais aussi pour l'impact sociétal positif que l'IA peut avoir sur la santé dans le monde réel.

En développant ce projet en entier et seul, j'ai pu mettre en application l'ensemble de mes compétences, de la collecte des données à leur analyse en passant par leur présentation et leur structuration en base de données dans une application complète et stimulante intellectuellement dans sa conception.

Besoins et objectifs

Contexte

Les avancées en apprentissage profond de ces dernières années ont trouvé en la radiologie un domaine d'application direct qui a un impact apparent : En reconnaissant des pathologies automatiquement, il serait possible de réduire la charge de travail des docteurs en radiologie, où sur certains domaines de la radiologie, certains modèles sont capables d'être aussi voire plus performants que des docteurs humains.

Si la recherche dans ce domaine fait de grandes avancées, son application en hôpital ou en clinique n'est encore qu'à ses premiers pas. En effet, il s'agit d'un des domaines d'application de l'IA pionniers où des questions éthiques se retrouvent immédiatement soulevées et alors que certains docteurs en radiologie poussent à son utilisation, d'autres s'en méfient. Si le domaine peine à avancer, c'est en partie dû au manque de compétences transversales entre l'informatique et la médecine. Malgré des progrès récents, en particulier au niveau de l'éducation des médecins ainsi que l'intégration de modèles avancés de vision par ordinateur dans les hôpitaux, les manques d'ergonomie et d'explicabilité de ces applications sont encore prévalents.

De plus, un des plus grands challenges de la vision par ordinateur appliquée à la radiologie est le manque de données labellisées correctement. Si certaines bases de données libres (explorées dans ce projet) existent dans ce domaine, trop de données ne sera jamais un problème réel.

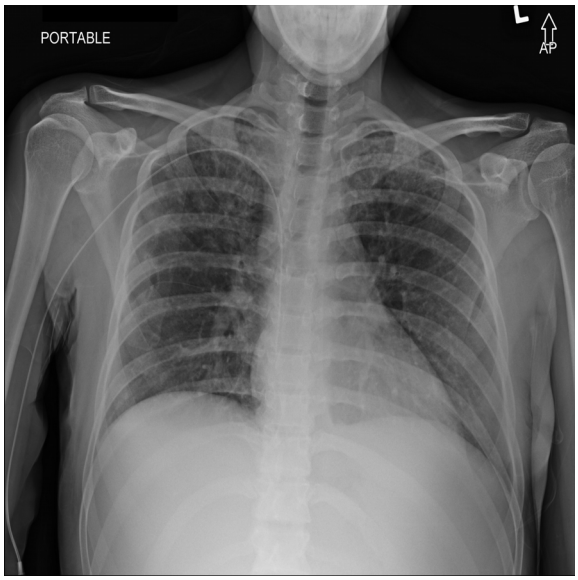
Motivation et enjeux

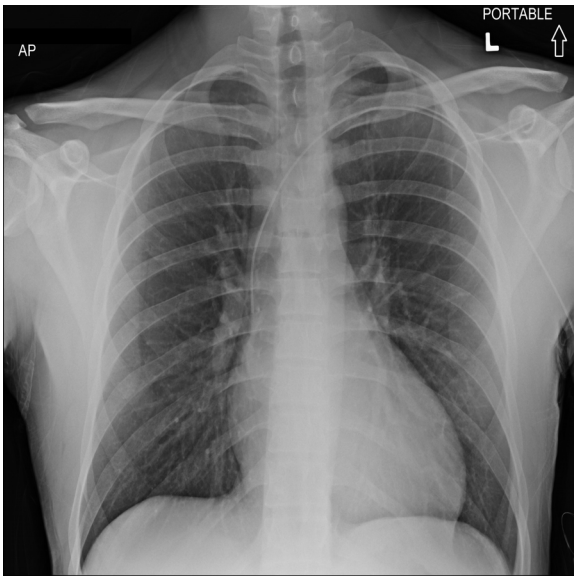
Après ce postulat, il m'est venu à l'idée de développer un application permettant de familiariser des radiologues avec cet outil puissant mais pas pour autant mystique qu'est l'IA, tout en accumulant une base de donnée open-source d'images de rayons X du thorax labellisée par des professionnels du métier en coopération avec une IA.

