Université de Bordeaux

Master de Bioinformatique

Filières de la Bioinformatique

Année scolaire 2020-2021

Le deep learning pour la classification d'images radiographiques d'implants articulaires d'épaule

Marine Alves de Barros, Mathieu Bolteau, Alexandre Cornier,
Martin Drancé, Abdelghani Neuhaus

Sommaire

Introduction

L'arthroplastie totale d'épaule [1] est une opération courante qui permet de remplacer la tête de l'humérus dans le but de reconstruire l'articulation de l'épaule. Ces opérations interviennent dans le cadre d'arthrose sévère ou de dommages à l'épaule ne pouvant pas être soignés. Pour assurer le bon déroulement de l'opération, et le suivi de celle-ci, des séries de radiographies sont faites avant et après l'opération. Aujourd'hui, plusieurs constructeurs proposent différents types de prothèses dans le but de s'adapter au mieux à chaque patient. En contrepartie, si le dossier médical n'est pas accessible (opérations à l'étranger) ou mal documenté, les interventions de suivi peuvent être rendues compliquées pour le praticien qui ne connaît pas la provenance de la prothèse. L'analyse des images de radiographies pour déterminer le type de prothèse demande énormément de temps, d'efforts et ne doit laisser place à aucune erreur. L'objectif de ces travaux de recherche est de déterminer la fiabilité de différentes méthodes d'apprentissage automatisées appliquées à la reconnaissance de ces prothèses.

Ces dernières années ont vu croître les performances des techniques de vision par ordinateur, en grande partie grâce aux avancées dans le domaine des réseaux neuronaux convolutifs (CNN). Ces réseaux, contrairement aux réseaux de neurones artificiels ou récurrents, sont directement inspirés par le fonctionnement du cortex visuel des animaux. L'autre avancée majeure qui a permit l'utilisation des CNN à grande échelle est la technique de l'apprentissage par transfert. Un réseau neuronal convolutif peut apprendre une nouvelle tâche en se basant sur ce qu'a appris un autre CNN. C'est de cette manière que sont nées différentes grandes familles de modèles qui peuvent être ré-utilisées pour de nouvelles tâches. Avant la publication de ces travaux, aucune de ces méthodes n'avaient été utilisées pour classer des radiographies d'implants d'épaules.

Cet article [2] étudie la comparaison de sept architectures différentes de réseaux convolutifs, ainsi qu'une comparaison entre les résultats obtenus par ces réseaux avec ceux d'algorithmes d'apprentissage qui ne sont pas des algorithmes de *deep learning*. En plus de la comparaison entre ces différentes méthodes, chaque CNN est testé avec différents paramètres tel que l'utilisation ou non d'apprentissage par transfert ou d'augmentation des donnéés. Pour chaque test, plusieurs indicateurs de la qualité des prédictions sont calculés et comparés entre eux. Chaque indicateur permet d'évaluer un aspect spécifique du modèle (précision, sensibilité, spécificité, ...).

Le code crée pour réaliser la comparaison avec les résultats obtenus dans l'article est disponible ici : `https://github.com/MDrance/Filbi-project`.

Partie 1

Présentation des outils et des données

1.1 Algorithmes de machine learning non *deep learning*

Les techniques de machine learning dites "classiques", par opposition aux techniques de deep learning, sont des outils puissants de prédiction et de classification basés sur la recherche d'un modèle mathématique optimal, qui représentera au mieux la réalité. Ces techniques permettent en général de produire des prédictions qui s'appuient sur des données relativement simples. Basées sur un apprentissage brut de celles-ci, elles manquent d'efficacité lorsque que les données sont trop complexes, désorganisées ou lorsqu'il faut intégrer au modèle des cas particuliers [3]. Cependant, ces méthodes restent celles qui sont le plus simple à mettre en place et il est intéressant d'évaluer leurs performances sur chaque nouveau cas d'étude.

Nous avons réalisé l'analyse des résultats de deux algorithmes de machine learning : le k-Nearest Neighbors (kNN) et le Random Decision Forest (RF).

Le (kNN) est un algorithme d'apprentissage non-supervisé qui permet de construire des classes. Chaque nouvelle donnée est classée en fonction du nombre de voisins, noté K. La classification d'une donnée est réalisée selon les classes et les valeurs de ses K voisins. Les algorithmes basés sur le kNN ne reposent pas sur des modèles mathématiques pour réaliser la prédiction : la classification se fait par l'évaluation des distances entre la donnée à classer et ses K voisins.

