CLASSIFICATION DES TUMEURS RÉNALES À L'AIDE D'IMAGERIE PAR RÉSONANCE MAGNÉTIQUE DANS UN CONTEXTE D'APPRENTISSAGE MULTITÂCHES

par

Alexandre Ayotte

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 24 mai 2023

Le 24 mai 2023

Le jury a accepté le mémoire de Alexandre Ayotte dans sa version finale

Membres du jury

Professeur Martin Vallières
Directeur
Département Informatique

Julien Cohen-Adad
Professeur agrégé
Membre externe
Département Génie électrique
École Polytechnique de Montréal

Professeur Maxime Descoteaux
Président-rapporteur
Département Informatique

Sommaire

Pour les patients présentant une lésion au rein, la connaissance de la malignité, ainsi que du sous-type et du grade (dans le cas où la tumeur serait maligne), sont des éléments essentiels pour le pronostic. Actuellement, la procédure standard pour obtenir ces informations est de procéder à une biopsie de la lésion. La biopsie étant très invasive et manquant de fiabilité pour les tumeurs hétérogènes, l'analyse d'imagerie non invasive telle que l'imagerie par résonance magnétique (IRM) à l'aide d'intelligence artificielle est une alternative qui a suscité beaucoup d'intérêt au cours des dernières années. Bien que les récentes études montrent que les réseaux de neurones à convolution soient très prometteurs lorsqu'il s'agit de classifier les tumeurs rénales par rapport à leur malignité, leur sous-type ou leur grade, les résultats ne sont pas encore suffisamment convaincants pour constituer une solution de rechange à la biopsie.

Dans ce mémoire, nous vérifierons s'il est possible, à l'aide de l'apprentissage multitâche, d'améliorer les performances d'un modèle d'apprentissage profond pour la classification de la malignité, du sous-type et du grade des tumeurs rénales. Nous utiliserons un jeu de données comportant les images par résonance magnétique de pondération T1 avec contraste amélioré et T2 de 1082 patients qui présente une lésion à l'un des reins, la segmentation de cette lésion pour chaque modalité et un ensemble de données cliniques. Nous évaluerons la pertinence d'un modèle d'apprentissage à tâche unique utilisant l'imagerie par rapport à un modèle se basant sur les données cliniques. Par la suite, nous allons comparer les réseaux de neurones à convolution entraînés sur les tâches individuellement à des modèles entraînés à prédire les 3 caractéristiques simultanément. Pour finir, nous déterminerons s'il est bénéfique de forcer un modèle à simultanément classifier la tumeur et prédire un ensemble de caractéristiques radiomiques qui joueront le rôle de tâches auxiliaires.

Sommaire

Mots-clés: Imagerie médicale ; Apprentissage profond ; IRM ; Classification de tumeur rénale ; Radiomique ; Apprentissage multitâche.

Remerciements

Je tiens à prendre quelques lignes pour, dans un premier temps remercier, sincèrement mon directeur de recherche Martin Vallières qui a fait preuve de beaucoup de patience à mon égard et qui a toujours pris le temps de me rencontrer chaque semaine durant les deux premières années de ma maitrise, même en temps de pandémie et après l'arrivée de son premier enfant.

Je remercie aussi Harrison X. Bai de l'université de Brown pour nous avoir fournis l'accès à un jeu de données qui nous a permi de mener nos expériences.

Je tiens ensuite à remercier mes collègues de travail : Nicolas Raymond, Simon Giard-Leroux, Olivier Lefebvre, Mahdi Ait Lhaj Loutfi et Hakima Laribi, qui m'ont soutenu à travers les diverses étapes de ma recherche. Je dois tout autant une mention spéciale à deux de nos anciens stagiaires, soit Guillaume Cléroux et Charles Lévesque-Matte, qui ont participé de manière mémorable à l'ambiance de notre laboratoire.

Je voudrais aussi exprimer ma reconnaissance à mes soeurs Vicky et Katérie, qui m'ont supporté durant toute la durée de ma maitrise.

Enfin, je dois un grand merci à Sofia et Tangente qui m'ont servis de modèle pour quelques figures présentes dans ce mémoire.

Table des matières

Liste des figures

Liste des tableaux

LISTE DES TABLEAUX

Liste des algorithmes

Abréviations

AE Amélioration attendue

AMT Apprentissage multitâches

BCI Borne de confiance inférieur

CCR Carcinome des cellules rénales

CFR Caractéristique de fonctionnement du récepteur

CCM Coefficient de corrélation de Matthews

GTA Groupement de tâche par affinité

IRM Imagerie par résonance magnétique

LTB Learning-To-Branch

RAM Réseaux d'attention multitâche

MVS Machine à vecteur de support

PF Partage forcé

PI Partage incité

PCPA Partage de caractéristiques pleinement adaptatif

PMA Probabilité maximale d'amélioration

PostAct Post-activation

PreAct Pré-activation

ResNet Réseau résiduel

RF Radiofréquence

RI Région d'intérêt

Abréviations

RNC Réseau de neurones à convolution

SSC Surface sous la courbe

RATU Réseau d'attention à tâche unique

TC Tomographie computationnelle

TPE Arbre d'estimateur de Parzen

T1C Pondération T1 rehaussée par un contraste

T2 Pondération T2

Introduction

Pour l'année 2022, il est estimé qu'environ 79 000 nouveaux cas de cancer du rein seront diagnostiqués et environ 13 920 personnes décéderont aux États-Unis[1]. Pour l'année 2021 au Canada, l'estimation est de 7 800 nouveaux cas et de 1 950 morts[2]. Lorsqu'une tumeur rénale est diagnostiquée comme étant maligne, l'opération la plus courante consiste à retirer la masse rénale cancéreuse. Toutefois, les tumeurs rénales de petite taille étant difficiles à classifier à l'aide de l'imagerie seulement, cette information est parfois disponible seulement après la chirurgie [115]. Ainsi, environ 20% des masses rénales retirées sont en fait bénignes [76]. Cette opération coûteuse pourrait donc être évitée pour ainsi épargner au patient une période de convalescence. D'autre part, des informations supplémentaires sur la tumeur telles que le sous-type [90, 117, 120] et le grade [36, 37] sont importantes pour le pronostic. En effet, alors que les sous-types cellules claires, papillaires et chromophobes représentent plus de 90% [69, 120] des cas de carcinome des cellules rénales (CCR), qui est lui-même le type de cancer du rein le plus fréquent [157], leur distinction est importante dans la prédiction du taux de survie du patient [2, 8, 27, 89].

Bien que la biopsie soit encore la méthode la plus utilisée pour quantifier le grade d'une tumeur rénale, cette opération invasive, en plus comporter certains risques d'hémorragie, d'infection ou de rupture de la tumeur [20], est parfois sujette à des erreurs d'échantillonnage et de faux positif [108]. Avec la popularité et l'attention croissante pour la médecine personnalisée, les dernières avancées en imagerie médicale, la puissance de calcul grandissante et l'accessibilité des données depuis le début du siècle, les

1. Key Statistics About Kidney Cancer, consulté le 15 Janvier 2022 : https ://www.cancer.org/cancer/kidney-cancer/about/key-statistics.html

2. Statistiques sur le cancer du rein, consulté le 15 janvier 2022 : https ://cancer.ca/fr/cancer-information/cancer-types/kidney/statistics

méthodes d'apprentissage machine basées sur des modalités d'imagerie 3D, telles que les images par tomographie computationnelle (TC) et l'imagerie par résonance magnétique (IRM), sont devenues plus présentes dans les dernières années en recherche médicale [84, 13]. Dans un premier temps, un nouveau champ de recherche portant sur l'extraction et l'usage de caractéristiques d'images médicales prédéfinies, les caractéristiques radiomiques [87], a vu le jour il y a une dizaine d'années. Depuis, leur potentiel pour les problèmes liés au cancer du sein [134], des poumons [34, 51, 110], des reins [72, 102, 149] et autres [65] a largement été étudié.

En parallèle, l'intérêt sans précédent de l'apprentissage profond et plus particulièrement des réseaux de neurones à convolution (RCN) n'a pas manqué de s'inviter dans la recherche sur le cancer. Que ce soit pour des tâches de classification [80, 139], de segmentation d'image médicale [3, 30, 64, 129] ou encore l'analyse de survie [161], l'apprentissage profond n'a cessé de s'imposer comme l'un des futurs outils clés pour la médecine personnalisée et a même réussi à démontrer son potentiel à surpasser les spécialistes de la santé dans plusieurs tâches [78]. Enfin, l'apprentissage multitâche, une extension naturelle de l'apprentissage machine qui consiste à entraîner un modèle sur plusieurs tâches à la fois plutôt qu'une seule, reste peu utilisé dans le domaine médical [77, 112, 122] et encore moins pour l'imagerie médicale [3, 30, 139, 144]. Malgré tout, l'apprentissage multitâche a le potentiel d'améliorer les performances de prédiction pour les différentes tâches comme l'ont démontré Wang *et al.* [138] en développant un modèle multitâche pour la détection de maladies de la rétine en 2019.

Apprentissage multitâche

Alors que l'apprentissage machine prend un certains essor durant les années 80, une idée simple et naturelle commence à faire son chemin. Si deux tâches sont suffisamment similaires, ne serait-il pas bénéfique de créer un seul modèle entrainé à effectuer les deux tâches plutôt que deux modèles indépendants ? Une telle approche ne pourrait-elle pas amoindrir les problèmes de sur-apprentissage ? En 1990, Abu-Mostafa nomme indice des caractéristiques utilisées comme tâches auxiliaires plutôt que comme entrées dans un modèle de type réseau de neurones [4]. Une tâche auxiliaire est alors semblable à une tâche secondaire sur laquelle le modèle sera entrainé

mais qui n'aura pas d'importance lors de l'évaluation du modèle sur l'ensemble de test. Ce n'est qu'en 1993 que Caruana introduit le concept d'apprentissage multitâche (AMT) [15] et le défini en 1997 comme suit : «L'apprentissage multitâche est un mécanisme de transfert inductif dont l'objectif principal est d'améliorer les performances de généralisation» [3]. En plus de fournir un premier schéma d'un type d'architecture que nous nommons aujourd'hui partage forcé (*hard-sharing*), il démontre que l'AMT permet d'obtenir de meilleurs résultats pour plus de tâches en même temps.

Récemment, l'apprentissage multitâche est revenu au goût du jour et de nombreuses études proposent de nouvelles architectures pour mieux partager l'information [81, 91, 100, 101, 162], des algorithmes de recherche d'architecture [49, 83, 133] et d'autre pour déterminer les tâches qui devraient être entraînées ensemble [41, 124]. D'autres chercheurs s'attaquent plutôt directement au problème de transfert négatif [18, 22, 68, 148]. Le transfert négatif est un phénomène qui se produit lorsque plusieurs tâches compétitionnent pour les ressources du modèle plutôt que de les partager et se caractérise par une perte de performance pour une ou plusieurs de ces tâches [17]. Pour évaluer les performances de l'AMT en vision par ordinateur, ce sont des jeux de données tels que *CelebA* [85], *Cityscape* [28] et *Taskonomy* [159] qui sont le plus souvent utilisés. Pour finir, l'apprentissage multitâche, ses mécanismes et les récentes recherches seront discutés plus en détail dans la section 3.