Ce projet est motivé par une passion pour la vision par ordinateur; il s'agit à mon sens d'un des domaines d'application les plus prometteurs de l'IA, présentant des challenges logiciels comme matériels, les deux étant primordiaux dans l'avancée technologique de l'informatique au sens large. J'ai donc pu, à travers ce projet, en apprendre beaucoup sur les dernières méthodes dans ce domaine, essayer d'en développer moi-même et ait bien sur eu l'occasion de les mettre en application dans une solution intelligente.

Outre la technique c'est aussi son lien direct avec le domaine de la santé qui m'a motivé: était-il possible à mon échelle de développer de moi même une application utile dans un domaine social de la plus haute importance, et ceci dans une démarche open-source et orientée vers l'utilisateur: ici le médecin radiologue. Ce serait mentir que de dire que la récente crise mondiale de santé publique ne m'a pas inspiré, malgré le fait que celle-ci a précédé mon idée d'origine, elle m'a influencé malgré moi. Il était évident que ce genre d'application de partages de données cliniques et radiologiques publiques ont manqué. Si je n'ai pas personnellement l'ambition de résoudre ce problème à moi seul, l’intégrer dans ma réflexion dans la conception de cette application était à mon avis une démarche intéressante à suivre.

C’est donc avec ces observations que je me suis lancé dans le développement d’une application de classification et recherche inversée et mise en place de base de donnée libre pour des images rayon X du thorax.





Exemples des images traitées

Objectif et contraintes

Les objectifs de développement de cette application se divisaient en deux grandes parties : d'abord créer un "Produit Minimum Viable" (MVP) composé de 3 modules conteneurisés: un module "front-end", un module d'inférence de modèle d'apprentissage profond ainsi qu'un module de base de donnée. Une fois ce "MVP" terminé, la deuxième partie consiste en l'amélioration incrémentale de ces 3 modules.

Le premier module "front-end" a pour contrainte d’être facile à utiliser, simple et

ergonomique.

Le deuxième module d'inférence a pour contrainte d'être facilement déployable, facilement améliorable en ayant une fonctionnalité simple: une image en entrée et un tenseur en sortie.

Le troisième module de base de donnée a pour contrainte de pouvoir être facilement partagé sans pour autant être la cible d'abus, et ce tout en restant déployable et exploitable facilement. Cette base de donnée traitant des données de santé se doit d’être en accord avec la RGPD.

Présentation du projet

Travail préliminaire

Veille et état de l'art

-Base de données

Au vu de la simplicité de la structure de la base de données l'utilisation de base de donnée SQL est une évidence. Cela nous permettra d'avoir une base de donnée facilement requêtable via python avec un connecteur en plus d'être performante et robuste. Les données les plus importantes étant des images nous avons deux possibilités de stockage: via une la transformation d'un fichier binaire (ou blob) ou bien le stockage dans sur le volume de stockage (lié à l'instance Docker dans notre cas pour des raisons de simplicité) et dans la base de donnée SQL le chemin absolu de la dite image. Après quelques recherches sur StackOverflow notamment, il s'avère qu'il est de meilleure pratique d'utiliser la seconde solution.

-IA

Pour la classification, deux solutions sont possible: la construction en partant de zéro d'un réseau de neurones à convolution (CNN) puis réglage fin des hyperparamètres afin d'obtenir un résultat satisfaisant ou bien faire de l'apprentissage par transfert via un réseau pré-entraîné (par exemple un ResNet[1]. La différence de résultats est faible[2] ,

mais le temps d'entraînement nécessaire au vu de la taille du dataset d'entraînement (~11go) rendait cette méthode complexe au vu des ressources disponibles.

Pour le système de recherche d'image inversée, j'ai choisi d'utiliser la proximité cosine[3] utilisée dans les moteurs de recherches de nos jours. Pour obtenir des vecteurs représentatifs des images (vecteurs latents), j'avais le choix, comme pour la classification, entre utiliser un réseau pré-entraîné et prendre la sortie de l'avant dernière couche de celui-ci[4], ou bien entraîner un auto-encodeur de moi même, puis extraire la partie encodeur pour l'inférence de vecteurs latents. Pour des raison similaires à la classification, j'ai choisi la solution avec des réseaux pré-entraînés, d'autant plus que le problème de taille de jeu de données est doublé : Les images doivent être chargées en entrée comme en sortie.