Les Random Forest sont des algorithmes d'apprentissage supervisé qui permettent aussi de construire des classes. La différence réside dans le fait que la répartition des labels (classes à prédire) est donnée à l'algorithme pour que celui-ci affine sa prédiction en fonction de ses erreurs. Ces modèles sont construits à partir d'une multitude d'arbres de décision, chaque arbre de décision étant d'abord créé aléatoirement puis optimisé en fonction des erreurs faites lors de la classification. Au final, la prédiction retenue est celle qui a été la plus récurrente au sein de tous les arbres de décision.

1.2 Algorithmes de deep learning

Pour palier aux faibles performances des méthodes de machine learning décrites précédemment dans le domaine du *computer vision*, des architectures bien particulières de réseaux de neurones sont apparues : les réseaux neuronaux convolutifs (CNN) [4]. Plutôt que de créer des réseaux de neurones où chaque neurone est connecté à tous ceux de la couche suivante, les CNN s'appuient sur des couches supplémentaires pour extraire de l'image ses caractéristiques essentielles. Ainsi, en passant par différentes étapes de convolutions et de *pooling*, l'information contenue initialement dans l'image va petit à petit être filtrée pour en conserver uniquement l'essentiel (pour ensuite la décrire). Ce "résumé" de l'image va ensuite être transmis à une couche de neurones *fully connected* [5] qui va réaliser la phase d'apprentissage à partir de ces informations "essentielles".

Depuis la création du premier réseau convolutif, plusieurs autres types d'architectures ont émergées comme le VGG [6], ResNet [7] ou encore DenseNet [8]. Chacune de ces architectures peuvent produire des prédictions *from scratch* mais permettent aussi l'utilisation de l'apprentissage par transfert. En effet, tous ces modèles auront été préalablement entraînés à partir de la base de données ImageNet [9], qui contient plusieurs millions d'images classées par catégories.

Nous avons réalisé des prédictions en utilisant trois types de réseaux de neurones : un réseau neuronal artificiel et deux réseaux de neurones convolutifs (ResNet-15 et un *custom*). Contrairement à un réseau convolutif, un réseau neuronal artificiel est utilisé principalement pour réaliser des prédictions (bourse, prix immobilier, ...) ou des classifications de données. La classification d'images avec ce type de réseau demande une quantité de données et de traitement très volumineuse, bien qu'il puisse être utilisé pour la reconnaissance de motifs (par exemple, le jeu de données du MNIST) [10].

1.3 Les moyens d'évaluation d'un modèle

La qualité des prédictions ne repose pas uniquement sur la précision. La précision permet seulement de connaître le pourcentage de prédictions correctes, mais elle ne donne pas d'indications concernant où le modèle réussit et où il se trompe. L'évaluation des prédictions fait donc intervenir d'autres concepts qui sont : les vrais et faux positifs et vrais et faux négatifs. Également, la sensibilité permet d'évaluer la proportion de vrais positifs (sur toute les prédictions X, combien sont réellement des X). la spécificité, quant à elle, permet d'évaluer la proportion de faux positifs (sur tous les X existants, combien ont correctement été prédit comme X). Ces informations peuvent être analysées à l'aide de deux outils :

- Le score F1, qui est meilleur quand la sensibilité et la spécificité ont une valeur correcte,
- L'aire sous la courbe ROC : plus elle est élevée, plus la sensibilité et la spécificité ont de bonnes valeurs.

Il faut noter que la courbe ROC est une technique adaptée à l'évaluation de classification binaire, même si il existe aujourd'hui des moyens d'obtenir des courbes ROC pour les problèmes multi-

classes [11].

1.4 Présentation et traitement des données

La collecte de données n'est pas la seule étape nécessaire précédant la phase d'entraînement d'un modèle. Ces données doivent être traitées, normalisées ou encore augmentées pour garantir un apprentissage optimal. Pour faciliter le traitement de l'information lorsque l'on a des images, il est important de normaliser la valeur des pixels ainsi que la taille des images. Plus important encore, la technique de *data augmentation* permet de générer, de manière artificielle, de nouvelles images dérivées des images d'origine (par rotation, zoom, ...) [12]. Cette dernière méthode représente une étape clé dans le domaine de l'imagerie médicale. En effet, il est souvent compliqué de constituer une base de données assez conséquente dans un tel domaine de recherche, où les techniques d'acquisition d'images sont chères et utilisées uniquement dans des cadres particuliers.

Les données utilisées lors du projet proviennent du site *"US Shoulder Prostheses"* [13]. Elles sont composées de 597 images de radiographies d'épaules présentant des prothèses issues de quatre constructeurs différents : Cofield (83), Depuy (294), Tornier (71) et Zimmer (149). Ces images ont pour la plupart une taille de 250x250 pixels, avec des ratios et des contrastes différents.