Problématique

Un système d'aide à la décision pour le traitement des lésions rénales peut prendre plusieurs formes. La figure 1 donne un exemple d'un tel système qui utiliserait la connaissance de la malignité, du sous-type et du grade pour proposer suggérer le traitement le plus adéquat pour un patient donné. Pour obtenir les caractéristiques nécessaires à partir d'imagerie médicale, un ou plusieurs modèles d'apprentissage automatique pourraient être utilisés. Mais encore faut-il les développer.

Avant qu'un modèle d'apprentissage machine ne soit utilisé par les médecins pratiquants, il doit passer par trois phases principales : (i) le développement et la validation interne ; (ii) la validation clinique ; et (iii) le déploiement du modèle. La première

3. Traduit et tiré de [16]

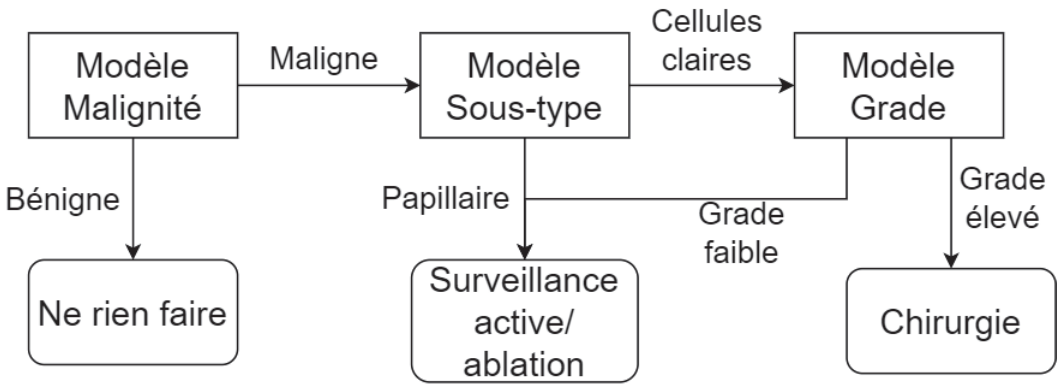


Figure 1 – Exemple d'un système d'aide à la décision. Image tirée de [4].

phase consiste simplement à la conception du modèle, son entrainement et son évaluation à l'aide d'un ensemble de données et très souvent la publication des résultats. La seconde partie consiste en une évaluation des performances du modèle avec un ensemble de données différent de celui utilisé durant la première phase. Il est évidemment nécessaire que cette validation externe soit effectuée par un ou plusieurs groupes d'individus indépendants au groupe de développeurs. La dernière phase consiste simplement au déploiement sur le terrain du modèle. Il faut donc, installer le modèle sur les différentes plates-formes et former les futurs utilisateurs. Pour rappel, l'étude effectuée dans ce mémoire se situe à la phase 1 de ce processus.

Dans le cas où l'on souhaite avoir des modèles de prédiction pour la classification de la malignité, du sous-type et du grade des lésions rénales, il faudrait passer par les phases 1 à 3 trois fois. Dans le pire des cas, après avoir segmenté manuellement la tumeur, le médecin pratiquant pourrait être obligé d'utiliser trois outils différents pour obtenir toute l'information nécessaire. Il serait plus rapide, économique et simple d'avoir un seul modèle qui effectue les trois tâches de classification. C'est la solution proposée. De cette manière, le développement, la validation clinique et le déploiement du modèle pourraient être faits une seule fois au lieu de trois fois.

Cependant, il est connu que l'apprentissage multitâche ne représente pas une solution miracle. Bien qu'il arrive parfois qu'un modèle multitâche obtienne des performances supérieures que plusieurs modèles à tâche simple indépendants, l'entrainement

4. Vallières, M. et al., Radiomics-Based decision-support system in renal cell carcinomas, ICCR 2019

multitâche peut aussi résulter en une perte de performances lorsque le modèle subit les effets du transfert négatif [17, 22, 148]. Puisqu'il serait difficile de justifier le déploiement d'un modèle aux performances moindre et dans le cas où il est impossible d'obtenir de meilleures performances pour toutes les tâches, il est toujours possible de désigner des tâches auxiliaires dont les performances finales ne comptent pas et qui n'ont pour but que d'aider dans l'entrainement des tâches principales. Si la plupart du temps de telles tâches auxiliaires ne sont pas disponible avec les données, il est tout de même possible de créer des tâches auxiliaires synthétiques à partir des données, comme par exemple, en demandant à un modèle de prédire des caractéristiques radiomiques.

Pour finir, alors que plusieurs modèles de prédictions ont été produits pour effectuer des tâches en lien avec les tumeurs rénales, que ce soit pour la classification de la malignité [79, 128, 145, 146, 163], du sous-type [44, 53, 130, 131] ou du grade [32, 52, 86, 151], ou encore la segmentation de ces tumeurs [129], aucun modèle d'apprentissage multitâche n'a encore été développé pour la classification des tumeurs rénales et le potentiel d'une telle approche reste inconnu.

Objectif

Dans le cadre de ce projet, nous nous sommes fixés trois objectifs distincts.

1) Confirmer qu'un réseau de neurones à convolution 3D fonctionnant avec des images par résonance magnétique est mieux adapté qu'un modèle de base utilisant des données cliniques tabulaires, telle qu'une MVS, pour la prédiction de la malignité d'une lésion rénale et dans le cas où elle est maligne, la prédiction du sous-type et du grade.

2) Vérifier si ces différentes tâches de classification de tumeurs rénales peuvent bénéficier de l'apprentissage multitâche. En d'autres termes, est-ce qu'un réseau de neurones entrainé à effectuer plusieurs de ces trois tâches en même temps peut obtenir de meilleures performances que nos modèles à tâche simple ?

3) Déterminer s'il est possible d'identifier les caractéristiques radiomiques les mieux

adaptées pour être utilisées comme tâches auxiliaires dans un cadre d'apprentissage multitâche avec nos problèmes de classification de tumeurs rénales énoncés précédemment.

Pour finir, nous voudrions proposer au final un modèle multitâche unique qui pourrait donc accélérer les étapes de validation et de déploiement. Nous mettrons une attention particulière à utiliser des architectures qui soient le plus simples possible, pour qu'elles soient facilement adaptable à d'autres problèmes.

Structure du mémoire

Dans le présent mémoire, nous tenterons de vérifier la validité de ces hypothèses à travers six chapitres. Le premier chapitre abordera les concepts théoriques de l'imagerie médicale, alors que les deux chapitres suivant aborderons respectivement les concepts de bases de l'apprentissage machine et l'apprentissage multitâche. Le chapitre 4 donnera tout les détails de notre méthodologie. Celui qui suit présentera les résultats de notre recherche. Le dernier chapitre sera dédié à l'analyse de nos résultats.

Chapitre 1

Analyse d'imagerie médicale

Dans ce chapitre, nous aborderons l'imagerie par résonance magnétique, la représentation mathématique d'une image et les caractéristiques radiomiques. Si le premier sujet a pour but d'écrire la façon dont nos données imagées ont été acquises, le second servira à mettre en place quelques concepts fondamentaux à l'imagerie de manière plus générale qui seront largement réutilisés à travers les autres sections et notre chapitre 2. La dernière sous-section, quant à elle doit simplement expliciter la nature des caractéristiques que nous utiliserons principalement comme tâches auxiliaires lors de nos dernières expériences.

1.1 Imagerie par résonance magnétique

Avant même le début du 20e siècle, des méthodes d'imagerie sont développées pour visualiser l'intérieur du corps humain sans nécessiter d'opération chirurgicale. Dès 1896, les rayons X sont utilisés pour la radiologie dentaire [118]. Vers 1930, une première utilisation apparaît dans une tentative infructueuse pour diagnostiquer des tumeurs cérébrales [94]. Une vingtaine d'années plus tard, la tomographie par émission de positrons qui est développée pour localiser les tumeurs cérébrales [123]. Toutefois, il faudra attendre les années 70, avec les avancements en informatique qui permettent d'effectuer des calculs longs et complexes, pour voir apparaître l'imagerie médicale telle que nous la connaissons aujourd'hui. C'est à cette époque que sont développées à la fois la tomodensitométrie et l'imagerie par résonance magnétique. La présente

sous-section abordera les grandes lignes de la théorie et du fonctionnement de l'IRM.

1.1.1 Physique et résonance magnétique nucléaire

L'imagerie par résonance magnétique se base sur la résonance magnétique nucléaire. Plus précisément, c'est le champ magnétique émis par les molécules d'eau présentes dans les différents tissus du corps qui est manipulé et utilisé pour produire des images. Toutes les molécules d'eau sont composées d'un atome d'oxygène et de deux atomes d'hydrogène, ce sont ces derniers qui nous intéressent. Un atome d'hydrogène n'est composé que d'un seul proton, qui, de façon imagée, agit comme un aimant en possédant un pôle nord et un pôle sud. Le moment magnétique de ce proton a une orientation et il tourne sur lui-même autour de son axe. Le moment intrinsèque angulaire de ce moment magnétique est nommé spin. Les spins effectuent un mouvement de rotation incliné aussi appelé mouvement de précession.

Naturellement, l'orientation des spins est aléatoire et il en résulte que le champ magnétique produit par les différents tissus est nul. C'est pourquoi, la première étape de l'IRM est d'appliquer un champ magnétique externe B_0. Assujetti à ce champ, les spins vont s'aligner avec la direction de B_0, mais pas tous dans le même sens. Nous nommerons spins de basse énergie ceux qui sont orienté dans le même sens que B_0, alors que les autres porteront le noms de spins de haute énergie (voir figure 1.1). Dans cette situation, les spins présente une résonance à une fréquence bien précise qui est la fréquence de Larmor ω, donnée par l'équation

$$\omega = \gamma B,$$

où γ représente le ratio gyromagnétique, une constante unique à chaque type d'atome. Pour un type d'atome donné, la fréquence de Larmor dépend uniquement du champ magnétique B.