Choix des technologies

-Module BDD

Le choix de ce côté s'impose de part sa simplicité et sa robustesse ainsi que la flexibilité d'utilisation : mySQL est une évidence. La base de données pourra être requêtée et mise à jour via python et un connecteur. À ce niveau, le choix du connecteur importe peu : j'utiliserai le connecteur mySQL intégré a Flask. Pour la sécurité et le développement de la base de données, j'utilise des procédures PL/SQL enregistrées. Ceci a pour avantage de faciliter la gestion des droits utilisateurs, simplifier le code du serveur ainsi que protéger des injection SQL.

-Module entraînement/inférence:

Du fait de la taille du dataset d'entraînement (11go environ) J'avais besoin d'une méthode optimisée pour intégrer et faire le preprocessing. J'ai donc utilisé la méthode « flow » de TensorFlow du fait de la structure de mes données (Fig 1.). Cette fonction permet une injection itérative des données dans la mémoire GPU, évitant les débordements de mémoire avec un pré-process « à la volée » des images. Ce pré-process consiste simplement en une normalisation. J'ai choisi de ne pas déformer/retourner les images car ce n'est pas un cas réaliste en radiologie.

Figure 1: Arbre de structures des données

Pour l'inférence j'utilise Tensorflow-Serving en protocole gRPC (gRPC Remote Procedure Calls)[5] pour récupérer un JSON après une requête POST sur un serveur Flask qui s'occupera de transformer le résultat de l'inférence en protocole REST pour le front end. Ce serveur Flask s'occupera aussi du calcul de la similarité cosine.

-Module application

L'utilisation de Flask est une évidence pour l'intégration facile des autres modules.

Quelques modules jQuery seront nécessaires pour une application réactive.

Enfin pour finir afin de faciliter le déploiement, chaque module sera instancié dans un conteneur Docker. Le déploiement de ses modules se fait donc via docker-compose.

Développement

Collecte et préparation des données

La collecte des données peut être divisée en deux parties : les données d'entraînement et les données "utilisateur".

Pour la première partie, j'ai commencé par explorer le jeu de données CheXray[7] mais celui ci s'est avéré mal labellisé[8], je me suis donc tourné vers un jeu de données mieux réputé et utilisé dans le milieu de deep learning appliqué à la radiologie, CheXpert[6]. Celui-ci contient 224,316 images venant de 65,240 patients(Table 1.) pour 14 pathologies thoraciques différentes :

- Non trouvé
- Cardiomédiastin élargi
- Cardiomégalie
- Opacité pulmonaire
- Lésion pulmonaire
- Œdème
- Consolidation
- Pneumonie

- Atelectasie
- Pneumothorax
- Effusion pleurale
- Pleurale autre
- Fracture
- Dispositifs de soutien

Il s'agit d'un jeu de données spécialisé fait par des professionnels du deep learning et de la radiologie. Il a donc l'avantage d'être particulièrement propre et complet en plus d'être facile à utiliser grâce à un fichier .csv qui contient des labels, identificateurs des patients, leur âge, ainsi que des données sur le type de l'image.

Pathology	Positive (%)	Uncertain (%)	Negative (%)
No Finding	16627 (8.86)	0 (0.0)	171014 (91.14)
Enlarged Cardiom.	9020 (4.81)	10148 (5.41)	168473 (89.78)
Cardiomegaly	23002 (12.26)	6597 (3.52)	158042 (84.23)
Lung Lesion	6856 (3.65)	1071 (0.57)	179714 (95.78)
Lung Opacity	92669 (49.39)	4341 (2.31)	90631 (48.3)
Edema	48905 (26.06)	11571 (6.17)	127165 (67.77)
Consolidation	12730 (6.78)	23976 (12.78)	150935 (80.44)
Pneumonia	4576 (2.44)	15658 (8.34)	167407 (89.22)
Atelectasis	29333 (15.63)	29377 (15.66)	128931 (68.71)
Pneumothorax	17313 (9.23)	2663 (1.42)	167665 (89.35)
Pleural Effusion	75696 (40.34)	9419 (5.02)	102526 (54.64)
Pleural Other	2441 (1.3)	1771 (0.94)	183429 (97.76)
Fracture	7270 (3.87)	484 (0.26)	179887 (95.87)
Support Devices	105831 (56.4)	898 (0.48)	80912 (43.12)

Table 1: Répartition des labels dans le jeu de données CheXpert

Pour la deuxième partie, il s'agira des données fournies par l'utilisateur. Non pas leur données personnelles mais plutôt un système de labellisation et de commentaires. Après la classification d'une image et son stockage sur le serveur, le médecin pourra valider ou non le choix du modèle ainsi que noter un commentaire public lié à cette image.