Partie 2

Mise en place des algorithmes

2.1 Machine learning non *deep learning*

Dans un premier temps, les deux modèles qui ont été utilisés sont le kNN et le Random Forest. Pour optimiser le processus et pouvoir comparer nos résultats à ceux qui font office de référence, il est important de conserver le même *workflow*. Avant l'utilisation de chacun de ces algorithmes, les données ont été d'abord augmentées, converties au format CSV puis normalisées.

L'augmentation des données est faite grâce au programme Python *dataAugment*. Il permet d'effectuer aléatoirement une rotation de plus ou moins 15 degrés sur chacune des images, puis de sauvegarder ces nouvelles images. Ainsi, la taille du jeu de données est doublée, sans pour autant créer de redondance au sein des images.

Dans un second temps, il est nécessaire de centraliser les données issues des images ainsi que d'associer à chaque image le nom du constructeur. Pour faire cela, le fichier Python *imgtocsv* permet de stocker toutes les valeurs de pixels image par image dans un fichier CSV, ainsi que le nom du constructeur associé à chaque image. Ainsi, chaque ligne du fichier correspond à une image, avec une colonne par pixel plus une dernière colonne qui correspond au label pour cette image.

La dernière étape de traitement des données est la normalisation. La normalisation permet de ramener chacune des valeurs de pixels dans une même gamme de valeurs, ici entre 0 et 1. Dans notre cas, les valeurs des pixels sont déjà contenues dans une même gamme de valeurs (0 et 255), cependant il reste important de normaliser ces données pour deux raisons :

- Des valeurs plus faibles sont plus faciles à manipuler par l'algorithme,
- Dans le cas où certaines images contiendraient du bruit, il est important que celui-ci n'est pas une influence significative durant l'apprentissage du modèle (confusion entre des pixels représentant l'os (blanc à 255) et du bruit).

2.1.1 k-Nearest Neighbors

En plus de la phase d'entraînement, les meilleurs hyper-paramètres du kNN ont été recherchés afin d'obtenir le modèle le plus performant possible. Deux hyper-paramètres ont été optimisés :

— Le nombre de voisins utilisés pour réaliser la classification (entre 5 et 40),
— La méthode de calcul de la distance entre les points (*Manhattan* ou Euclidienne).

Pour chacune des combinaisons possibles nous avons utilisés l'outil de *cross validation*. Ici, nous avons divisé les données d'entraînement en cinq sous-parties : quatre sous-jeux de données pour l'entraînement et le cinquième pour l'évaluation des combinaisons d'hyper-paramètres. Le meilleur modèle sera celui retenu pour la phase de prédiction avec les données tests.

2.1.2 Random Forest

La même procédure d'optimisation a été suivie pour l'algorithme de Random Forest ; cette fois trois hyper-paramètres ont été optimisés :

— Le nombre d'arbres dans la forêt (100, 200, 300, 400, 500),
— La fonction utilisée pour mesurer la qualité de l'ajout d'un nouvel arbre dans la forêt,
— L'utilisation ou non du *bootstrapping* pour la construction de chacun des arbres.

Ici aussi, seulement le meilleur modèle issue de la *cross validation* a été choisi pour évaluer la phase de prédiction sur les données tests.

2.2 Réseaux neuronaux

Après avoir utilisé différents algorithmes de machine learning "classiques", nous avons testé trois algorithmes de *deep learning* : un réseau de neurones artificiels (ANN), un réseau de neurones convolutifs *customisé* (CNN) et un réseau de neurones convolutifs d'architecture *ResNet*. Les résultats nous permettront de comparer notre travail au *gold standard* établit pour le CNN et ResNet ainsi que de comparer notre réseau neuronal artificiel aux convolutifs.

2.2.1 Réseau de neurones artificiel

Ce type de réseau neuronal est utilisé pour la prédiction (météo, immobilier, bourse, ...) mais aussi la classification de données (sous forme de tableaux). Il peut aussi être utilisé avec des formes géométriques simples. Il est organisé en plusieurs couches cachées, contenant un nombre d'unités décroissants. La dernière couche correspond à une seule unité cachée. Bien que ce type de réseau n'est pas été utilisé dans l'article, nous avons décidé de le tester car un des

réseaux convolutifs que nous testons (ResNet), est justement dérivé des ANN. Nous comparons ainsi un réseau utilisé pour la classification et la prédiction (ANN) ou un ANN spécialisé dans le *computer vision* (ResNet). Notre réseau artificiel s'organise comme suit :

— Une couche d'entrée contenant 250 unités cachées (car 250 pixels par image),
— Deux couches cachées successives contenant respectivement 180 et 150 unités cachées,
— Une étape de *dropout* ("inhibition" d'un pourcentage d'unités cachées d'une couche pour réduire l'*overfitting* sur la couche numéro 2)
— Quatre couches cachées successives contenant respectivement 128, 64, 64 et 32 unités cachées,
— Une étape de *dropout* sur la couche précédent la couche de sortie
— Une couche de sortie dont la fonction d'activation est la fonction sigmoïde.
— Les couches de cachée effectuant la prédiction dont la fonction d'activation est la fonction *ReLU*.