Au final, l'aimantation des différents tissus du corps sera formée de deux composantes. Une composante longitudinale parallèle au sens du champ magnétique que l'on associe à l'axe z et une transversale perpendiculaire à B_0 qui est représenté dans le plan xy. La composante longitudinale résulte des spins de basse énergie opposés aux spins de haute énergie. Les spins de basse énergie étant plus nombreux, la composante

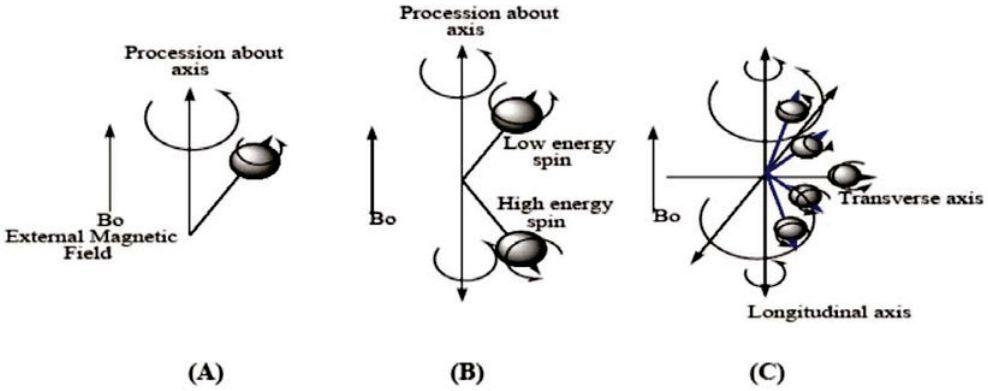


Figure 1.1 – Mouvement de précession d'un spin en présence d'un champ magnétique B_0. Image tirée de [107].

longitudinale est positive. La composante transversale est, quant à elle, produite par le mouvement de précession des différents spins. Puisque ces derniers ne sont pas en phase, la composante transversale est donc nulle.

Pour maximiser l'aimantation transversale, il est possible d'utiliser un émetteur pour émettre un pulse de radiofréquence à la fréquence de Larmor des spins. Pendant cette phase, qui sera nommée la phase d'excitation, des spins d'énergie basse passeront en énergie haute en absorbant l'énergie de cette onde, résultant ainsi en une diminution de l'aimantation longitudinale. Dans un même temps, les spins vont entrer en phase et la composante transversale deviendra maximale. À l'arrêt de cette onde de radiofréquence, la période de relaxation sera entamée par les spins qui retourneront à leur état initial et se déphaseront. Durant cette phase, l'énergie absorbée précédemment sera libérée sous la même forme qu'elle fût absorbée.

1.1.2 Temps de relaxation et contraste

Lors de la phase de relaxation, l'aimantation tissulaire longitudinale et transversale retournent à leur état d'équilibre, c'est-à-dire à leur état avant le pulse RF, à des vitesses différentes selon le type de tissus. C'est cette différence de vitesse qui va permettre de créer un contraste dans l'image et d'identifier les différentes structures anatomiques.

En résonance magnétique nucléaire, il y a deux temps de relaxation que nous nommons T1 et T2. Le temps T1 est associé au temps qu'il faut pour que la composante longitudinale du vecteur de magnétisation retrouve 63% de sa valeur initiale, alors que le temps T2, lui, est lié au temps nécessaire pour que la composante transversale ne soit plus qu'à 36% de la valeur maximale atteinte durant la phase d'excitation. Pour mieux comprendre pourquoi le temps T1 n'est pas égal au temps T2, il faut savoir que la restitution de la valeur longitudinale est due au phénomène où les spins excités retournent à leur état de basse énergie tandis que la perte de l'aimantation transversale est due au déphasage des spins.

Si l'on pose $\vec{i}, \vec{j}$ et $\vec{k}$ comme vecteurs unitaire servant de base à notre espace, alors l'équation de Bloch [93] qui décrit le comportement du vecteur de magnétisation s'écrit comme suit :

$$\frac{dM(t)}{dt} = M(t) \times \gamma B(t) - \frac{M_x(t)\vec{i} + M_y(t)\vec{j}}{T2} - \frac{(M_z(t) - M_0)\vec{k}}{T1}. \tag{1.1}$$

La résolution de l'équation 1.1 permet de calculer les valeurs de l'aimantation M_x, M_y et M_z.

1.1.3 Application de gradients

Si le champ magnétique B_0 est uniforme dans le scanneur utilisé pour imagé l'anatomie d'un patient, il ne sera pas possible de distinguer l'aimantation des différents tissus du corps. Il faut donc trouver une astuce afin d'ajouter une composante de localisation spatiale pour régler ce problème. L'idée principale pour identifier la position spatiale du signal est d'ajouter des gradients aux champs magnétiques afin de modifier son intensité. Puisqu'il y a trois dimensions, il y aura aussi trois gradients.

Le premier se nomme le gradient de sélection de coupe G_s. Ce gradient présent durant toute la durée de l'acquisition d'image, consiste à faire varier linéairement l'intensité de B_0 tout au long de l'un des axes, par exemple l'axe longitudinal (des pieds à la tête du patient). Ainsi la fréquence de précession des spins sera différente, tout comme leur fréquence de Larmor, et il sera donc possible d'ajuster le pulse de RF afin d'exciter les spins présents dans une seule tranche du corps à la fois.

Le second gradient est nommé gradient de phase G_p et il modifie l'intensité du

champ magnétique le long d'un autre axe. Il est appliqué pendant un court instant avant la réception du signal afin de modifier la vitesse de précession des spins. Suite à l'application de ce gradient, le mouvement de précession des spins sera de la même vitesse pour tous, mais ils auront une phase différente en fonction de leur position dans l'axe choisi.

Le dernier gradient appliqué est celui de fréquence G_f et il est appliqué le long du dernier axe restant pendant la réception du signal afin de modifier la fréquence de Larmor. Ainsi, la fréquence du signal nous permettra de localiser sa position sur la dernière dimension.

Maintenant que la position en z (profondeur) du signal est déterminée par le moment d'acquisition et que la position dans les deux dernières dimensions est encodée dans la phase et la fréquence du signal, nous pouvons réécrire la fonction du champs magnétique $B(t)$ présente dans l'équation de Bloch (1.1) comme étant $B(t) = B_0 + G_s + G_p(t) + G_f(t)$.

1.1.4 Séquence écho de spin

Bien qu'il existe une multitude de séquences différentes pour l'imagerie par résonance magnétique, nous prendrons le temps de décrire seulement la séquence écho de spin pour les pondérations T1 (illustré dans la figure 1.2) et T2, puisque ce sont celles-ci qui sont utilisées dans nos travaux.

En pratique, seule l'aimantation transversale peut être mesurée puisque le récepteur est aligné dans le sens de B_0. Ainsi pour la séquence en pondération T1, l'astuce pour mesurer la composante longitudinale consiste dans un premier temps à envoyer un pulse de RF de 90°, afin de mettre en phase les spins, et d'envoyer un second pulse de 180° avant la fin de la phase de relaxation, afin de retransformer l'aimantation longitudinale en aimantation transversale. Puisque le temps de relaxation des différents tissus est différent, la valeur du vecteur longitudinale ne sera pas la même avant l'envoi du second pulse. Ainsi, il y aura une différence de signal transversal entre les tissus après le 2e pulse ; c'est ce qui permettra d'obtenir un contraste visible entre les tissus. Ces différences de signals seront donc dues aux différences d'aimantations longitudinales, mais aussi aux différences de relaxation des aimantations transversales.

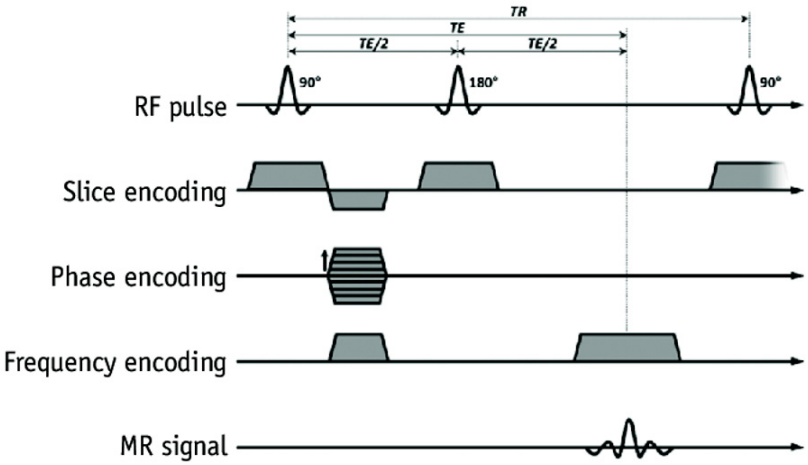
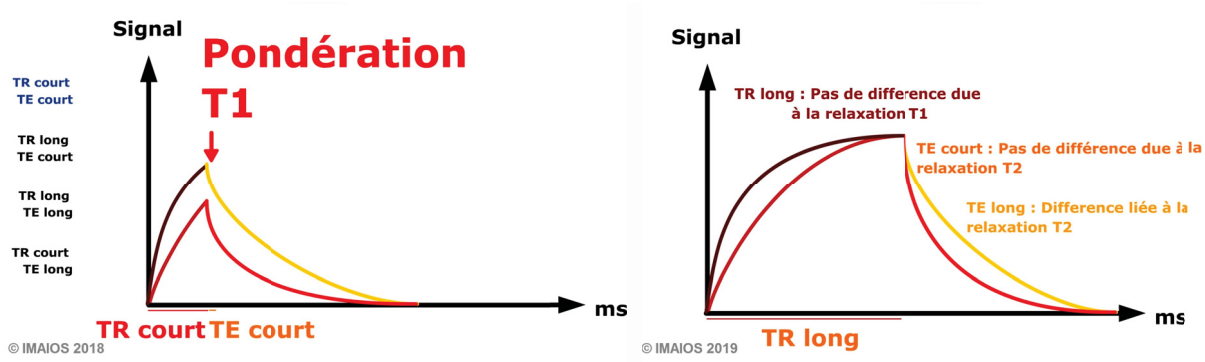


Figure 1.2 – Séquence écho de spin. Image tirée de [67].

C'est pourquoi nous parlerons de pondération T1 plutôt que de T1 tout court.

Pour la pondération T2 du signal, il y aura aussi deux pulses de RF qui seront émis, mais pas pour les mêmes raisons. Lorsque les spins entrerons en phase de relaxation, après l'arrêt du premier pulse de 90°, les spins vont commencé à se déphaser. Après un certain moment, un second pulse de RF sera envoyé, mais cette fois-ci de 180°. Ce dernier est utilisé pour enlever les hétérogénéités dues à B_0. Ainsi, après le second pulse, les spins font se rephaser avant de recommencer leur phase de relaxation. Toutefois, le vecteur d'aimantation transversale ne retournera pas à sa valeur maximale même une fois rephasé. Cela sera dû aux propriétés T2 du tissu. Le pulse de 180° sert donc à mesurer la décroissance du signal T2.

En IRM, l'intervalle de temps entre l'émission du pulse de RF de 90° et l'instant où la puissance du signal est mesurée est nommé temps d'écho (TE), alors que l'intervalle de temps entre l'émission de deux pulses de 90° est nommé temps de répétition (TR). Il est important de noter que pour la pondération T2, le pulse de 180° est émis à la moitié du temps d'écho. Bien que les pondération T1 et T2 sont acquise avec le même type de séquence, elles nécessitent des temps d'écho et de répétition qui sont différents. La figure 1.3 illustre la différence entre ces deux modalités.



(a) Pour la pondération T1, un TE et un TR courts sont utilisés pour maximiser la différence de signal.