La préparation des données se fera en sous-échantillonnant les images en 224x224x3 images afin d'avoir un temps d'entraînement raisonnable. Ces images seront ensuite normalisées. La séparation des données d'entraînement et de validation est de 90 %/10 % avec 200 images de test, avec un faible échantillon (100 images max) qui sera

utilisé comme premières images mise en ligne sur l'application par un médecin fictif - avec une ou deux images mises de côté pour la démonstration.

De plus, s'agissant d'une problématique multi-classe, chaque image est associée a une liste de labels notés 1 pour positif, 0 pour négatif et -1 pour incertain (Table 2.). Plusieurs approches pour ce problème sont possible, j'ai choisi dans un soucis de simplicité de remplacer les labels incertains par des labels positifs.

	Path	Sex	Age	Frontal/Lateral	AP/PA	No Finding	Enlarged Cardiomediastinum	Cardiomegaly	Lung Opacity	L Le
0	CheXpert-v1.0-small/train/patient00001/study1/...	Female	68	Frontal	AP	1.0	NaN	NaN	NaN	Na
1	CheXpert-v1.0-small/train/patient00002/study2/...	Female	87	Frontal	AP	NaN	NaN	-1.0	1.0	Na
2	CheXpert-v1.0-small/train/patient00002/study1/...	Female	83	Frontal	AP	NaN	NaN	NaN	1.0	Na
3	CheXpert-v1.0-small/train/patient00002/study1/...	Female	83	Lateral	NaN	NaN	NaN	NaN	1.0	Na
4	CheXpert-v1.0-small/train/patient00003/study1/...	Male	41	Frontal	AP	NaN	NaN	NaN	NaN	Na

Table 2: Extrait du tableau de labels du jeu de données

Les images envoyées par les utilisateurs sont quand à elles redimensionnées en 224x224x3 pour l'inférence et en taille réelle pour observation et comparaison.