Comme précédemment, les données ont été normalisées puis augmentées aléatoirement pendant la phase d'entraînement de plusieurs façons : zoom léger, retournement horizontal, rotation de plus ou moins 15 degrés. La fonction de perte utilisée pour calculer les erreurs est la *Cross Entropy* et la fonction d'optimisation est *ADAM*. Le pas d'apprentissage est fixé à 0.05. Chacun des *batchs* d'entraînement contient soixante-quatre images.

2.2.2 Réseau de neurones convolutifs personnalisé

Ce type de réseau convolutif est le réseau "par défaut" (le premier décrit, [4]). Il consiste en une suite d'étapes de convolution et de *pooling*, qui aboutiront sur un ensemble de neurones tous connectés les uns aux autres. Ce sont eux qui réaliseront l'étape de prédiction. Les couches de convolutions font passer un noyau de convolution (ou *kernel*) d'une certaine taille sur l'image. Cela permet de simplifier l'information contenue dans la zone pour en extraire uniquement l'information importante. Les couches de *pooling* s'organisent aussi en zones d'une taille déterminée et permettent de réduire la taille de l'image, afin de la simplifier et de limiter le sur-apprentissage.
Notre CNN s'organise comme suit :

— Une couche de convolution de taille 3x3,
— Une couche de *pooling* de taille 2x2,
— Une couche de convolution de taille 3x3,
— Une couche de *pooling* de taille 2x2,
— Une couche de convolution de taille 3x3,
— Une couche de *pooling* de taille 2x2,
— Une couche de neurones entièrement connectés (avec la fonction d'activation ReLU),
— La couche de sortie effectuant la prédiction (avec la fonction d'activation softmax).

Comme précédemment, les données ont été normalisées puis augmentées aléatoirement pendant

la phase d'entraînement de plusieurs façons : zoom léger, retournement horizontal, rotation de plus ou moins 15 degrés. La fonction de perte utilisée pour calculer les erreurs est la *Cross Entropy* et la fonction d'optimisation est le *Stochastic Gradient Descent* (SGD). Le pas d'apprentissage est fixé à 0.01. Chacun des *batchs* d'entraînement contient seize images.

2.2.3 Architecture ResNet-15

ResNet est une architecture particulière de réseau convolutif qui se base sur un concept simple : toutes les couches d'un réseau de neurone ne sont pas nécessaires à chaque étape d'apprentissage. En effet, dans les CNN les plus puissants, on peut retrouver plusieurs dizaines de couches de convolution et de *pooling*. L'utilisation de chacune de ces couches à chaque étape d'apprentissage pose deux problèmes majeurs :

— Certains paramètres appris au fur et à mesure peuvent disparaître en atteignant la valeur de 0 et restent à 0 pendant tout le reste de la phase d'entraînement (on parle de *Vanishing Gradient Problem*),
— Les premières étapes d'apprentissage d'un CNN servent à isoler et grossir les caractéristiques de l'image. Dans un réseau trop important, cette multitude de couches échoue parfois à extraire les bonnes informations, amenant à des *features* trop réduites ou au sur-apprentissage par l'algorithme.

Dans une réseau type ResNet-15, certaines couches de convolutions et de *pooling* sont ignorées dans les premières phases d'apprentissage, pour être ré-activées au fur et à mesure que la phase d'entraînement avance.

En plus de sa structure particulière, cet algorithme peut être utilisé en étant déjà entraîné à partir de milliers d'images disponibles sur la base de données ImageNet [9]. Ainsi, toutes les couches successives seront conservées avec des paramètres déjà optimisés et seulement une couche de neurones complètement connectés sera ajoutée à la fin du processus. C'est uniquement cette couche qui sera entraînée à faire la bonne prédiction (avec notre jeu de données). On aura donc un réseau pré-entrainé. Ici, il a été entraîné en utilisant des *batchs* de 64 images. La fonction de perte à optimiser est la fonction de *Cross Entropy*. La fonction d'optimisation de gradient utilisé est le *Stochastic Gradient Descent* ; le pas d'apprentissage est égal à 0.01. De plus, ce pas d'apprentissage sera diminué d'un facteur trois toutes les sept *epochs*, permettant d'atteindre au mieux le minimum de la fonction de perte.

