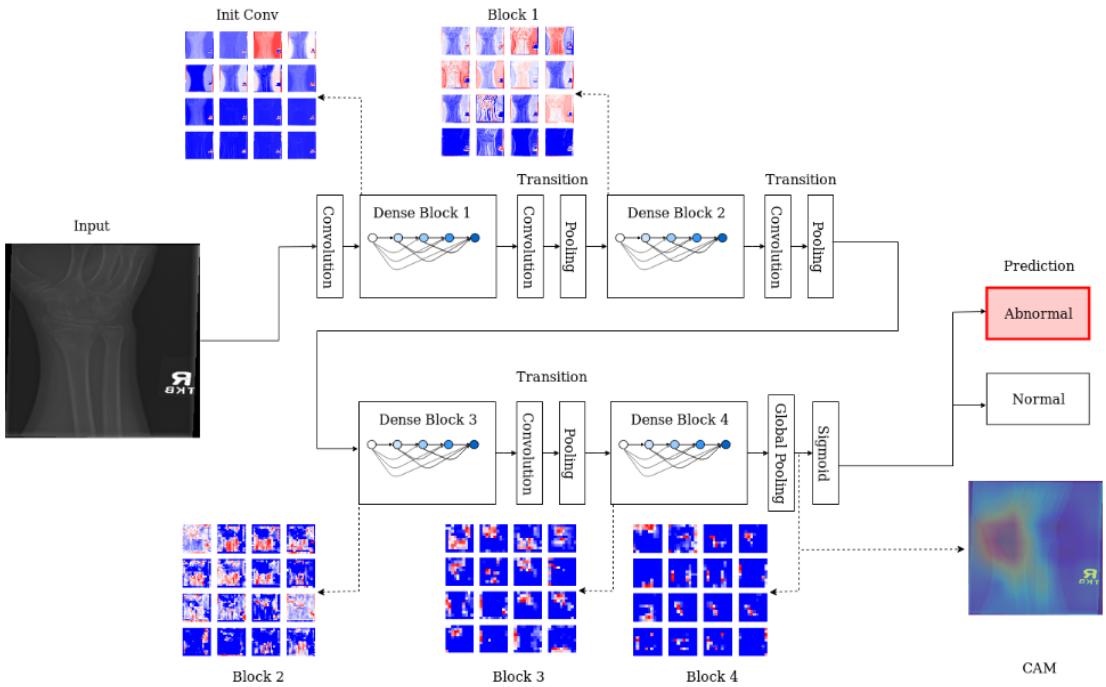
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
Université du Québec

SYS843-01
Rapport expérimental : Classification de Fractures Osseuses



Tiré de (Ananda et al., 2021)

Étudiant : Kevin Becquet - BECK85340001

Professeur : Marco Pedersoli

Table des matières

Mise en situation

Domaine d'application

Dans le monde, on recense 1.7 milliards de personnes atteintes de maladies musculo-squelettiques. Ces troubles causent toutes sortes de problèmes dont les fractures osseuses font partie. En effet, selon des rapports réalisés en France (International Osteoporosis Foundation, s.d.) et au Canada (Canada, 2020) portant sur la fragilité osseuse, nous reportons 380 000 fractures en France en 2017 et 130 000 au Canada en 2016.

Si ces fractures sont prises en charge tardivement, cela peut entraîner des complications de l'état du patient. La lésion peut s'aggraver ce qui peut augmenter la douleur ressentie par la personne blessée, créer des condition prématurée telle que l'arthrite voire causer, dans de plus rares circonstance, la perte de la mobilité du membre endommagé (Merck Manual, s.d.). Il est donc très important de détecter ces fractures au plus vite.

Cependant, ce diagnostic peut être long à réaliser, même pour un radiologue. C'est donc pour cela qu'il est intéressant de mettre en place des outils afin de rendre ce diagnostic plus rapide et de meilleure qualité. Ce projet est donc un projet de classification qui a pour but de déterminer la présence ou non d'une fracture sur une image de radiologie. Nous comparerons deux modèles couramment utilisés dans ce contexte.

Problématique abordée

Ce projet est un problème de classification est tout d'abord un problème de reconnaissance d'image. L'un des défis que nos modèles vont avoir à faire face sera la qualité des images. Les images de radiologies ne sont pas contraintes à un format uni : le fond des images n'est pas toujours de la même couleurs (généralement noir il peut aussi être blanc), et la qualité de ces dernières est également variable. En plus de cela, la forme et la taille des fractures est également différente selon chaque cas, même des radiologues confirmés ont parfois du mal à les identifier.

Enfin, il va nous falloir trouver un compromis entre temps de calcul nécessaire au modèle et précision des modèles. Un modèle qui prendra trop de temps à donner une réponse ne sera effectivement pas forcément efficace dans le cadre de son utilisation dans un centre hospitalier.

Objectif du projet

L'objectif du projet est donc de comparer les performances de différents modèles de classificateurs utilisés dans la littérature. Ces modèles seront des CNN car très performants dans la détection de caractéristiques présentes dans les images. Nous testerons les architectures de CNN suivantes : Inception-ResNet-v2, qui semble être l'une des architectures les plus efficaces pour ce type de problème ainsi que DenseNet-201. Les architectures DenseNet sont plutôt populaires pour ce type de problème et fournissent également des performances satisfaisantes (Ananda et al., 2021).