Conception de la base de données

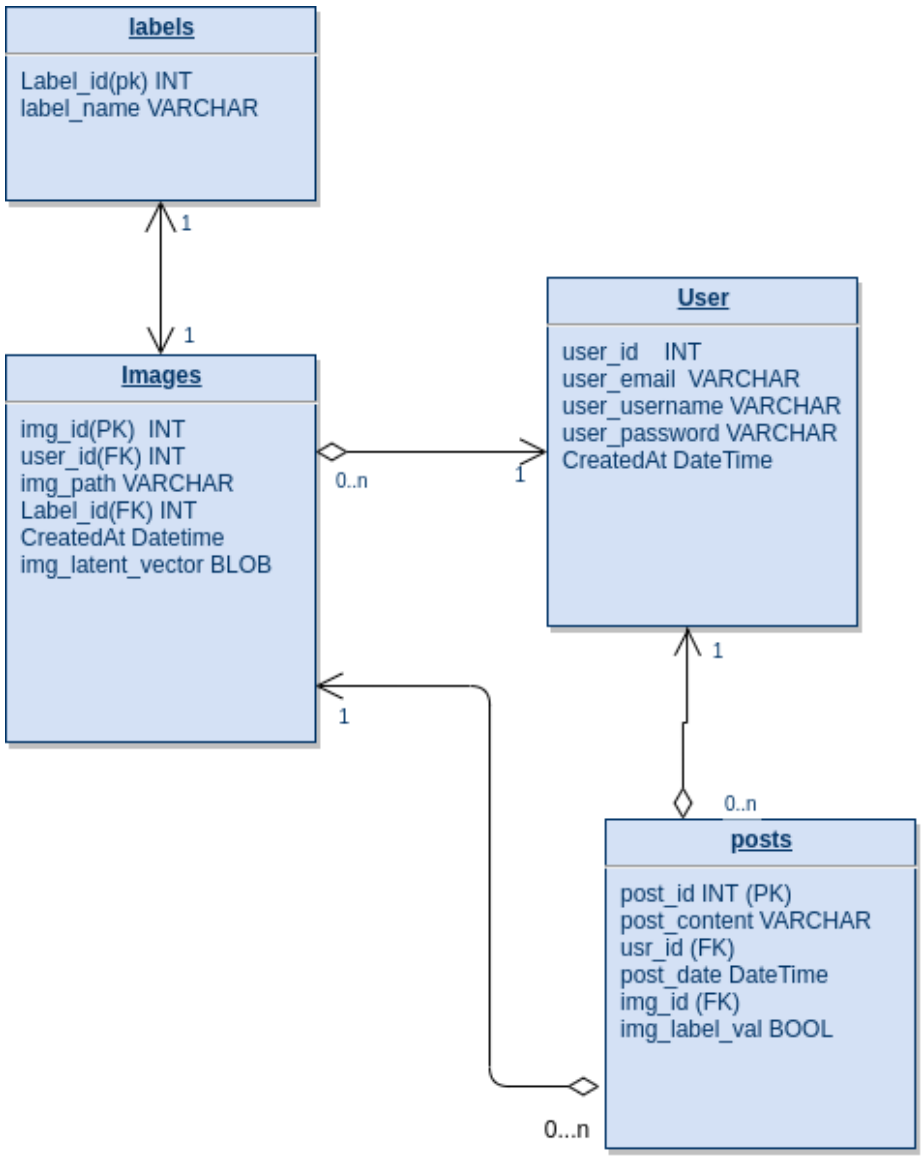


Figure 2: MCD de la Base de données

Des procédures PL/SQL m'ont permis de définir les opérations possibles sur la base de données :

-Ajout/suppression d'utilisateur

-Ajout/suppression d'image

-Ajout/suppression de vecteur latent

-Ajout/suppression d'un label

-Sauvegarde et export de la BDD

Afin de rester dans la norme RGPD, les données patients sont anonymisées.

Conception de l'algorithme de classification

Comme vu dans l'état de l'art, j'ai utilisé un réseau pré-entrainé et avec une opération de transfer learning. Nous gelons pour cela les couches profondes de différents réseaux et autorisons l'entraînement des deux dernières couches, ainsi qu'un changement de la dernière couche de classification. Différents réseaux ont été testés:

-Xception[9]

-ResNet50V2[10]

-InceptionV3[11]

-MobileNetv2[12]

-DenseNet121[13]

Ces 5 modèles ont pour avantage d'être relativement légers (<100mb), ce qui permet un entraînement et une inférence plus rapide.

J'ai finalement choisi InceptionV3 (Fig 3.) qui m'a donné les meilleurs résultats sur les données de test.

	precision	recall	f1-score	support
No Finding	0.86	0.90	0.88	6778
Enlarged Cardiomediastinum	0.78	0.58	0.66	6300
Cardiomegaly	0.73	0.69	0.71	8838
Lung Opacity	0.83	0.86	0.84	24478
Lung Lesion	0.93	0.99	0.96	8706
Edema	0.81	0.74	0.78	15281
Consolidation	0.79	0.67	0.73	12283
Pneumonia	0.76	0.62	0.69	6819
Atelectasis	0.79	0.78	0.79	17804
Pneumothorax	0.78	0.64	0.71	5840
Pleural Effusion	0.86	0.80	0.83	19978
Pleural Other	0.92	1.00	0.96	5309
Fracture	0.96	0.95	0.95	7754
Support Devices	0.81	0.92	0.86	24981
micro avg	0.83	0.81	0.82	171149
macro avg	0.83	0.80	0.81	171149
weighted avg	0.83	0.81	0.82	171149
samples avg	0.81	0.80	0.79	171149

Figure 3: Performance d'InceptionV3 sur les différentes pathologies

Conception de l'application front-end

Je me suis basé sur un code front-end préexistant d'une application libre sur github de reconnaissance d'image avec un VGG16, qui était suffisante dans le cadre de ce projet (Fig 4.).