(b) Pour la pondération T2, un TE et un TR longs sont utilisés pour maximiser la différence de signal.

Figure 1.3 – Contraste du signal dans les pondérations T1 et T2. Image tirée de [1].

1.1.5 Amélioration du contraste

Bien que la résonance magnétique est déjà mieux adaptée que la tomographie computationnelle pour différencier les tissus, il existe encore des méthodes pour améliorer cela. L'une d'elles consiste à injecter ce que l'on appelle un agent de contraste dans le corps du patient. Ce sont des produits médicamenteux injectés dans le corps qui ont pour rôle de réduire le temps de relaxation T1 et/ou T2 des tissus avoisinants. Ce faisant, les zones où l'agent de contraste est présent seront plus claires, sur les images en pondération T1, si c'est la relaxation T1 qui est raccourcie et plus sombre sur les images en pondération T2 ou T2* si c'est le temps de relaxation T2 qui est affecté. Il est donc détecté indirectement grâce à son interaction avec les molécules d'eau présentent dans les tissus. Couramment, les images de pondération T1 et de pondération T2 qui ont été rehaussées par un agent de contraste sont renommées respectivement T1C et T2C ou encore T1CE et T2CE (C : contraste ; CE : contraste amélioré).

Les agents de contraste les plus souvent utilisés sont ceux basés sur le gadolinium qui affecte surtout le temps de relaxation T1. Pour réduire le temps de relaxation T2, ce sont des solutions formées à partir de particules de fer qui sont utilisées [73].

1. Cours d'IRM (Imagerie par résonance magnétique) de IMAIOS, consulté le 3 février 2022 :https://www.imaios.com/fr/e-Cours/e-MRI.

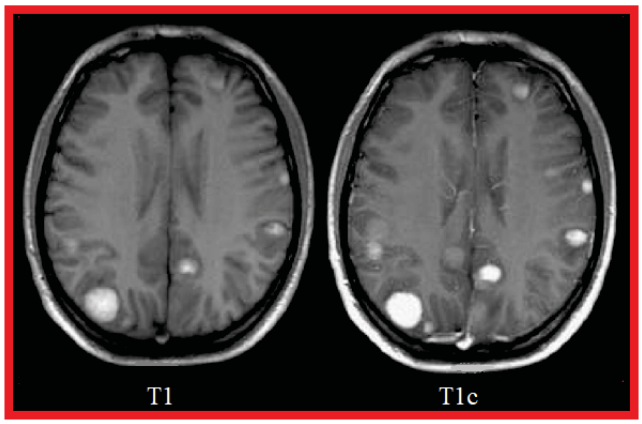


Figure 1.4 – L'utilisation d'un agent de contraste permet de rehausser le contraste des images en pondération T1. L'image de gauche est de pondération T1 alors que celle de droite est dite de pondération T1C. Image tirée de [1].

1.2 Représentation mathématique d'une image

Précédemment, nous avons vu comment il était possible de produire une image à l'aide de la résonance magnétique. Cette sous-section mettra au clair la façon dont ces images sont représentées en informatique. Les concepts clés au prétraitement de nos images, tels que la position et l'orientation du patient dans l'image par rapport à la scène, ainsi que le calcul du volume, de la surface et du centre de masse d'une région d'intérêt donnée, seront aussi abordés.

En mathématiques et en informatique, une image 2D en niveaux de gris est exprimée comme une matrice de nombres où chacun d'entre eux représente un pixel dans l'image (voir figure 1.5). Lorsque cette image est en couleur, il faut trois matrices pour la représenter puisque la couleur de chaque pixel est définie par un code de trois nombres nommé RVB (rouge, vert et bleu). Il y aura donc une matrice pour exprimer la teinte de rouge de chaque pixel, une pour le vert et une autre pour le bleu. Étant donné que les éléments r_{ij}, v_{ij} et b_{ij} donnent tous de l'information sur le même pixel ij, ces matrices sont nommées les canaux de l'image. Lorsque l'image est en 3D, comme c'est souvent le cas en imagerie médicale, ce ne sont pas des matrices qui seront utilisées, mais plutôt des 3-tenseur et les pixels de ce type d'image seront renommés voxels.

1.2.1 Position, orientation et taille des voxels

En imagerie médicale, de nombreuses données accompagnent l'image. Parmi ces informations figurent la taille des voxels, l'orientation de l'image et la position du patient par rapport à un point de référence dans la machine. Ces trois informations peuvent être représentées à l'aide d'une matrice affine, mais avant d'aller plus loin, commençons par discuter des trois points mentionnés plus haut.

Taille des voxels : À la différence de leur contrepartie 2D, les voxels possèdent une profondeur dans une troisième dimension. Ils auront une largeur, une longueur et une profondeur pour exprimer l'espace qu'ils occupent et ils seront tous de même dimension. Dès lors que la longueur est égale à la largeur et à la profondeur, les voxels seront dits isotropiques.

Orientation : Un 3-tenseur est une version en trois dimensions d'une matrice, ce qui implique qu'il faut trois coordonnées pour spécifier un élément. Soit le tenseur T et ses éléments t_{ijk}, où i, j et k qui représentent les coordonnées le long de chacune des dimensions que nous nommerons x, y et z, alors la question est donc : que représente chacune de ces dimensions ? Puisqu'il s'agit dans notre cas de l'image d'un patient, nous utiliserons des termes plus précis que simplement largeur, longueur et profondeur. Ce seront les termes gauche-à-droite, postérieur-à-antérieur et inférieur-

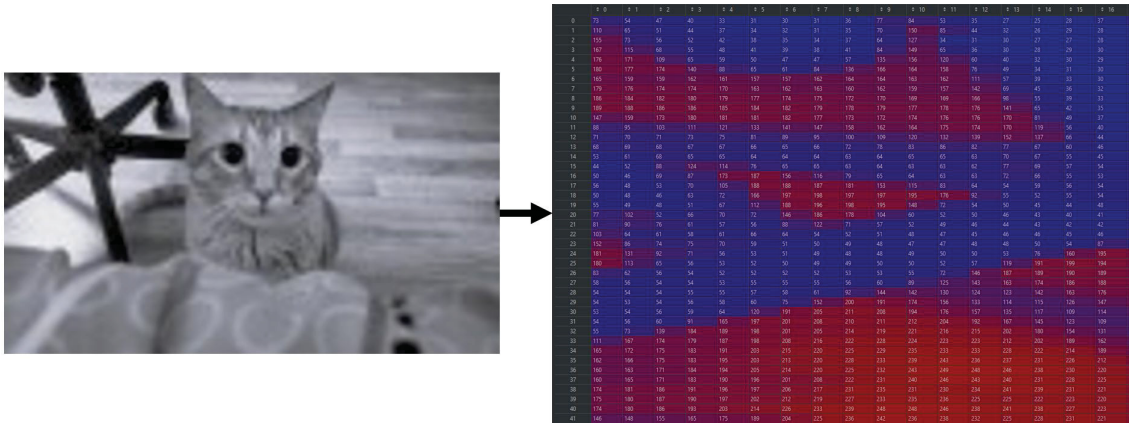


Figure 1.5 – Représentation d'une image 2D en teintes de gris sous forme de matrice.

à-supérieur qui seront utilisés. De cette manière, si nous définissons que la direction positive de l'axe x est gauche-à-droite, cela nous indique aussi que les éléments dont la coordonnée en x est de 0, sont ceux qui sont les plus à gauche du patient. Il est commun d'indiquer l'orientation à l'aide d'un code de trois lettres, tel que RSA qui signifie x : gauche-à-droite, y : inférieur-à-supérieur et z : postérieur-à-antérieur.

Position : La position correspond simplement à la distance en millimètres du voxel en position $(0, 0, 0)$, c'est-à-dire en haut à gauche dans la première tranche de l'image, par rapport à un référentiel dans l'espace de la machine dans laquelle le patient a été placé durant l'examen. Cette information peut être utile pour aligner approximativement deux images si elles ont été prises durant la même séance, sinon il sera nécessaire d'utiliser des méthodes plus sophistiquées telle que le recalage.

Matrice affine : La taille des voxels, l'orientation et la position peuvent toutes être encodées dans la matrice dite des transformations affines ou plus simplement matrice affine. En infographie, elle permet d'appliquer toutes les combinaisons de transformations affines à l'aide d'une seule multiplication de type vecteur-matrice qui sera appliquée aux coordonnées de chacun des voxels. En imagerie médicale, cette matrice sert à faire passer les coordonnées de l'espace des voxels i, j, k vers ceux de l'espace de référence, aussi nommé la scène, i', j', k'. Ce changement d'espace, s'effectue simplement à l'aide de l'équation

$$\begin{bmatrix} i' \\ j' \\ k' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ j \\ k \\ 1 \end{bmatrix}. \tag{1.2}$$

On compte trois types de transformations affines : la translation, la rotation et le zoom. Regardons ces trois opérations.

Translation : La translation est l'opération affine qui justifie l'utilisation d'une matrice de taille 4×4 plutôt qu'une de taille 3×3. Normalement, cette opération aurait pu être faite à l'aide d'une addition de deux vecteurs, mais pour l'exprimer

sous forme de multiplication matricielle, il n'y pas d'autres solutions que d'ajouter une colonne dans laquelle l'élément a, b ou c (voir la Matrice T) va multiplier le 1 du vecteur des coordonnées avant de s'additionner à la coordonnée correspondante. Ce sont donc les valeurs a, b et c qui nous permettent de connaître la position du patient dans l'image.

$$T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation : Cette opération n'est pas vraiment différente avec notre matrice de taille 4×4 qu'elle ne l'aurait été avec une matrice de taille 3×3. Par exemple, pour effectuer une rotation autour de l'axe z, la matrice affine prendra la forme suivante

$$R_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

C'est à l'aide d'opérations de rotation qu'il est possible d'exprimer l'orientation de l'image.

Zoom : Le zoom ou encore mise à l'échelle, sert à agrandir ou réduire la distance entre les voxels qui est de 1 dans l'espace des voxels. Dans le cas où il n'y pas de rotation à effectuer, ce sont donc les valeurs s_1, s_2 et s_3 de la matrice

$$S = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

qui nous permettent de connaître la taille des voxels en millimètres.

1.2.2 Région d'intérêt et calculs divers

En imagerie médicale, il est courant de désigner une région de l'image comme étant d'intérêt. Généralement, elle permet de délimiter l'espace occupé par une tumeur, une lésion ou encore un organe que l'on souhaite étudier. La région d'intérêt (RI) est identifiée par une segmentation qui est créée à partir d'un système de coordonnées et de triangles ou à l'aide d'une seconde image 3D dont les voxels ne peuvent prendre qu'une valeur parmi deux possibilités : 1 si le voxel est dans la région d'intérêt et 0 sinon. Cette dernière représentation est nommée masque de segmentation. La segmentation des tumeurs est généralement faite par un expert de la santé spécialisé. Elle nous permet, entre autres, de calculer diverses caractéristiques de la région telle que l'aire de la surface, son volume ou encore son centre de masse.