Structure du document

Dans ce document nous commencerons par décrire les deux modèles que nous allons comparer en détaillant brièvement leur fonctionnement à l'aide d'un schéma bloc et d'une description de leur algorithme. Par la suite, il sera présenté la méthodologie utilisée lors des expérimentations comprenant la base de données sur laquelle nous avons travaillée, le protocole expérimental et les métriques de comparaison. Enfin, seront mis en avant les résultats de ces expérimentations puis une interprétation de ces derniers. Il sera ajouté en annexe plus d'explication sur le code utilisé lors de ce projet ainsi que comment y accéder et des informations sur les articles principaux sur lesquels se basent ce projet.

Techniques utilisées

Dans cette section seront présentés les deux modèles utilisés dans ce projet. Bien que différents ils partagent quelques similarités tel que le fait que ce soit tous deux des CNN, très efficaces sur ce type de problématique. De plus, tous deux vont utiliser l'optimiseur Adam au cours de l'entraînement.

Inception Resnet v2

Ce modèle est une architecture CNN. Il reprend à la fois les concepts d'Inception, en créant des blocs contenant plusieurs plus petits filtres dans l'optique de s'étendre en largeur, et de ResNet en ajoutant des connexions résiduelles dans les blocs Inception-ResNet pour favoriser la transmission d'information au travers du réseau.

Voici le schéma principal du modèle utilisé :

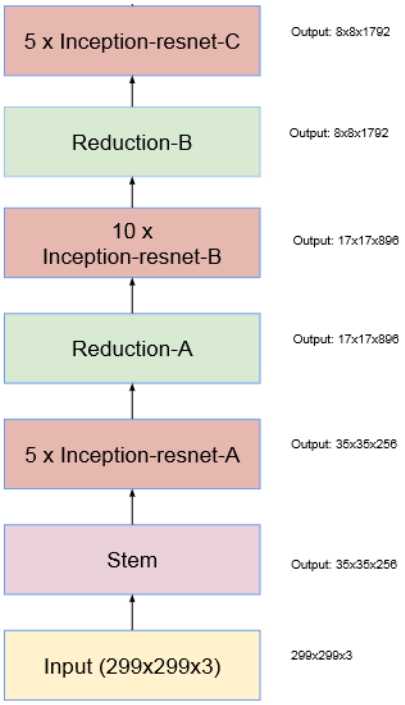


Figure 1 - Schéma du modèle Inception-ResNet-v2 tiré de (Szegedy, Ioffe, Vanhoucke, & Alemi, 2016)

Après le dernier bloc Inception-ResNet-C, on y ajoutera d'une couche Flatten et Fully connected à la fin du modèle. En effet, à ces dernières ont été ajoutées afin de pouvoir réaliser la classification. Comme le problème auquel nous faisons face est un problème binaire, cette dernière couche aura une unique sortie activée par la fonction sigmoïde.

Un bloc Inception-ResNet se caractérise par son étalement en largeur grâce à l'utilisation de plusieurs filtres sur la même couche ce qui va donc permettre d'extraire plus de caractéristiques par couche. Voici les blocs Inception-ResNet utilisés dans le la figure 1 :

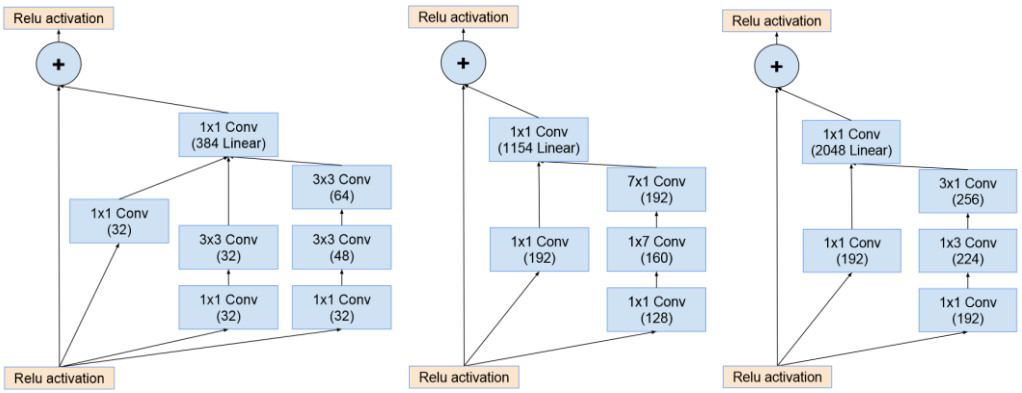


Figure 2 - Blocs Inception-ResNet A B et C (de gauche à droite) tiré de (Szegedy et al., 2016)

DenseNet-201

Ce modèle est également un CNN. C'est une variante de l'architecture DenseNet contenant de 201 couches. Cette architecture est composée de plusieurs blocs DenseNet intercalés de couches de pooling.