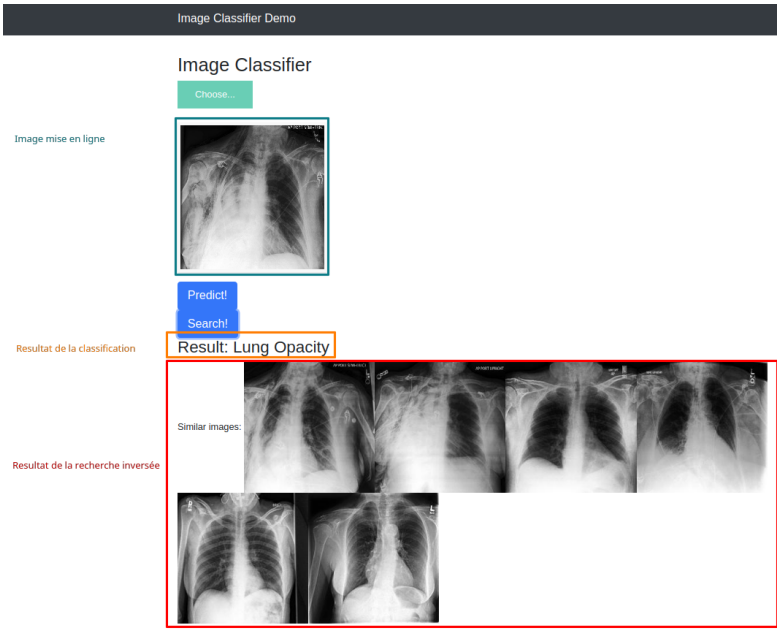


Figure 4: Capture d'écran de l'application

Conception de l'architecture de déploiement

Une fois chaque module de mon application développés, j'ai décidé de les mettre en forme pour déploiement dans des conteneurs. J'ai choisi docker-compose pour la mise en place de différents conteneurs ainsi que la configuration du réseau et des volumes de stockage partagés entre les différentes instance. L'architecture de ce déploiement - et accessoirement de toute l'application est la suivante (Fig 4.):