Partie 3

Analyses des résultats

3.1 Analyses des résultats de référence

Les résultats issus de l'article [2] sont répartis en cinq catégories sous forme de tableaux organisés en six colonnes. On retrouve dans chaque tableau le type de méthode utilisé, la précision, la sensibilité, la spécificité, le F1-score et l'aire sous la courbe ROC, et ceci pour chaque modèle. Le tableau 1 correspond aux résultats de la mesure des performances pour les techniques de machine learning non *deep learning*. On remarque que les quatre types de *classifiers* utilisés présentent tous une précision proche de 50%, le Random Forest étant celui qui présente les meilleurs résultats avec une précision de 56%. De même, la spécificité du Random Forest est la plus élevée avec un score de 62%, soit quatre points de plus que pour le Gradient Boosting.

Concernant le second tableau, nous nous intéresserons uniquement à la première partie (sans augmentation des données test). On constate que dans l'ensemble, la précision est bien meilleure avec les techniques de *deep learning*, atteignant au maximum 80,4% de précision pour le modèle NASnet. En revanche, en observant le tableau 5, qui représente les mêmes résultats mais sans aucune augmentation des données, le modèle NASnet obtient un score de précision de 64,5%, soit une baisse de 15,9 points. On observe une baisse significative de l'ensemble des outils d'évaluation sur chacun des modèles entraînés sans utiliser l'augmentation des données. Plus concrètement, la moyenne des scores de précision est de 76.9% pour les modèles entraînés avec augmentation des données, pour seulement 62% de précision moyenne sans aucune augmentation des données. Cette différence prouve que l'augmentation de données est un outil efficace pour optimiser l'entraînement d'un modèle en utilisant une quantité de données importante.

Enfin, le tableau 3 présente les résultats des performances des réseaux de neurones convolutifs sans pré-entraînements. Ceux-ci sont globalement inférieurs aux résultats du tableau 2 et cela pour toutes les méthodes d'évaluation. Par exemple, la précision moyenne sans apprentissage par transfert est de 54.5%, avec une précision maximale de 57% pour le modèle VGG-19. En comparaison, le moins bon score de précision pour les modèles pré-entraînés avec l'augmentation des données est de 74% pour VGG-16. Autre exemple avec la spécificité, qui est en moyenne de 40% pour les modèles sans pré-entraînements, alors que celle-ci est de 75,5% pour les modèles pré-entraînés. Ces résultats démontrent encore une fois l'utilité de l'apprentissage

par transfert dans le cadre où les données sont soit trop complexes, soit limitées.

3.2 Analyses de nos résultats

La Tableau 1 présente les résultats des différents algorithmes que nous avons implémentés et présentés dans le chapitre précédent. Des deux algorithmes de machine learning non *deep learning*, le k-Nearest Neighbor est celui qui produit les meilleurs résultats, avec une précision de 54%. C'est 5 points au dessus d'une prédiction aléatoire qui aurait 49% de chance de prédire la classe majoritaire. Ces résultats sont obtenus en utilisant la distance euclidienne (pour calculer la distance entres les points) et un nombre de voisins votant égal à 25. On constate que ce modèle est plus sensible (0.54) qu'il n'est spécifique (0.43), et ses prédictions donneront beaucoup de faux positifs. L'algorithme de Random Forest, quand à lui, obtient une précision maximale de 49% pour un nombre d'arbres par forêt égal à 500, avec l'utilisation du *bootstrapping*. Contrairement au kNN, la Random Forest est plus spécifique (0.55) que sensible (0.49) et a donc tendance à trouver peu de vrais positifs. Les courbes ROC sont présentées en figure 2.

Modèle	Précision [%]	Spécificité	Sensibilité	F1-score
kNN	54	0.43	0.54	0.43
Random Forest	49	0.55	0.49	0.41
CNN	55	0.53	0.31	0.40
ResNet	75	0.79	0.80	0.79
ANN	21	0.18	0.21	0.15

Tableau 1 : Mesures des performances pour les cinq algorithmes implémentés.