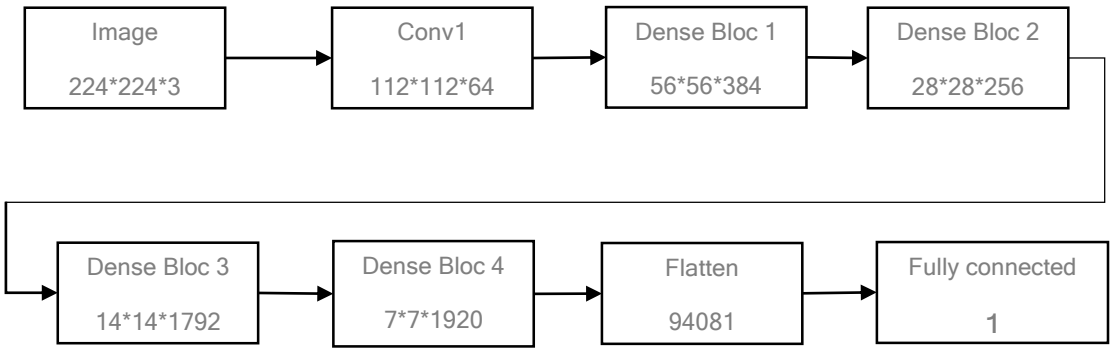


Figure 3 - Schéma du DenseNet-201 utilisé avec la taille des sorties de chaque bloc

Ces blocs denses ont la particularité de concaténer la sortie de chacune des couches convolutives avec son entrée si bien que chacune des couches suivantes va être connectée à toutes les entrées des couches précédentes. Cela a pour avantage de réduire le phénomène de disparition du gradient lors de l'entraînement du réseau.

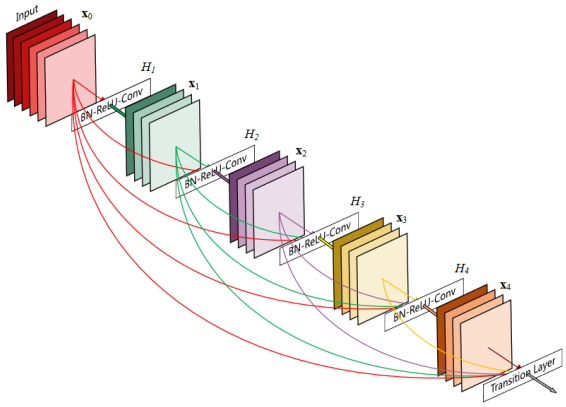


Figure 4 - Schématisation d'un bloc Dense tiré de (Huang, Liu, van der Maaten, & Weinberger, 2018)

Grâce à cette méthode, le réseau de neurones peut se permettre de s'étendre en profondeur et ainsi permet l'extraction de caractéristiques plus complexes.

Nous pouvons remarquer l'utilisation d'une couche Flatten et Fully connected à la fin du modèle. En effet, à ces dernières ont été ajoutées afin de pouvoir réaliser la classification. Comme le problème auquel nous faisons face est un problème binaire, cette dernière couche aura une unique sortie activée par la fonction sigmoïde. L'ensemble du modèle dénombre un total d'environ 18.4 millions de paramètres et, une fois enregistré, prend une place sur le disque de 211 Mo.

Ainsi, la principale différence entre les deux modèles va être que le DenseNet va plus s'étendre en profondeur tandis que l'Inception ResNet va prendre plus de place en largeur. Ces deux solutions sont intéressantes car elles approchent le problème de deux optiques différente. Le DenseNet, dû à sa profondeur, va permettre une extraction de caractéristiques plus complexes présentes dans l'image.

Méthodologie

Base de données MURA

Nous allons, dans ce projet, classifier des radiographies réalisées sur des patients ayant subi, ou non, une fracture osseuse.

La base de donnée choisie pour cela se nomme MURA (Musculoskeletal Radiographs). Elle a été créée par l'université de Stanford dans le cadre d'un concours, clôt en 2018, ayant pour but d'évaluer différents modèles de détection et de classification de fractures.

Cette base de données regroupe un total de quasiment 15 000 cas labellisés. Chacun de ces cas représente un regroupement de plusieurs images présentant l'os analysé sous plusieurs angles. MURA contient ainsi un total de 40 005 images.

Study	Train		Validation		Total
	Normal	Abnormal	Normal	Abnormal	
Elbow	1094	660	92	66	1912
Finger	1280	655	92	83	2110
Hand	1497	521	101	66	2185
Humerus	321	271	68	67	727
Forearm	590	287	69	64	1010
Shoulder	1364	1457	99	95	3015
Wrist	2134	1326	140	97	3697
Total No. of Studies	8280	5177	661	538	14656

Figure 5 - Détail du contenu de la base de données MURA tiré de (Rajpurkar et al., 2018)

Dû à très son grand nombre de cas, cette base de données est populaire parmi les articles traitant ce problème de classification ainsi que le problème de détection lié à la problématique.