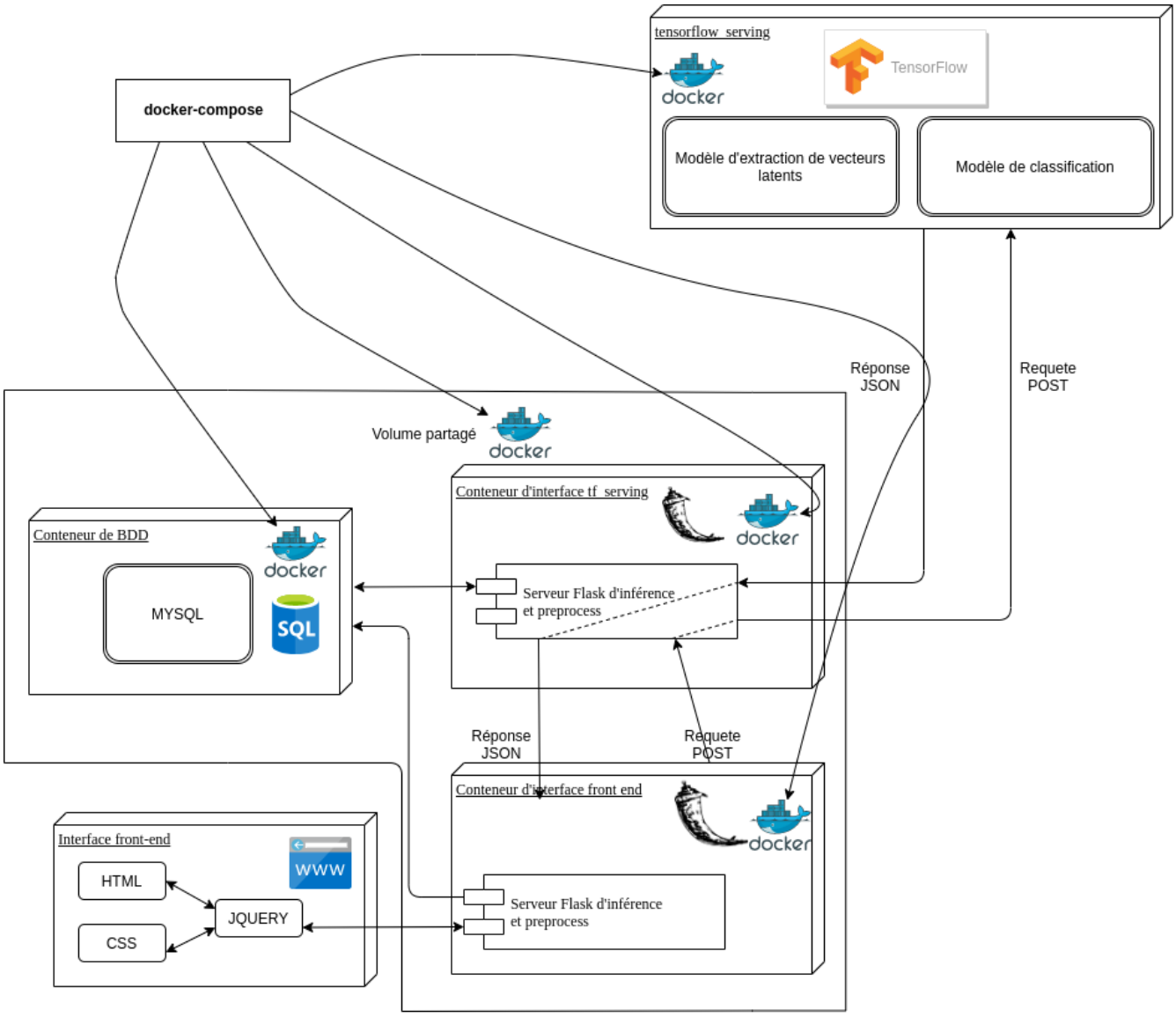


Figure 5: Architecture de déploiement

Gestion de projet

Planification

En divisant ce projet en 3 sous modules, j'ai pu avancer a mon rythme en changeant de module à ma commodité, ce qui m'a permis de changer de sujet sans me lasser en restant bloqué sur un seul sujet, les trois modules étant relativement indépendant les uns des autres car je les ai définies en "boites noire" dans leur interactions: chaque module est défini par ses entrées et ses sorties. Ces modules ont été divisées en fonctionnalités et sous fonctionnalités classifiées en deux sous catégories: "MVP", et "produit fini". La première catégorie étant la plus prioritaire.

Outils de gestion

Du fait de ma planification, Trello(Fig 5.) était un choix évident qui me permettait de

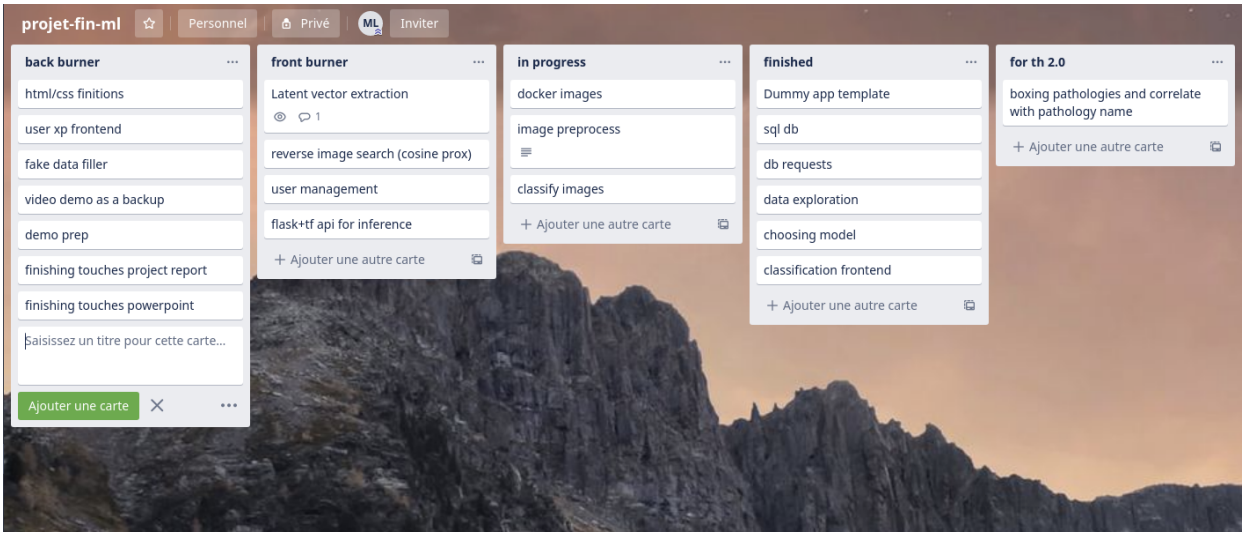


Figure 6: Exemple d'avancement du développement sur Trello

suivre mon avancement de manière flexible du fait que c'était un projet développé par une seule personne. Au fur et à mesure de l'avancement du projet j'ai pu fractionner les tâches définies en début de projet : plus j'explorais une tâche, plus je me rendais compte des sous tâches nécessaires au développement