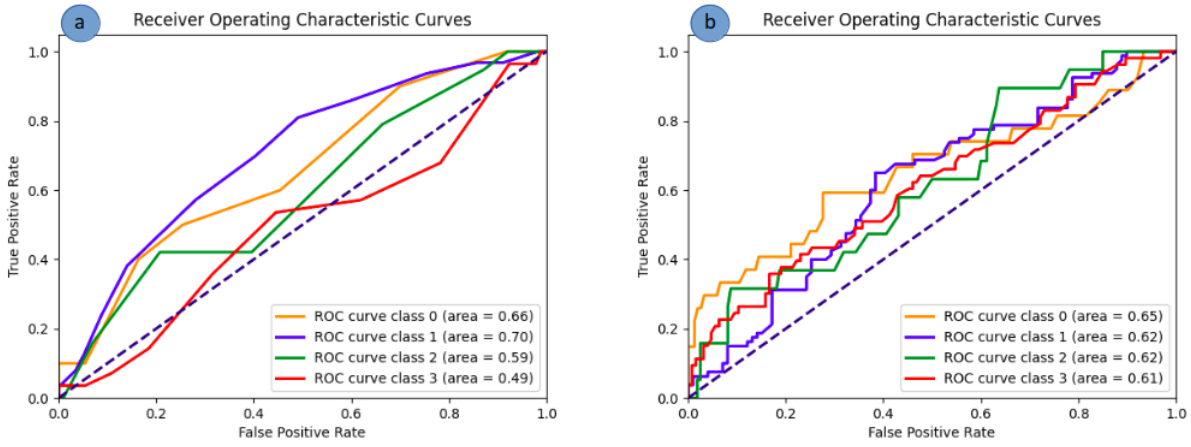

Figure 2 : Courbes ROC (*Receiver Operating Characteristic*) pour le k-Nearest Neighbor (a) et la Random Forest (b).

Les résultats de comparaison des modèles de deep learning sont visibles en figure 4. L'architecture ResNet-15 est celle qui montre les meilleurs performances avec une précision de 75%.

Ce modèle est autant sensible (0.80) qu'il est spécifique (0.79). On observe que ResNet apprend relativement rapidement, puisque c'est seulement au bout d'une dizaine d'epochs que ces résultats sont obtenus. Très rapidement, les erreurs commises sur les données d'entraînement ne diminuent plus et la précision stagne à cette valeur.

Le CNN *custom* permet d'obtenir une précision de 55%, soit seulement 6 points au dessus d'une prédiction aléatoire en utilisant que la classe dominante. Cet algorithme est bien plus spécifique (0.53) que sensible (0.31) et produira donc peu de vrais positifs. Ces résultats démontrent l'utilité d'utiliser une architecture ResNet dans les problèmes de classification d'images complexes. En effet, au bout de seulement une dizaine d'epochs, ce réseau convolutif n'apprend plus et ne parvient pas à extraire plus d'informations lui permettant de différencier chacune des classes d'images, là où pour un même temps d'entraînement, ResNet était déjà 20 points de précision au dessus.

Enfin, le réseau neuronal artificiel (ANN) produit de très mauvais résultats avec une précision maximale de seulement 21%. Bien que peu adapté à ce genre de problématique, un ANN peut tout de même parfaitement fonctionner pour classer des images simples, comme par exemple avec les données MNIST [10]. Ces résultats indiquent clairement que lorsque les images deviennent trop complexes, le réseau artificiel est un très mauvais modèle puisqu'il est 28 points en dessous d'une prédiction aléatoire de la classe dominante. L'algorithme n'arrive pas à isoler correctement l'information essentielle, mais se focalise sur toute l'image (et donc du bruit, le fond de l'image,...) et produit des résultats encore plus mauvais qu'une prédiction qui se baserait sur des statistiques.

Ces résultats démontrent, ANN mis à part, que ResNet-15 est le seul modèle testé qui permet de produire des résultats satisfaisants. En effet, le k-Nearest Neighbor, la Random Forest et le réseau convolutif "personnalisé" ne permettent que très peu d'améliorer la prédiction correcte de la bonne classe d'une image. De plus, les deux réseaux convolutifs arrivent très tôt, durant la phase d'entraînement, à la valeur maximale de précision qu'ils peuvent atteindre. Ceci indique que l'apprentissage doit être limité par plusieurs facteurs :

— Les différences entre les images de chaque classe sont très faibles,
— Les classes sont très mal équilibrées,
— Toutes les images ne sont pas standardisées car ayant été acquises avec différents outils et différents paramètres.

3.3 Comparaisons aux résultats de référence

Le comparatif entre nos résultats et ceux proposés par le papier sont présentés en figure 3. Globalement, nos résultats concordent avec ceux faisant office de *gold standard*. Cependant, on observe, pour chaque modèle, de légères différences de spécificité et de sensibilité.