Cependant, la séparation entre entrainement et validation initialement présente dans cette base de données était destinée au concours de l'université de Stanford. Les modèles soumis étaient testés sur une base de test non publiée.

Par conséquent, les bases d'entraînement, validation et test utilisées dans le cadre de ce projet sont séparées sur l'ensemble des images.

Protocole expérimental

Ce projet est tout d'abord un projet dans lequel nous voulons comparer deux modèles de CNN différents pour effectuer la tâche de classification. Nous utiliserons la base de données présentée dans la section précédente séparée en bases d'entraînement, validation et test correspondant respectivement à 80%, 10% et 10% du nombre d'images utilisées.

L'entraînement des modèles pouvant être long, ces derniers ont été entraînés sur Google Colab pour ainsi profiter de l'accès à un GPU et accélérer le processus.

Cependant, dû au fait que l'accès à ces GPU est plutôt limité en temps, l'entraînement sur la totalité de la base de données est compromis. Il n'a pas été Par conséquent plusieurs entraînement ont été réalisés sur différents sets plus restreints d'images afin de pouvoir réaliser un entrainement pertinent et de constater si le changement dans ce nombre d'images de référence a un effet significatif sur les performances des modèles.

Ainsi, la même expérience a été réalisé sur nos deux modèles en changeant le nombre d'images utilisées. Après avoir sélectionné 100 puis 300 puis 10 000 images de la base MURA, l'échantillon pris a été séparé en bases d'entrainement, validation et test dans les proportions définies précédemment. Nos deux modèles sont tous deux pré-

entrainés dur le dataset ImageNet puis ont été entraînés sur 10 epochs dans un premier temps afin de voir au bout de combien d'epochs les modèles commencent à sur-apprendre. Par la suite, on réalisera l'expérience sur un total de 10 000 images avec un entraînement en utilisant un nombre d'epochs le plus optimal possible pour chacun des deux modèles évalués.

Métriques utilisées

Le but de ce projet est d'évaluer ces modèles pour trouver celui qui est le plus adapté dans le cadre de leur utilisation en tant qu'outils d'aide au diagnostic par un radiologue. Ainsi, les métriques qui seront utilisées en guise de comparaison sont la **précision** des modèles sur une base de test ainsi que leur **temps d'exécution** sur les bases d'entraînement et de test. Nous comparerons les résultats obtenus en terme de précision avec ceux des travaux de (Ananda et al., 2021) majoritairement. Ces derniers présentent une comparaison de plusieurs modèles entraînés sur la base de données MURA mais n'ont pas été pré-entrainés. Il pourra donc être intéressant de comparer les performances de nos modèles et de voir ce que ce pré-entraînement apporte dans le cadre de notre problème.

Résultats

Présentation des résultats

Dans cette section nous présenterons l'ensemble des expériences qui ont été réalisées et comparerons les modèles dans chacune d'entre elle. Nous présenterons les courbes résultant de l'entrainement de nos modèles puis les résultats de ces derniers sur la base de test selon les métriques mises en avant précédemment.

100 images

Cette expérience s'est déroulée sur 80 images pour la base d'entraînement et 10 pour la validation.

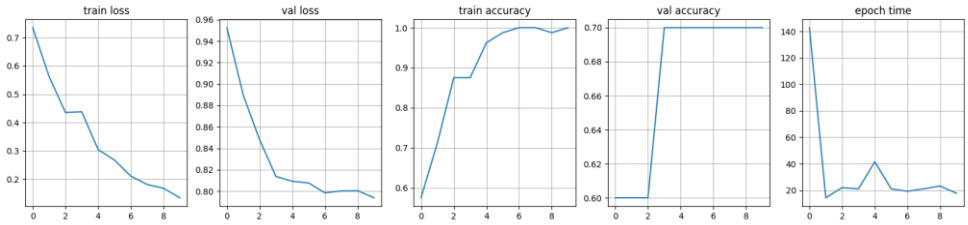


Figure 6 - Courbes d'entraînement de DenseNet-201

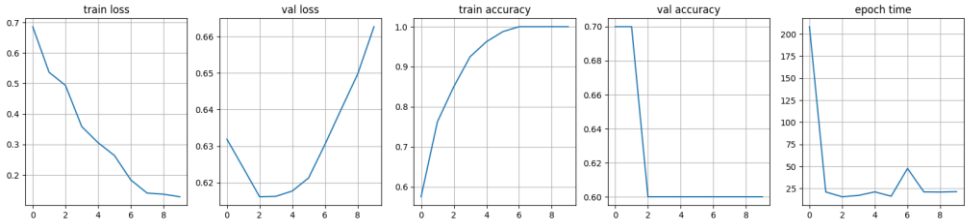


Figure 7 - Courbes d'entraînement d'Inception-ResNet-v2

On remarque que le modèle Inception-ResNet-v2 commence à sur-apprendre à partir de la 3ème epoch. En effet, sa fonction d'erreur sur la base de validation commence à augmenter alors que celle de la base d'entraînement continue de diminuer. Cela a effectivement affecté les résultats du modèle comme nous pouvons voir sa précision sur la base de validation diminuer à partir de ce temps-ci.