Aire de la RI : Cette mesure est la plus simple à calculer, mais il faut obligatoirement passer par le système de coordonnées et de triangles. À partir de ce système, il suffit de calculer l'aire de chaque triangle et de les additionner.

Volume de la RI : Il existe deux façons distinctes de calculer le volume de la région d'intérêt. La plus simple consiste à utiliser le masque de segmentation pour calculer le nombre de voxels qui sont inclus dans la RI avant de multiplier ce nombre par le volume d'un voxel. La seconde, plus complexe, utilise le système de coordonnées et de triangles. Pour chaque triangle, il faudra prendre les sommets de ce dernier (t_1, t_2 et t_3) et y ajouter le centre de référence des voxels comme nouveau sommet pour former un tétraèdre. À partir du volume occupé par chaque tétraèdre et de l'orientation du vecteur normale de sa base, il sera possible de calculer le volume total à l'aide des équations suivante :

$$V_t = \frac{t_1 \cdot (t_2 \times t_3)}{6},$$

$$V = |\sum_{t=1}^{n} V_t|.$$

Centre de masse de la RI : Pour calculer le centre de masse (cdm) de la région d'intérêt, il faut prendre le masque de segmentation et considérer les coordonnées de

chacun des voxels v comme étant un vecteur ayant pour origine le centre de référence de l'espace des voxels. Ensuite, comme le montre l'équation suivante,

$$cdm = \frac{1}{n}\sum_{i=1}^{n}\vec{v}_i,$$

il suffit de faire la moyenne de ces vecteurs pour connaître les coordonnées du centre de masse.

1.3 Caractéristiques radiomiques

Les caractéristiques radiomiques sont un type de biomarqueur, c'est-à-dire des caractéristiques biologiques mesurables. Plus précisément, ce sont des biomarqueurs d'image quantitatifs. Ils ont pour objectif de décrire divers aspects d'une région d'intérêt donnée dans une image tels que la sphéricité, le contraste ou encore la rugosité. Elles se divisent en 11 familles qui se divisent en trois groupes principaux : morphologiques (une famille), basées sur l'intensité des voxels (quatre familles) et texturales (six familles). Afin qu'elles soient reproductibles et fiables, les caractéristiques radiomiques doivent être calculées à partir d'images ayant subi un prétraitement strict qui est décrit dans le manuel de référence de l'*image biomarker standardisation initiative* (IBSI) [158] et qui sera résumé ci-dessous.

1.3.1 Prétraitement des images

Comme le montre la figure 1.6, les images doivent passer par tout un processus de prétraitement avant de pouvoir en extraire les caractéristiques souhaitées. Nous résumerons dans les prochains paragraphes les différentes étapes du prétraitement. Nous mettrons davantage l'accent sur les étapes les plus complexes telles que la segmentation, l'interpolation, la resegmentation et la discrétisation.

Pour commencer, la conversion des données vise à convertir une image dans son format brut vers une forme plus standard et interprétable. Un exemple courant en imagerie médicale est la valeur de fixation normalisé (SUV), qui est utilisé pour définir la fixation du Fluorodéoxyglucose (FDG) en tomographie par émission de positron

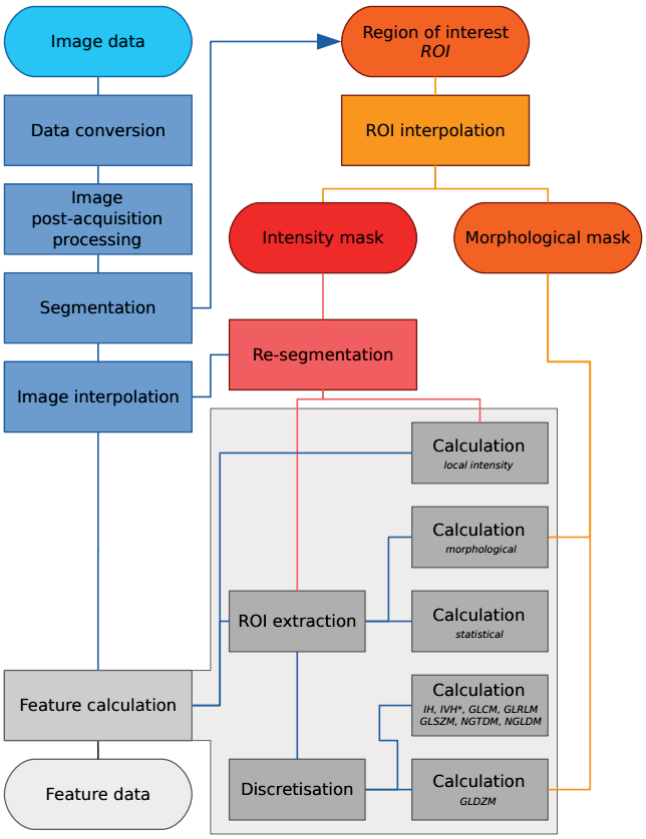


Figure 1.6 – Schéma des différentes étapes pour le prétraitement d'image avant l'extraction des caractéristiques radiomiques. Image tirée de [158]

(TEP) [14]. Ensuite, des corrections sur l'image, telles que du débruitage, peuvent être apportées. Ces corrections sont faites durant l'étape de prétraitement post-acquisition.

Lorsque la taille des voxels est différente d'une image à une autre dans un même jeu de données, il devient nécessaire d'uniformiser la taille de ces voxels. De plus, afin de rendre certains calculs invariants aux rotations, il faut aussi que les voxels soient isotropiques. L'interpolation consiste donc à changer l'espacement de ces derniers et d'interpréter la valeur que doivent prendre ces nouveaux voxels. Il existe plusieurs algorithmes d'interpolation tels que l'interpolation trilinéaire, tricubique ou encore du plus proche voisin. Le masque de segmentation doit aussi être interpolé pour garder les mêmes dimensions que l'image. Dans ce cas, il est conseillé pour le masque de segmentation d'utiliser une interpolation trilinéaire et d'arrondir la valeur des voxels

ou d'utiliser une interpolation du plus proche voisin.

Alors que le masque servant au calcul des caractéristiques radiomiques morphologiques ne subit par la suite aucune modification, il peut être nécessaire de resegmenter celui utilisé pour calculer certaines caractéristiques radiomiques basées sur l'intensité des voxels. C'est le masque d'intensité. Cette resegmentation est faite à l'aide d'un algorithme qui discrimine certains voxels en fonction de si leur intensité est considérée comme une valeur aberrante ou si elle sort d'un certain intervalle définit par l'utilisateur. Certains voxels à l'intérieur de la RI peuvent ne pas être retirés dans ce processus. Le masque d'intensité de la région d'intérêt est par la suite utilisé pour déterminer les voxels de l'image qui seront utilisés pour calculer certaines caractéristiques radiomiques. C'est l'étape de l'extraction de la RI.

L'étape finale du prétraitement des images est la discrétisation des intensités. Cette étape, nécessaire dans le calcul de certaines caractéristiques texturales, a pour but de limiter les valeurs que peut prendre l'intensité des voxels à une quantité dénombrable. Il y a au moins deux méthodes pour y parvenir. La première consiste à déterminer à l'avance le nombre de valeurs possibles que peut prendre l'intensité des voxels dans une image. Par exemple, dans le cas où l'on souhaite qu'il y ait n valeurs possible, il suffit de prendre l'intervalle défini par la différence entre la plus grande intensité et la plus petite et on sépare cet intervalle en n sous-intervalles uniformes. Les intensités qui tombent dans un même sous-intervalle seront donc représentées par une même valeur. La seconde méthode, quant à elle, consiste à prédéterminer la longueur des intervalles plutôt que leur nombre.

Pour finir, les différentes étapes énoncées précédemment ont toutes un impact très important dans le calcul des caractéristiques radiomiques. Certains paramètres du prétraitement des images peuvent même parfois être considérés comme des hyperparamètres (voir sous-section A.2) et leur choix peut avoir des conséquences sur les résultats des modèles d'apprentissage machine qui sont basés sur l'utilisation des caractéristiques radiomiques.

1.3.2 Les différentes familles de caractéristiques radiomiques

Les caractéristiques radiomiques sont réparties en trois groupes qui cumulent ensemble 11 familles différentes pour un total de 172 caractéristiques d'images.

Morphologique : Le premier groupe ne contient qu'une seule famille et il s'agit des caractéristiques morphologiques. Comme son nom l'indique, ces caractéristiques décrivent la forme de la RI. Voici quelques exemples : surface, volume et sphéricité.

Intensité : Le groupe des caractéristiques radiomiques d'intensité, qui en rassemble 48, est formé de quatre familles distinctes.

Intensité locale : Tente de caractériser l'intensité de la région de $1cm^3$ la plus lumineuse dans la RI.

Statistique basée sur l'intensité : Un ensemble de mesures statistique de bases, telles que la variance, le kurtosis, l'écart interquartile, calculées à partir de l'intensité des voxels présents dans la RI.

Histogramme d'intensité : Semblable à la précédente famille, à la différence que les intensités sont discrétisées à la manière d'un histogramme.

Histogramme d'intensité-volume : Les radiomiques de cette famille « décrivent la relation entre l'intensité discrétisée i et la fraction du volume contenant au moins l'intensité i, v » [158].

Texturales : Pour finir, le dernier groupe est celui des caractéristiques radiomiques texturales qui réunit 95 caractéristiques réparties en six familles. Pour chaque famille, il faut d'abord calculer une ou plusieurs matrices, agréger l'information dans une seule matrice et ensuite calculer des statistiques à partir de celle-ci. C'est la définition de la matrice à calculer qui décrit le mieux une famille de radiomiques texturales.

Matrice de co-occurence des niveaux de gris (MCCNG) : Cumule la fréquence ou calcul la distribution à laquelle une intensité discrétisée j est adjacente à une intensité discrète i dans une direction donnée.

Matrice de longueur de plage de niveaux de gris : Cumule la fréquence des suites consécutives de longueurs maximales l, d'une même intensité discrétisée i, dans une direction donnée à l'intérieur de la RI.

Matrice de taille des zone de niveaux de gris (MTZNG) : Cumule la fréquence des groupes rassemblant n voxels adjacents et d'une même intensité discrétisée i dans la RI.

Matrice de distance de zone de niveau de gris (MDZNG) : Cumule la fréquence des groupes de voxels adjacents d'une même intensité discrétisée i, de taille quelconque, qui se situe à une distance d de la frontière de la RI.

Matrice de différence des voisinage de niveau de gris (MDVNG) : Cette matrice cumule, pour chaque intensité i, le nombre absolu et la distribution des voxels de cette intensité qui sont et qui possèdent un voisinage d'un rayon donné (distance de Chebyshev) dans la RI, ainsi que la somme des différences d'intensité entre les voxels d'intensité i, possédant un tel voisinage, et la moyenne des intensités de leur voisinage respectif.