On observe dans un premier temps, que notre meilleur modèle hors deep learning est le KNN et non la Random Forest. Cependant, pour le kNN comme le RF, la différence la plus importante viens du score de sensibilité. En effet, ces deux modèles sont plus sensibles que ceux présentés dans l'article mais moins spécifiques. Ainsi, nos modèles proposent plus de vrais

positifs mais prédisent plus souvent des faux positifs.

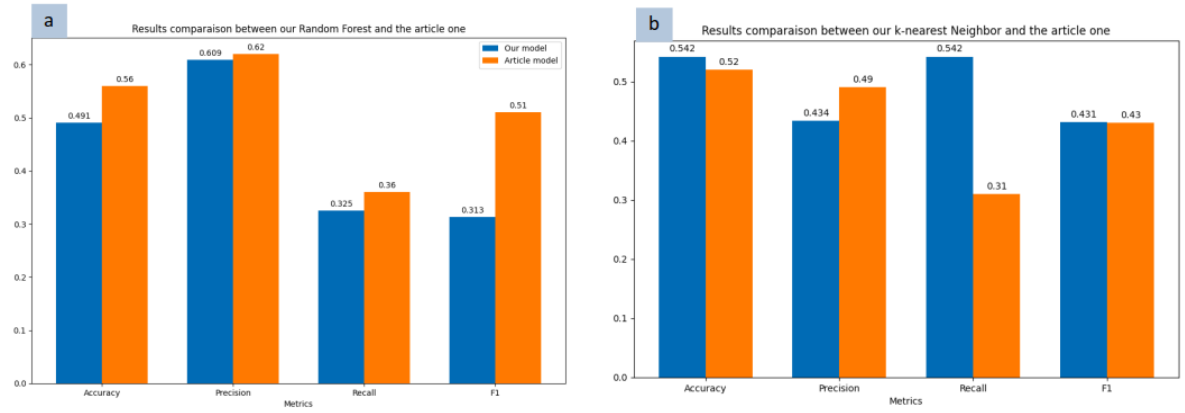

Figure 3 : Comparaisons de nos résultats au gold standard pour la Random Forest (a) et le k-Nearest Neighbors (b).