Pour la sauvegarde et le versionnage, j'ai utilisé GitHub.

Conclusion et axes d'amélioration

Par ce projet j'ai pu explorer des problématiques liées entre l'IA et la médecine. Que ce soit au niveau, des choix des jeux de données ou des choix des algorithmes, tout doit se faire avec un réflexion très différent qu'un cas plus classique tel que la détection des visages. J'ai aussi pu beaucoup apprendre sur les techniques de déploiement d'application d'apprentissage machine, notamment grâce à docker et docker compose. Mes essais sur la phase d'entraînement m'ont aussi poussé à en apprendre sur la gestion de la mémoire vidéo lorsque l'on entraîne un modèle sur GPU. En résumé, malgré le fait que le but de ce projet était de démontrer mes connaissances acquises lors de cette formation, celui-ci m'a exposé à des problématiques nouvelles et m'a donc aussi fait apprendre de nouvelles technologies.

Du côté des axes d'améliorations, je compte bien mener ce projet à son terme et donc le rendre fonctionnel à 100%: j'ai à ce jour une structure d'application solide, où de part son architecture je peux y additionner des modules complémentaires très facilement. Une de mes priorités serait de créer une interface utilisateur fonctionnelle avec une API visuelle permettant de télécharger des images mises en lignes et classifiées. Un autre module important serait une mise en place d'une procédure de réentrainement des modèles sur les nouvelles données de façon automatique.

D'autres idées d'amélioration, sans ordre particulier:

- Amélioration de l’interface utilisateur
- Entraînement plus fin des modèles sur un machine adaptée
- Ajout de modèles pour d’autres types d’image de radiologie (scans CT, imagerie oculaire…)
- Intégration de Kubernetes pour un déploiement en situation réelle
- Processing et utilisation des commentaires par de la NLP
- Intégration des données cliniques des patients

Bibliographie

[1] A. Rios, R. Vanderpool, P. Shaw, and R. Kavuluru, “A Multi-Label Classification Approach for Coding Cancer Information Service Chat Transcripts,” *Proc. Int. Fla. AI Res. Soc. Conf. Fla. AI Res. Symp.*, vol. 2013, pp. 338–343, May 2013.

[2] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, “Transfusion: Understanding Transfer Learning for Medical Imaging,” *ArXiv190207208 Cs Stat*, Oct. 2019, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1902.07208.

[3] R. S. R. Machado, “Image visual similarity with deep learning: application to a fashion ecommerce company,” 2017.

[4] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, “Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 105–109, Jan. 2016, doi: 10.1109/LGRS.2015.2499239.

[5] “Introduction to gRPC,” *gRPC*. https://grpc.io/docs/what-is-grpc/introduction/ .

[6] J. Irvin *et al.*, “CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison,” *ArXiv190107031 Cs Eess*, Jan. 2019, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1901.07031.

[7] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “ChestX-ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases,” p. 10.

[8] ~ Luke Oakden Rayner, “Exploring the ChestXray14 dataset: problems,” *Luke Oakden-*

Rayner, Dec. 18, 2017. https://lukeoakdenrayner.wordpress.com/2017/12/18/the-chestxray14-dataset-problems/ .

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *ArXiv161002357 Cs*, Apr. 2017, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1610.02357.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," *ArXiv160305027 Cs*, Jul. 2016, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1603.05027.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *ArXiv151200567 Cs*, Dec. 2015, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1512.00567.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *ArXiv180104381 Cs*, Mar. 2019, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1801.04381.

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *ArXiv160806993 Cs*, Jan. 2018, Accessed: Jun. 07, 2020. [Online]. Available: http://arxiv.org/abs/1608.06993.