En ce qui concerne les résultats sur la base de test, nous avons pu obtenir les résultats suivant.

	Temps d'exécution	Précision
DenseNet-201	0.5s	70%
Inception-ResNet-v2	180ms	60%

Tableau 1 - Résultats de l'expérience sur 100 images

3 000 images

Cette itération nous donne les courbes d'apprentissages suivantes :

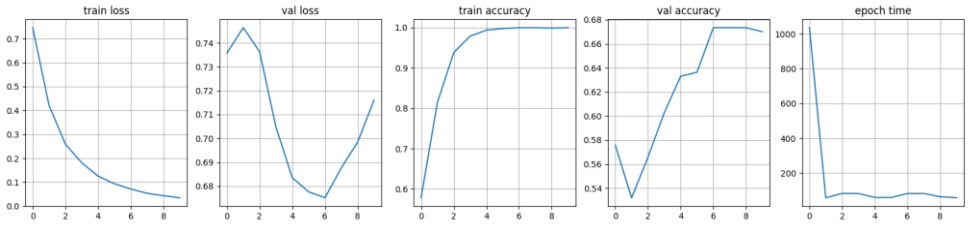


Figure 8 - Courbes d'entraînement de DenseNet-201

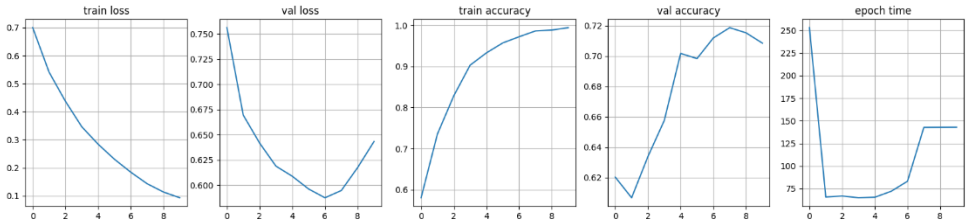


Figure 9 - Courbes d'entraînement d'Inception-ResNet-v2

Dans un premier temps on remarque que les deux modèles sont maintenant en sur-apprentissage après ces 10 epochs. Ainsi, il ne sera pas nécessaire de réaliser un entraînement sur autant d'epochs lors de la prochaine expérience. Cela nous arrange car l'expérience prendra moins de temps et aura donc plus de chance d'aboutir avant que Google Colab ne déconnecte l'accès au GPU.

Pour ce qui est des résultats sur la base de test, on se retrouve avec :

	Temps d'exécution	Précision
DenseNet-201	91s	68%
Inception-ResNet-v2	3s	60%

Tableau 2 - Résultats de l'expérience sur 3 000 images

Ici, on commence le modèle DenseNet-201 est plus précis que son concurrent. Cependant ce dernier est bien plus rapide, il prend 30 fois moins de temps pour traiter la même base de test.

10 000 images

Lors de cette expérience finale, nous avons pu régler le nombre d'epochs nécessaires à l'apprentissage grâce aux expériences précédentes. Il a été décidé de réaliser l'entraînement de DenseNet-201 sur 5 epochs tandis que l'entrainement d'Inception-ResNet-v2 s'est fait sur 4 epochs. Ces chiffres ont été trouvés du fait du sur apprentissage observé sur les itérations précédentes et de la limitation temporelle de Google Colab.

Commençons par présenter les courbes d'apprentissage de cette expérience :

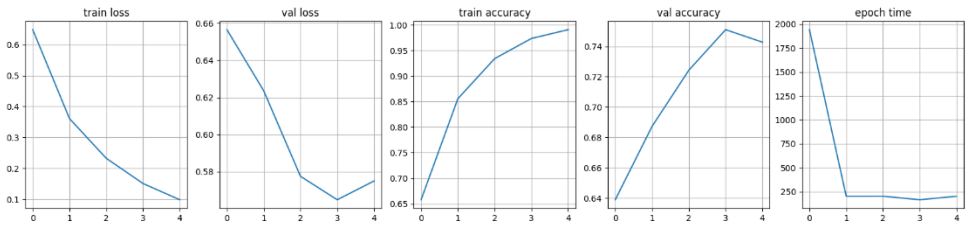


Figure 10 - Courbes d'entraînement de DenseNet-201

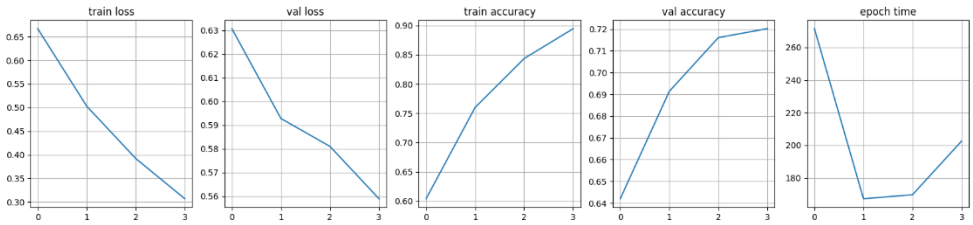


Figure 11 - Courbes d'entraînement d'Inception-ResNet-v2

Nous voyons ici que le modèle DenseNet est à son niveau optimal à l'epoch 3. C'est donc ce modèle, enregistré via callback, qui sera utilisé pour la comparaison sur la base de test. Pour ce qui est d'Inception-ResNet, on utilisera le modèle tel qu'il est à la fin de l'entraînement comme il n'a pas atteint son point de sur-apprentissage.