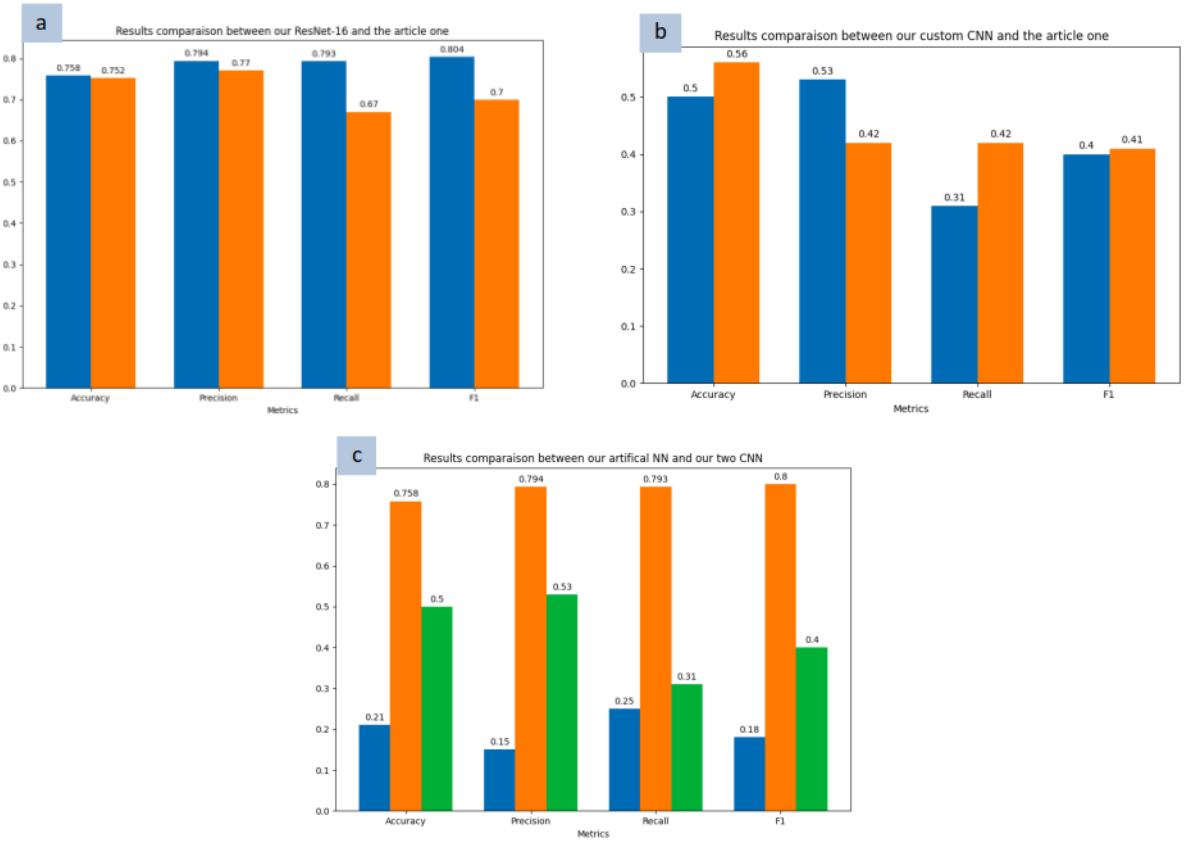

Figure 4 : Comparaisons de nos résultats au gold standard pour le ResNet (a) et le *custom* CNN (b). Comparaison entre notre ANN et les deux CNN (c) .

Pour les algorithmes de deep learning, nos résultats de précision sont les mêmes que ceux établis pas le *gold standard*. En revanche, le CNN personnalisé montre une plus grande spécificité que celui de l'article, mais est moins sensible. Contrairement aux algorithmes de machine

learning classiques, ce modèle va prédire moins de faux positifs mais aussi moins de vrais positifs. Concernant l'algorithme ResNet, nos résultats de précision sont identiques à ceux du *gold standard*, mais la spécificité et la sensibilité sont meilleures.

Conclusion

L'objectif de ce projet était de proposer une méthode adéquate pour la classification d'images de radiographies d'implants articulaires d'épaule. En se basant sur les travaux les plus récents qui évaluent des outils de machine learning, nous avons pu reproduire et valider les résultats faisant office de *gold standard*. Cependant, et en opposition avec l'article, le modèle de machine learning non *deep learning* le plus efficace est, dans notre cas, le kNN, et non le Random Forest. De plus, nos résultats de spécificité et sensibilité pour l'architecture ResNet-15 sont légèrement supérieurs à ceux de l'article. Ces différences indiquent que le meilleur modèle prédictif n'a peut-être pas encore été trouvé dans ce domaine d'application.

Dans le but d'optimiser un peu plus ces résultats, une base de données plus solide doit être construite. On remarque que les images actuelles, bien que semblant être en nombre suffisant, ne sont pas assez homogènes : les classes sont fortement déséquilibrées, le format des images n'est pas standardisé au niveau de : la taille, la luminosité, du zoom, de l'angle, etc... Ce manque de cohérence au sein du jeu de données reste l'élément le plus pénalisant lors du processus d'entraînement. En effet, on constate que les algorithmes les plus avancés (ResNet-15 ou autres types de réseaux convolutifs) apprennent rapidement mais stagne très vite. Au vu de ces résultats, il semblerait que la façon d'améliorer les prédictions soit l'optimisation de la qualité des données. Également, le test de plusieurs jeu de paramètres et d'autres types d'architectures de réseaux sont des points d'études futurs.

Bibliographie

[1] R H Cofield. Total shoulder arthroplasty with the neer prosthesis. *JBJS*, 66 :889–906, July 1984.

[2] GregorUrbana, SamanPorhemmata, MayaStark, BrianFeeley, KazunoriOkada, and Pierre-Baldi. Classifying shoulder implants in x-ray images using deep learning. *Computational and Structural Biotechnology Journal*, 18 :967–972, April 2020.

[3] Ayon Dey. Machine learning algorithms : A review. *(IJCSIT) International Journal of Computer Science and Information Technologies*, 7 :889–906, 2016.

[4] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *PROC OF THE IEEE*, November 1998.

[5] Page wikipédia de la couche *fully connected*. `https://en.wikipedia.org/wiki/Convolutional_neural_network#Fully_connected`. [Visité le 17-11-2020].

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR 2015*, April 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. December 2015.

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. January 2018.

[9] Page d'accueil de la base de données *ImageNet*. `http://www.image-net.org/`. [Visité le 1-11-2020].

[10] Base de données du mnist. `http://yann.lecun.com/exdb/mnist/`. [Visité le 17-11-2020].

[11] D.J. Hand and R.J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, pages 171–186, November 2001.

[12] CS. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. Understanding data augmentation for classification : When to warp ? *2016 International Conference on Digital Image Computing : Techniques and Applications (DICTA)*, pages 1–6, 2016.

[13] Page d'accueil du site web *Common US Shoulder Prostheses*. `https://faculty.washington.edu/alexbert/Shoulder/CommonUSShoulderProstheses.htm`. [Visité le 1-11-2020].