Matrice de dépendance des voisinage de niveau de gris (MDVNG) : Cumule la fréquence à laquelle un voxel d'intensité i, qui possède un voisinage d'un rayon donné (distance de Chebyshev) dans la RI, est voisin à n voxels de même intensité et qui sont présents aussi dans la RI.

Chapitre 2

Apprentissage profond

Avant d'aller plus loin dans la méthodologie et nos expériences, nous discuterons de la théorie sous-jacente à l'apprentissage automatique. En premier lieu, nous mettrons en place la terminologie et la notation qui seront utilisées pour le reste du mémoire en discutant de l'apprentissage supervisé. Dans un second temps, nous ferons un rappel sur les algorithmes d'apprentissage classique et les réseaux de neurones à convolution. Afin d'éclaircir certains détails qui seront présents dans notre méthodologie, les sections suivantes traiteront de régularisation et d'augmentation de données.

2.1 Apprentissage supervisé et autres concepts

L'apprentissage supervisé vient de l'idée d'une machine qui s'entraîne à modéliser une fonction $f : X \to Y$ à partir d'un échantillon de X et d'un superviseur qui fournit à la machine les cibles y associées à l'échantillon [46]. Dans cette section, nous présenterons les concepts clés liés aux différents ensembles de données, au processus d'entraînement, le sur- et sous-apprentissage et la recherche d'hyperparamètres.

2.1.1 Données et observations

Dans le cadre de l'apprentissage supervisé, nous considérons toujours un ensemble d'observations $D_{\text{obs}} = \{(x_i, y_i)\}_{i=1}^{n}$ formé de données x_i qui sont chacune associées à une étiquette y_i que nous devons prédire. Nous pourrions dire que les observa-

tions faisant parties de D_{obs} ne sont qu'un échantillon d'une distribution inconnue $D^* = (X, Y)$. Cet ensemble est très souvent séparé en trois parties : l'ensemble d'entraînement D_{train}, de validation D_{valid} et de test D_{test}. En fonction de la tâche à effectuer, y_i peut prendre plusieurs formes. Pour une tâche de régression, y_i sera représentée par un scalaire, pour la classification, ce sera un entier ou un vecteur de nombre binaire (encodage 1 parmis n) et pour une tâche de segmentation, y_i sera représentée par un tenseur. Dans le cadre de l'apprentissage multitâches, elle sera représentée par un ensemble d'étiquette $\{y_i^{(j)}\}_{j=1}^{t}$ où $y_i^{(j)}$ représente l'étiquette associée à la donnée i pour la tâche T_j. Il peut parfois arriver que pour une donnée d'entrée x_i, il n'y ait pas d'étiquette y_i^j qui lui soit associée pour la tâche T_j, mais qu'il y en ait une pour la tâche T_k.

2.1.2 Entraînement et fonction de coût

En apprentissage supervisé, on émet toujours l'hypothèse de l'existence, pour une tâche donnée, d'une fonction cible $f^*(x)$ qui permet de prédire parfaitement l'étiquette associée à chaque donnée x possible [106]. L'objectif est de créer une fonction $f(x, \theta)$ dont les poids θ seront ajustés à l'aide des observations de D_{obs} afin d'approximer $f^*(x)$. On peut voir les choses autrement. Nous pouvons supposer que $f^*(x) \sim Z = Y|X$. Dans ce cas-là, nous pourrions dire que $f(x, \theta)$ tente d'imiter une distribution de probabilité. Dans le cas d'un problème de classification nous dirions plutôt que $Z = \mathbb{P}(Y|X)$.

L'approximation de $f^*(x)$ par $f(x, \theta)$ se fait par la résolution d'un problème d'optimisation du type $\arg\min_{\theta} \mathbb{E}_{D_{\text{train}}} [\mathcal{L}(f(x, \theta), y)]$ où $\mathcal{L}$ est la fonction de perte à minimiser qui mesure la divergence entre notre modèle $f(x, \theta)$ et la fonction cible $f^*(x)$. La résolution de ce problème d'optimisation est nommée phase d'entraînement.

En fonction de l'algorithme d'apprentissage choisi, la phase d'entraînement peut se dérouler de deux façons différentes. La première est la résolution du problème d'optimisation à l'aide d'une solution analytique. La seconde implique l'utilisation d'un algorithme itératif de type descente du gradient afin de s'approcher de la solution optimale. Dans le cas où $\arg\min_{\theta} \mathbb{E}_{D_{\text{train}}} [\mathcal{L}(f(x, \theta), y)]$ n'est pas un problème convexe, comme c'est le cas avec les réseaux neuronaux, il n'existe alors pas de solution ana-

Algorithme 1 : Algorithme général de descente du gradient.

Entrées : $f(x)$: fonction à optimiser;
^0x : Poids initiales;
η : taux d'apprentissage;
ϵ : critère d'arrêt;
K : nombre d'itération maximale;

Initialiser $k \leftarrow 0$;
Initialiser $^kx \leftarrow {}^0x$;
Initialiser $Arrêt \leftarrow \epsilon\left(f(^kx)\right)$;
tant que $\neg Arrêt$ **et** $k < K$ **faire**
  $\nabla_x f \leftarrow \nabla_x f(^kx)$;
  $^{k+1}x \leftarrow {}^kx - \eta \cdot \nabla_x f$;
  $k \leftarrow k + 1$;
fin
Retourner ^kx

lytique et la solution itérative devient la seule option.

Algorithme de descente du gradient : Dans le cadre de l'apprentissage machine et des réseaux neuronaux, les algorithmes itératifs qui sont les plus souvent utilisés sont des algorithmes de descente et plus précisément des algorithmes de descente du gradient. L'algorithme de descente du gradient général est relativement simple. Prenons une fonction $f(x)$ que nous souhaitons minimiser par rapport à x, soit $\arg\min_x f(x)$. La stratégie consiste brièvement à partir d'un point initial ^0x et de modifier cette valeur à chaque itération en nous basant sur une direction de descente et en faisant un pas de grandeur η dans cette direction jusqu'à ce que l'on dépasse un nombre K d'itération ou que les conditions pour le critère d'arrêt ϵ soient atteintes par $f(^kx)$. La plupart du temps, le critère d'arrêt utilisé est $\|\nabla_x f(^kx)\| < \epsilon_0$ où ϵ_0 représente une valeur très près de 0. Ce critère permet de vérifier si ^kx est un point critique. L'algorithme 1 donne les détails de la descente du gradient générale.

Dans le cas de l'apprentissage automatique, les choses sont un peu différentes. Pour commencer, nous tenterons d'optimiser notre modèle $f(x, \theta)$ par rapport aux

poids du modèle θ en trouvant la solution au problème suivant :

$$\arg\min_{\theta} \mathbb{E}_{D_{\text{train}}} \left[\mathcal{L} \left(f(x, \theta), y \right) \right] = \arg\min_{\theta} \frac{1}{N} \sum_{(x_i, y_i) \in D_{\text{train}}} \mathcal{L} \left(f(x_i, \theta), y_i \right).$$

La fonction ne dépend donc pas seulement de θ, mais aussi des données d'entraînement (x, y). Ensuite, afin d'éviter le problème de sur-apprentissage, qui sera discuté à la section 2.1.4, les données d'entraînement seront séparées en m lots et il y aura donc deux boucles dans l'algorithme. Pour éviter toute forme de confusion, nous nommerons cycle ou cycle d'entraînement (en anglais *epoch*) plutôt qu'itération, l'occurrence

Algorithme 2 : Algorithme de descente du gradient pour l'apprentissage machine.

Entrées : $\mathcal{L}$: fonction de coût;
f : modèle d'apprentissage;
${}^0\theta$: Poids initiales;
D_{train} : Ensemble d'entrainement;
D_{valid} : Ensemble de validation;
η : taux d'apprentissage;
ϵ : critère d'arrêt;
N : nombre de cycle d'entrainement maximum;

Initialiser $n \leftarrow 0$;
Initialiser $k \leftarrow 0$;
Initialiser ${}^k\theta \leftarrow {}^0\theta$;
Initialiser $\mathit{Arrêt} \leftarrow \epsilon \left(\mathbb{E}_{D_{\text{valid}}} \left[\mathcal{L} \left(f(x, {}^k\theta), y \right) \right] \right)$;
tant que $\neg \mathit{Arrêt}$ **et** $n < N$ **faire**
  $\{\mathcal{S}\}_{i=1}^{n} \leftarrow$ Séparer D_{train} en m lot aléatoirement.;
  pour chaque $\mathcal{S}_i \in \{\mathcal{S}\}_{i=1}^{n}$ **faire**
    $\nabla_\theta \mathcal{L} \leftarrow \nabla_\theta \, \mathbb{E}_{\mathcal{S}_i} \left[\mathcal{L} \left(f(x, {}^k\theta), y \right) \right]$;
    ${}^{k+1}\theta \leftarrow {}^k\theta - \eta \cdot \nabla_\theta \mathcal{L}$;
    $k \leftarrow k + 1$;
  fin
  $n \leftarrow n + 1$;
  $\mathit{Arrêt} \leftarrow \epsilon \left(\mathbb{E}_{D_{\text{valid}}} \left[\mathcal{L} \left(f(x, {}^k\theta), y \right) \right] \right)$;
fin
Retourner ${}^k\theta$;

de la première boucle où tous les lots sont utilisés pour entraîner le modèle. Pour finir, le modèle utilisera un ensemble de données de validation lorsqu'il sera évalué par le critère d'arrêt ϵ et celui-ci mesurera très souvent les performances du modèle plutôt que de vérifier si un point critique a été atteint. L'algorithme 2 donne les détails de la descente du gradient pour l'apprentissage automatique.

Pour finir, de nombreuses variantes de l'algorithme de descente du gradient ont été développées pour l'apprentissage automatique au cours des dernières années [38, 47, 48, 70, 82, 152, 156], mais leur description tombe hors de la portée de ce mémoire.

2.1.3 Métriques

En apprentissage automatique, les performances générales d'un modèle sur des données de test sont mesurées à l'aide de métriques. Alors que pour les problèmes de régression l'erreur des moindres carrées, donnée par l'équation

$$\mathbb{E}_{D_{\text{test}}}\left[\mathcal{L}(f(\vec{x},\theta),y)\right] = \frac{1}{N}\sum_{i=1}^{N}(f(\vec{x}_i,\theta)-y_i)^2, \tag{2.1}$$

est une mesure très souvent plus que suffisante, les problèmes de classification peuvent nécessiter davantage de métriques pour mieux comprendre les performances en test [46]. La vaste majorité des métriques utilisées pour ce type de problème sont calculées à partir de ce que l'on nomme la matrice de confusion. C'est en fait un tableau de taille $c \times c$, où l'élément à la position i, j indique le nombre de données de classe i pour lesquelles le modèle a prédit la classe j. La figure 2.1 illustre la forme générale de la matrice de confusion des problèmes de classification binaire.

Dans un problème de classification binaire, les classes un et deux sont remplacées par les étiquettes « positif » et « négatif ». Il existe plusieurs raisons pour lesquelles

Tableau 2.1 – Matrice de confusion pour les problèmes de classification binaire.