Ainsi, les résultats pour cette expérience sur la base de test sont :

	Temps d'exécution	Précision
DenseNet-201	242s	81%
Inception-ResNet-v2	10s	71%

Tableau 3 - Résultats de l'expérience sur 10 000 images

Ici, DenseNet est significativement plus précis que son concurrent avec 10% de précision en plus. Cependant, Inception-ResNet reste bien plus rapide à produire un résultat.

Ainsi, sur ces trois expériences, on remarque une augmentation de la précision des modèles suivant l'augmentation du nombre d'images utilisées. Le modèle DenseNet, particulièrement, voit sa précision augmenter plus significativement que son concurrent. Sur le plan de la complexité temporelle, on observe également une augmentation en fonction du nombre d'images mais cette fois ci, c'est le modèle Inception-ResNet qui se trouve plus performant.

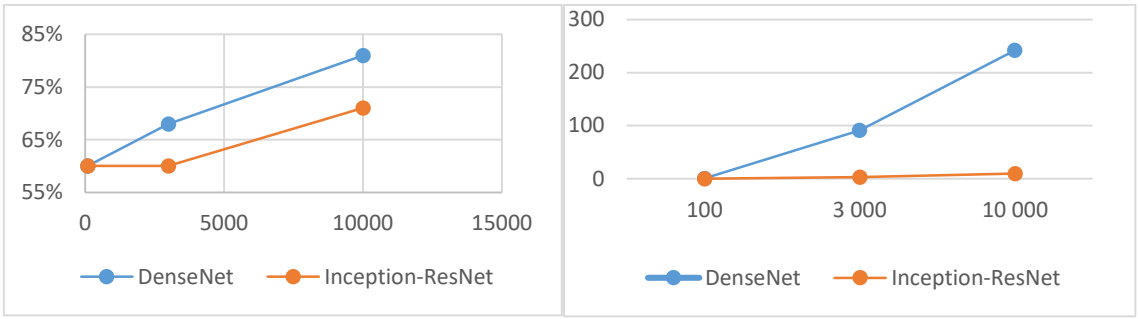


Figure 12 - Comparaison des précision et temps d'exécution des modèles sur les bases de test en fonction du nombre d'images utilisées

Interprétation des résultats

Dans un premier temps, lorsque l'on regarde nos résultats seuls, on peut en conclure deux choses. Premièrement, le modèle Inception-ResNet-v2 est sensiblement plus rapide à réaliser son pronostic. En effet, il peut prendre jusqu'à 30 fois moins de temps pour un même nombre de radiographies à analyser. Cela était attendu. En effet, le modèles Inception-ResNet fait appel aux principes de ResNet pour faciliter la propagation du gradient au travers le réseau de neurones à l'aide des connexions résiduelles entre les différentes couches du réseau. Si bien que la rétro-propagation lors de l'apprentissage est accélérée. Le modèle DenseNet, n'utilisant pas cette méthode va donc avoir besoin de plus de temps afin de fournir un résultat.

En contrepartie de cette vitesse, Inception-ResNet-v2 se trouve moins précis que DenseNet-201. Et cela se voit de plus en plus quand le nombre d'images utilisé durant l'expérience augmente. On passe en effet d'une précision plutôt similaire sur la base de test lorsque l'on utilise 100 images à 10% de différence quand on en utilise 10 000. Cependant, sur l'itération utilisant 10 000 images, il n'est pas sûr que le modèle Inception-ResNet soit totalement optimisé. On a effectivement remarqué que sa courbe d'erreur sur la validation n'a pas forcément atteint son minimum. Il est donc

possible que la précision de ce modèle soit meilleure avec quelques epochs d'entraînement supplémentaires.

Au vu de la progression des résultats par rapport au nombre d'images utilisées lors des expériences, on peut s'attendre à ce que ces tendances s'accentuent si l'on réalise l'expérience sur l'ensemble des 40 000 images de la base de données.

De plus, nous avons pu remarquer lors de nos expériences que le modèle DenseNet est bien plus léger que le Inception-ResNet. En effet, après leur enregistrement dans un fichier .h5, on constate que DenseNet pèse 211 Mo tandis que Inception-ResNet pèse 624 Mo. Cela peut également être un argument de décision en faveur d'un modèle ou de l'autre.

Enfin, il va être possible de comparer ces résultats avec ceux de la littérature. La précision du modèle DenseNet-201 établie dans (Ananda et al., 2021) est de 69.5%. Ainsi, nous avons été capable d'avoir une meilleure précision sur ce modèle. Cela peut être expliqué par l'utilisation d'un modèle pré-entraîné sur la base de données ImageNet qui pourrait améliorer considérablement les performances de notre système.