		Prédiction	
		Positif	Négatif
Étiquette	Positif	Vrai Positif (VP)	Faux Négatif (FN)
	Négatif	Faux Positif (FP)	Vrai Négatif (VN)

Tableau 2.2 – Liste des différentes mesures utilisées pour les problèmes classification.

Nom	Équation
Taux de vrai positif (TVP)	$\frac{VP}{VP+FN}$
Taux de vrai négatif (TVN)	$\frac{VN}{VN+FP}$
Taux de faux positif (TFP)	$\frac{FP}{FP+VN}$
Taux de faux négatif (TFN)	$\frac{FN}{VP+FN}$
Taux de fausse omission (TFO)	$\frac{FN}{FN+VN}$
Taux de fausse découverte (TFD)	$\frac{FP}{VP+FP}$
Valeur prédictive négative (VPN)	$\frac{VN}{FN+VN}$
Précision (Pr)	$\frac{VP}{VP+FP}$
Justesse	$\frac{VP+VN}{VP+VN+FP+FN}$
Justesse balancée (JB)	$\frac{TVP+TVN}{2}$
Moyenne géométrique des rappels	$\sqrt{TVP * TVN}$
Score F1	$\frac{2VP}{2VP+FP+FN}$
Coefficient de corrélation de Matthews (CCM)	$\sqrt{TVP \times TVN \times Pr \times VPN} - \sqrt{TFN \times TFP \times TFO \times TFD}$

il est nécessaire d'utiliser plusieurs métriques lors de l'évaluation des performances d'un modèle de classification. Prenons par exemple un cas où les classes seraient très déséquilibrées avec 1% des données dans la classe positive et 99% dans celle négative. Un modèle qui prédit l'étiquette négatif à toutes les données aurait une justesse de 99% et un taux de vrai positif de 0%. Ici, une mesure telle que la justesse balancée (voir tableau 2.2) sera plus représentative des performances du modèle, mais il est tout de même nécessaire d'observer le taux de vrai positif pour identifier le problème. D'autres mesures, telles que le coefficient de corrélation de Matthews et le score F1, ont aussi été proposées afin de mieux refléter les performances du modèle.

Surface sous la courbe de la CFR : Dans un problème de classification binaire, la plupart des modèles produiront une prédiction qui indique la probabilité $\mathbb{P}(y_i = 1|x_i)$. Or, une véritable prédiction nécessite de trancher à un certain seuil α, de telle manière que si $\mathbb{P}(y_i = 1|x_i) \geq \alpha$, alors on associera l'étiquette 1 à x_i. Par défaut, on donne à α la valeur de 0.5, mais rien n'empêche de donner une autre valeur. La réduction de la valeur du seuil α a généralement pour effet d'augmenter à la fois le taux de vrai positif et le taux de faux positif. L'augmentation de la valeur de α aura l'effet

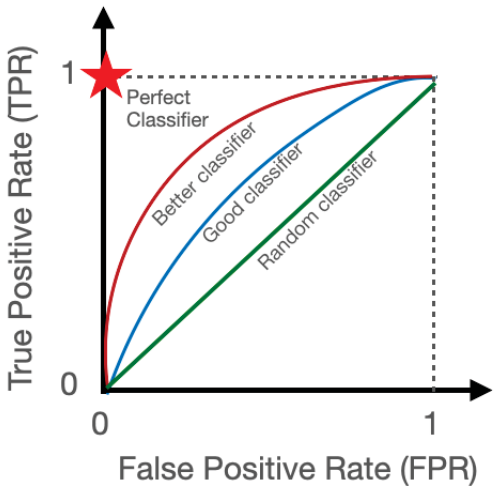


Figure 2.1 – Illustration de la courbe de Caractéristique de fonctionnement du récepteur (CFR). Image tirée de [1].

inverse. Idéalement, un bon modèle aura un taux de vrai positif élevé et un taux de faux positif faible.

Si l'observation de ces deux mesures au seuil optimal α est pertinent, la variation de ces mesures en fonction du seuil est aussi un bon indicateur de performance. Le graphique illustrant les taux positif et négatif pour toutes les valeurs α entre 0 et 1 est nommé la courbe de CFR (voir figure 2.1). Une façon d'interpréter ce graphique est de vérifier si le taux de vrai positif et de faux positif reste bon pour un très grand intervalle de valeur de α. Si c'est le cas, alors le modèle est très confiant dans ses prédictions. Le modèle parfait serait celui qui produirait la probabilité $\mathbb{P}(y_i = 1|x_i) = 1$ pour toutes les données positives et $\mathbb{P}(y_i = 1|x_i) = 0$ pour les négatives. La courbe CFR permet donc de faire la différence entre un bon modèle avec une justesse de 100% (pour un seuil α optimal) et un modèle parfait. Une façon de transformer ce graphique en mesure est de mesurer la surface sous la courbe CFR, que l'on nomme plus simplement SSC.

1. Riccardo Di Sipio, mai 2021, A Quick Guide to AUC-ROC in Machine Learning Models, dans Toward Data Science, retrouvé de https://towardsdatascience.com/a-quick-guide-to-auc-roc-in-machine-learning-models-f0aedb78fbad, consulté le 9 septembre 2022

2.1.4 Sur- et sous-apprentissage

Les deux problèmes les plus courants dans le développement d'un modèle d'apprentissage automatique sont le sur-apprentissage et le sous-apprentissage. C'est avec l'ensemble de validation que nous observons le sur-/sous-apprentissage durant la phase d'entraînement. On parlera de sur-apprentissage lorsque notre modèle obtient de très bons résultats sur l'ensemble d'entraînement, mais de mauvais résultats sur l'ensemble de test/validation. En d'autres mots, le modèle a appris à reconnaître les images d'entraînement, mais n'a pas su généraliser les connaissances apprises aux autres données. Le problème du sous-apprentissage lui se caractérise par de mauvais résultats autant sur l'ensemble d'entraînement que sur celui de test/validation. On pourra donc dire ici que la tâche est trop complexe pour notre modèle. La figure 2.2 illustre visuellement ces problèmes.

Ces deux phénomènes sont liés à ce que l'on appelle la capacité d'un modèle, elle représente le pouvoir d'un modèle d'apprendre des observations qui lui sont fournies.

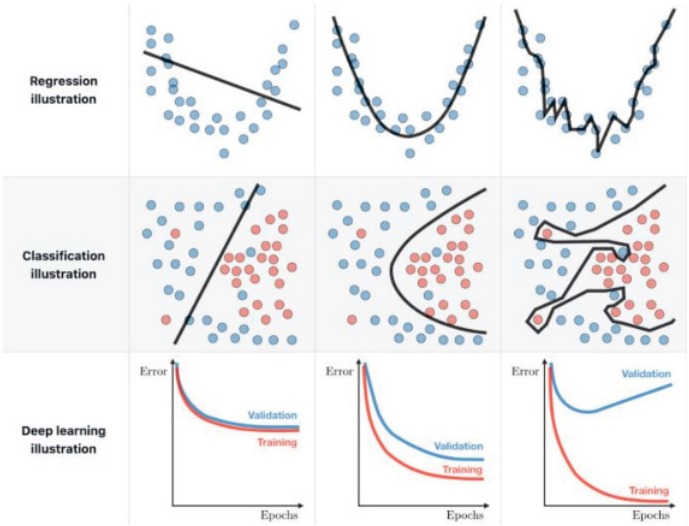


Figure 2.2 – Sur-apprentissage et sous-apprentissage dans différents scénarios. Les colonnes représentent, respectivement, les cas de sous-apprentissage, de scénario optimale et de sur-apprentissage. Image tirée de [2].

2. Amit Chauhan, septembre 2021, Underfitting and Overfitting with Python Examples, dans Towards Ai, retrouvé de https ://pub.towardsai.net/underfitting-and-overfitting-with-python-examples-5a66cb470ebd, consulté le 17 mars 2023

Par exemple, dans le cas d'un réseau de neurones, il est possible d'augmenter la capacité en augmentant le nombre de couches de celui-ci ou le nombre de neurones par couche. Dans le cas du sous-apprentissage, la principale solution est d'augmenter la capacité du modèle. Dans le cas du sur-apprentissage, outre l'idée de réduire la capacité du modèle ou d'augmenter la taille de D_{obs}, il existe deux autres techniques, la régularisation et l'augmentation de données artificielle. Les stratégies de régularisation sont un ensemble de techniques appliquées directement sur les poids du modèle d'apprentissage et qui permettent de limiter la capacité de ce dernier, alors que l'augmentation de données rassemble plusieurs méthodes permettent de créer des données synthétiques à partir des données déjà existantes. La régularisation et l'augmentation de données sont discutées plus en détail à la section A.1 de l'annexe A.

2.1.5 Recherche d'hyperparamètres

Dans le développement d'un modèle d'apprentissage, il y a toujours des paramètres qui ne peuvent pas être déterminés par l'algorithme d'apprentissage lui-même et qui doivent être déterminés autrement. Ce sont les hyperparamètres. Ces derniers contrôlent généralement l'architecture du modèle ou encore le déroulement de la phase d'apprentissage et sont donc essentiels. Le choix du noyau pour le MVS ou encore le taux d'apprentissage des réseaux de neurones ne sont que quelques exemples parmi tant d'autres. Puisque le nombre d'hyperparamètres peut parfois être très élevé et qu'il est difficile de déterminer manuellement la meilleure configuration d'hyperparamètres à utiliser, des algorithmes de recherche d'hyperparamètres ont été développés pour automatiser cette étape. Par souci de concision, nous avons choisi de déplacer les quelques pages décrivant les algorithmes de recherches d'hyperparamètres que nous avons utilisés pour notre recherche dans l'annexe A à la section A.2.

2.2 Algorithme classique

Avant de parler d'apprentissage profond, nous devons passer par les algorithmes d'apprentissage classique qui forment le fondement de l'apprentissage machine. Dans cette section, nous présenterons deux algorithmes d'apprentissage bien connus : la

machine à vecteur de support et le perceptron multi-couches. La compréhension de ces deux algorithmes pourrait nécessiter la lecture de théorie supplémentaire à propos du perceptron et des méthodes à noyau disponible à l'annexe A.

2.2.1 Machine à vecteur de support

Parmi les algorithmes d'apprentissage supervisé classiques encore utilisés aujourd'hui, la machine à vecteur de support (MVS) vient certainement se placer dans les premières places. Ce type de modèle d'apprentissage inventé par Vladimir Vapnik [10, 33] se base sur les méthodes à noyau et peut même être considéré comme une version améliorée de ces derniers. Puisque ce sera le principal modèle que nous utiliserons pour la classification des tumeurs rénales à l'aide des données tabulaires, nous avons cru bon d'écrire quelques lignes sur son fonctionnement.