La même étude nous fournis également une idée de la précision d'Inception-ResNet-v2 qu'il est possible d'atteindre. Celle-ci, dans l'étude, est de 74.7%. Nous avons donc une précision légèrement moins bonne lors de nos tests qui est explicable par un entraînement sur un plus petit nombre d'images. Au vu de la progression de la précision des modèles par rapport au nombre d'images, on pourrait s'attendre à ce que ce modèle soit, lui aussi, plus performant que dans présenté dans (Ananda et al., 2021) si son entraînement se faisait sur la totalité de la base de donnée.

Cependant il est possible d'exprimer quelques critiques en ce qui concerne les limites des expériences réalisées lors de ce projet. Comme il n'a pas été possible d'entrainer

nos modèles sur la totalité de la base de données, il aurait pu être plus intéressant de séparer totalement une base de test plus grande contenant un nombre d'images indépendant du reste de l'expérience. En effet, la séparation en pourcentage des bases de validation et de test fonctionne bien moins à petite échelle, les résultats obtenus peuvent avoir une forte variance dû au fait que la sortie d'un modèle sur une seule image peut représenter une différence de plusieurs pourcents de précision sur le résultat final. Ainsi, les résultats des modèles sur de très petites quantité d'images deviennent moins probant du fait de la variabilité du résultat sur la base de test.

Conclusion

Pour conclure sur cette comparaison, nous avons suivant la caractéristique que l'on regarde un modèle qui surpasse l'autre. En ce qui concerne le temps de calcul nécessaire pour avoir un résultat, Inception-ResNet gagne étant entre 20 et 30 fois plus rapide que DenseNet-201. Cependant, DenseNet-201 est meilleur que son concurrent en ce qui concerne la justesse du résultat avec jusqu'à 10% de précision en plus qu'Inception-ResNet-v2.

Ces résultats ne nous permettent pas de choisir un modèle meilleur que l'autre en toutes circonstances mais nous pourrions préférer l'un à l'autre en fonction de la situation. Par exemple, on pourrait voir ces deux modèles implémentés simultanément dans un même hôpital. Un service d'urgence trouverait plus d'avantage à utiliser Inception-ResNet-v2 car sa capacité à donner des résultats rapidement permettrait de réaliser un premier diagnostic de patient afin d'accélérer sa reconduite vers le service les plus adapté à sa condition tandis que, de l'autre côté, un service de radiologie apprécierait plus la justesse du DensNet-201 pour seconder le radiologue afin de classifier des fractures plus compliquées à détecter.

Il serait cependant intéressant d'ajouter à ce projet des méthodes de détection de fractures. En effet, même s'il est intéressant de signaler la présence d'une fracture dans une radiographie, pouvoir la localiser est aussi une tâche cruciale à sa prise en charge. Il était initialement prévu de mettre en place, au cours de ce projet, une méthode de Class Activation Map pour permettre de comprendre quelle zone des images analysées par nos modèles ont permis leur classification en tant qu'image contenant une fracture ou non. Cela n'a finalement pas été inclut dans le rapport car suite à un manque de temps aucune expérience n'a pour l'instant pu être réalisée.

Références

Ananda, A., Ngan, K. H., Karabağ, C., Ter-Sarkisov, A., Alonso, E., & Reyes-Aldasoro, C. C. (2021). Classification and Visualisation of Normal and Abnormal Radiographs; A Comparison between Eleven Convolutional Neural Network Architectures. *Sensors*, *21*(16), 5381. https://doi.org/10.3390/s21165381

Canada, A. de la santé publique du. (2020, 27 novembre). Rapport du Système canadien de surveillance des maladies chroniques : L'ostéoporose et les fractures connexes au Canada, 2020. [recherche]. Repéré à https://www.canada.ca/fr/sante-publique/services/publications/maladies-et-affections/osteoporose-fractures-connexes-2020.html

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018, 28 janvier). Densely Connected Convolutional Networks. arXiv. Repéré à http://arxiv.org/abs/1608.06993

International Osteoporosis Foundation. (s.d.). Latest projects - Broken Bones, Broken Lives | International Osteoporosis Foundation. Repéré à https://www.osteoporosis.foundation/what-we-do/science-and-research/latest-projects--broken-bones-broken-lives

Kingma, D. P., & Ba, J. (2017, 29 janvier). Adam: A Method for Stochastic Optimization. arXiv. Repéré à http://arxiv.org/abs/1412.6980

Merck Manual. (s.d.). Présentation des fractures - Lésions et intoxications. *Manuels MSD pour le grand public*. Repéré à https://www.merckmanuals.com/fr-ca/accueil/l%C3%A9sions-et-intoxications/fractures/pr%C3%A9sentation-des-fractures

Rajpurkar, P., Irvin, J., Bagul, A., Ding, D., Duan, T., Mehta, H., … Ng, A. Y. (2018, 22 mai). MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs. arXiv. Repéré à http://arxiv.org/abs/1712.06957

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016, 23 août). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv. Repéré à http://arxiv.org/abs/1602.07261

Annexe

Code implémenté

Ce projet a été codé en Python dans l'environnement de Google Colab. Le code ainsi que les résultats des expérimentations se trouve dans le dossier Google Drive suivant :

https://drive.google.com/drive/folders/1-gDDLCOukGwPkkserlfdynuYJaAA3gp1?usp=sharing

Le code principal, « Projet.ipynb » a été organisé de la façon suivante :

- Acquisition des données et création des bases de training, validation et test
- Entraînement et test de DenseNet-201
- Entraînement et test de Inception-ResNet-v2

Pour la première partie le challenge a été de créer des bases de données utilisables par les fonctions de keras à partir d'un dossier d'images pas forcément fait pour faciliter son usage : on a à notre disposition un fichier .csv contenant l'ensemble des chemin d'accès vers les images. Ainsi, pour créer nos bases de données d'entraînement, de validation et de test, nous avons dû commencer par créer des Dataframe avec les chemins d'accès des images à partir de Pandas. Ces derniers sont tirés de manière aléatoire à partir du fichier .csv et placés dans lesdits Dataframes associés au label de l'image. Une fois ces Dataframes créés, on peut, par la suite, créer les datasets associant les images et leur label à l'aide de la méthode *flow_from_dataframe()* de keras.

Ensuite, les deux parties d'entrainement sont similaires excepté le modèle. Dans les deux cas on commence par charger le modèle à partir de la bibliothèque keras pui y ajoutons les couches Flatten et entièrement connectée afin d'avoir notre résultat de classification en sortie. Par la suite on entraîne le modèle à partir d'une boucle for sur le nombre d'epoch souhaitées. A chaque itération de la boucle on récupère les valeurs

de précision et d'erreurs obtenues sur les bases de validation et d'entraînement ainsi que le temps qu'il a fallu pour réaliser l'epoch d'entraînement. On affiche ces différentes valeurs sur des graphes comme présentés dans la section Résultats de ce document. Ces graphes ainsi que les modèles sont enregistrés dans le dossier graphe à la fin de l'entraînement mais également à chaque epoch grâce à l'utilisation de callbacks.

Enfin, les modèles sont évalués sur la base d'entraînement avec la méthode *evaluate()* de keras nous permettant d'obtenir la précision et le temps de calcul sur la base indiquée.

Cependant, pour critiquer ce travail, il aurait pu être intéressant de fixer chaque image à un sous-ensemble training, valid ou test afin de s'assurer du fait qu'un modèle déjà entraîné et sauvegardé, dans le cas où on compte les réutiliser après la déconnexion du GPU de Google Colab, ne voit pas d'images utilisées lors de l'apprentissage durant le test. Mais comme le test est réalisé juste après la phase d'entraînement, ce problème n'est pas apparu lors des expérimentations.

En plus de cela, l'implémentation de la méthode grad-CAM, une ouverture à ce projet* que je comptais mettre en place, a été commencée mais non terminée pour l'instant dû à une difficulté à extraire la dernière couche de convolution d'un modèle provenant d'un fichier préalablement enregistré. D'où son absence dans ce rapport. Je compte tout de même réaliser cette étape, qui me semble vraiment intéressante dans l'optique de rendre l'utilisation de ces modèles pertinente dans le cadre du problème initial de classification de fractures.

Présentation des articles

(Ananda et al., 2021)

Ce projet se base principalement sur les résultats des travaux réalisés par (Ananda et al., 2021). Dans cet article est réalisé une comparaison de la précision de 11 modèles utilisant 3 optimiseurs différents entraîné sur la base de données MURA. Cet article conclut que le modèle Inception-ResNet-v2 obtient les meilleurs résultats sur la moyenne de la précision obtenue à partir des 3 optimiseurs avec une précision de 72.3% et que DenseNet-201 se trouve juste en dessous avec une moyenne de 71.7% de précision.

En plus de cela, l'article présente l'implémentation d'une Class Activation Map réalisée sur le modèle Inception-ResNet-v2 entraînée sur la même base de données ayant, cette fois-ci, subi une augmentation en appliquant des transformations aux images.

(Szegedy et al., 2016)

Cet article présente les modèles Inception-ResNet-v2 ainsi que Inception v3 et v4. Il y détaille leur architecture ainsi que leurs différences. Il réalise par la suite une comparaison de l'erreur de ces modèles sur la base de validation de ImageNet en fonction du nombre d'epochs de l'apprentissage. On y conclut que les taux d'erreur d'Inception-ResNet-v2 sont inférieurs aux autre modèles.

(Huang et al., 2018)

Cet article présente le concept d'architecture DenseNet et introduit plus précisément DenseNet 121, 169, 201 et 264. On y trouve une explication du fonctionnement d'un bloc Dense ainsi que le schéma général d'un réseau utilisant cette architecture. Après

cette explication, on retrouve des comparaisons de l'erreur de ces modèles avec des modèles ResNet, autre concept voulant réaliser la même chose que DenseNet : faciliter l'expansion en profondeur des modèles. On y conclut l'un des avantages principaux de DenseNet par rapport à ResNet est que pour la même tâche, DenseNet permet une réduction du nombre de paramètres étant aussi performant avec 3 fois moins de paramètres.