Tout comme les méthodes à noyaux, les MVS utilise aussi une fonction, souvent non linéaire, pour projeter les données dans un espace ou elles seront linéairement séparable. Elles apportent toutefois trois améliorations principales. La première est l'introduction de la marge de l'hyperplan qui est définie par la distance entre la frontière de décision et la donnée de l'ensemble d'entrainement qui en est le plus près. Ainsi, lorsqu'il y a une infinité de solution différente qui permettent de séparer les données, la marge de l'hyperplan permet de déterminer laquelle de ces solutions est la meilleure.

La seconde est ce que l'on appelle la marge souple, qui permet d'avoir une frontière de décision qui laisse quelques données mal classée. Cet atout est très utile lorsque certaines données aberrante sont présente dans l'ensemble de données d'entrainement. Une pénalité est tout de même appliqué aux données mal classées pour limiter leur nombre. La sévérité de cette pénalité est contrôlé par un hyperparamètre C.

La dernière amélioration importante est le nombre réduit de données d'entrainement que le modèle doit garder en mémoire durant la phase d'inférence. De fait, seul certaines données, que nous nommerons vecteurs de support, sont nécessaire durant la phase d'inférence. Comme le montre la figure 2.3, les vecteurs de supports sont composées des données qui sont mal classées et celles qui se trouve sur la marge de l'hyperplan.

Pour résumés, ces trois améliorations ont pour avantage de déterminer plus efficacement un hyperplan de séparation malgré quelques données aberrantes et d'accélérer la phase d'inférence du modèle en n'utilisant que les vecteurs de support du modèle. Pour garder ce mémoire à l'essentiel, nous référons au lecteur l'article de Dominik Francoeur *Machines à vecteurs de support* [43] pour obtenir tous les détails sur ce modèle.

2.2.2 Réseau de neurones multi-couche

Encore aujourd'hui, la figure la plus emblématique de l'apprentissage profond est le perceptron multi-couche. Il s'agit probablement du premier algorithme d'apprentissage machine à utiliser le concept de couche pour modulariser son architecture. Nous commencerons par aborder la fonction de coût utilisée le plus souvent pour les problèmes de classification utilisant ce modèle avant de présenter l'architecture. Pour finir, nous expliquerons en détail le fonctionnement de la rétro-propagation utilisée pour mettre à jour les poids d'un réseau neuronal.

Entropie Croisé : Bien que la *Hinge Loss* soit encore très utilisée pour l'entrainement des MVS, c'est généralement une autre fonction de coût qui est utilisée lors de l'entrainement des réseaux de neurones pour les tâches de classification. Cette fonction de coût est nommée la perte d'entropie croisée et se base sur la théorie de l'informa-

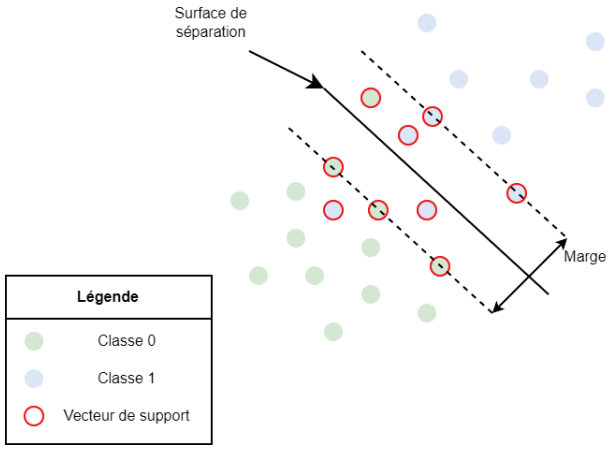


Figure 2.3 – Machine à vecteur de support avec marge souple.

tion. Avant toute chose, il faut voir rapidement ce qu'est l'entropie, la divergence de Kullback-Leiber et l'entropie croisée.

Commençons par poser un espace de probabilité discrète $Y = \{y_1, y_2, ..., y_c\}$ et deux distributions de probabilités sur ce même espace p et q. Nous quantifierons l'information apportée par un évènement y suivant une distribution p à l'aide de $I_p(y)$ défini par l'équation

$$I_p(y) = -\log(p(y)).$$

Cette mesure permet de s'assurer que les événements les plus rares apportent davantage d'information, alors que les plus fréquents en apportent moins. Ensuite, pour une variable aléatoire Y, nous définirons son entropie $H(Y)$, aussi nommée entropie de Shannon, comme étant la quantité d'information moyenne apportée par cette variable. Cette quantité est exprimé à l'aide de l'équation suivante :

$$H(p) = \mathbb{E}_{y\sim p}\left[I_p(y)\right] = -\sum_{i=1}^{c} p(y_i)\log(p(y_i)).$$

Par la suite, il est aussi possible de mesurer la distance entre deux distributions de probabilité sur un même espace à l'aide de la divergence de Kullback-Leiber, donnée par l'équation

$$D_{KL}(p||q) = \mathbb{E}_{y\sim p}\left[\log\frac{p(y)}{q(y)}\right] = \mathbb{E}_{y\sim p}\left[\log(p(y)) - \log(q(y))\right]. \tag{2.2}$$

Cette mesure ne prend la valeur 0 que lorsque les deux distributions sont identiques dans le cas de variables aléatoires discrètes. Comme on le voit avec l'équation 2.2, $D_{KL}(p||q) \neq D_{KL}(q||p)$. L'entropie croisée a le même rôle que la mesure précédente, quantifier la différence entre deux distributions. Ainsi, comme le montre son équation,

$$H(p,q) = H(p) + D_{KL}(p||q) = -\mathbb{E}_{y\sim p}\left[\log(q(y))\right],$$

la distribution q qui minimise l'entropie croisée lorsque p est fixe est aussi celle qui minimise la divergence de Kullback-Leiber puisque $\log(p(y))$ est indépendant de q.

Pour l'entraînement des réseaux neuronaux, l'utilisation de l'entropie croisée permet de mesurer la distance entre les prédictions de notre modèle $f(\vec{x}_i, \theta)$ et celles de

la fonction cible $f^*(\vec{x}_i) = \vec{y}_i$ où $\vec{y}_i$ est le vecteur un parmis n (*one-hot vector*) associé à la donnée $\vec{x}_i$. Dans le cadre d'un problème de classification à c classes, la fonction de perte à minimiser durant l'entraînement est donnée par la formule ci-dessous :

$$\mathbb{E}_{D_{\text{train}}}\left[\mathcal{L}\left(f(\vec{x},\theta),\vec{y}\right)\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_{ij}\log(f(\vec{x}_i,\theta)).$$

Perceptron multi-couches : Comme il sera discuté à la section A.3, le perceptron ne peut fonctionner que pour les données qui sont linéairement séparables. Les méthodes à noyau et les MVS corrigent le problème en utilisant un noyau prédéfini afin de projeter les données dans un espace où elles seront linéairement séparables. Cependant, ces deux algorithmes apportent tout de même un nouveau problème, le choix du noyau. Choisir et/ou définir le noyau optimal pour projeter les données peut être très souvent fastidieux.

L'idée derrière le perceptron multi-couche est donc d'ajouter, avant le classifieur, une ou plusieurs couches de poids qui seront chacune suivies d'une fonction non-linéaire pour que le modèle puisse apprendre à projeter les données dans un espace où elles seront linéairement séparables. Afin de préserver les améliorations apportées par les MVS quant au calcul de l'hyperplan optimal, le perceptron multi-couches utilise une fonction Softmax: $\mathbb{R}^m \to]0,1[^m$ défini par :

$$\text{Softmax}\left(\vec{x}_i\right)_j = \frac{e^{x_{ij}}}{\sum_{k=1}^{m} e^{x_{ik}}}, \tag{2.3}$$

pour imiter une distribution de probabilité et utiliser la fonction de perte d'entropie croisée pour son entraînement. Comme nous l'avons vu plus haut, cette fonction de coût permet au modèle de tolérer des erreurs qui peuvent être dues à des données aberrantes. De plus, la nature de la fonction Softmax qui projettent les données vers l'intervalle $]0,1[^m$, force le modèle à projeter les données le plus loin de la frontière de décision pour minimiser la fonction de perte.

La figure 2.4 donne un exemple d'un perceptron multi-couches à deux couches défini pour un problème de classification à c classes. La fonction de prédiction de ce perceptron multi-couche est donnée par

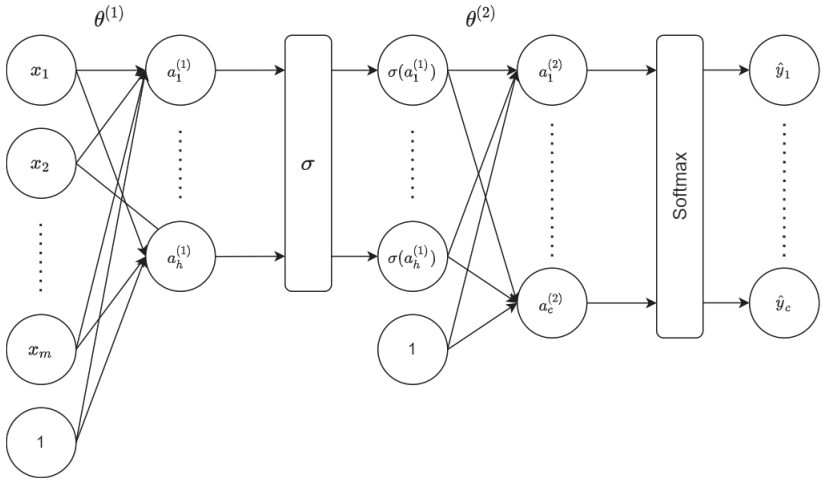


Figure 2.4 – Architecture d'un perceptron à deux couches défini pour un problème de classification à c classes. La première couche utilise une fonction d'activation quelconque nommée σ. $\hat{y}$ représente la prédiction faite par le modèle.

$$\hat{y} = f(\vec{x}, \theta) = \text{Softmax}\left(\theta^{(2)\intercal}\, \sigma(\theta^{(1)\intercal}\vec{x} + \vec{b}^{(1)}) + \vec{b}^{(2)}\right).$$

Pour éviter toute confusion, nous avons renommé les vecteurs de poids associés aux biais $\vec{b}^{(1)}$ et $\vec{b}^{(2)}$. Dans la figure, $\vec{a}^{(1)}$ et $\vec{a}^{(2)}$, qui sont respectivement égale à $\theta^{(1)\intercal}\vec{x}+\vec{b}^{(1)}$ et à $\theta^{(2)\intercal}\vec{a}^{(1)} + \vec{b}^{(2)}$, représentent les couches de neurones du modèle. Dans ce sens, chaque neurone représente une multiplication entre un vecteur d'entrée et un vecteur de poids. Ces couches sont aussi communément appelées couches pleinement connectées, puisque chaque neurone est connecté à chaque élément de la couche précédente.

Rétropropagation : L'avantage des réseaux neuronaux est qu'ils peuvent être décomposés sous forme de couche. Par exemple, prenons un réseau de neurones $f(\vec{x}, \theta)$ que nous décomposerons sous la forme d'une suite de l fonctions $\{g_k\}_{k=1}^{l}$ qui représentent les couches de notre réseau de la première à la dernière. La fonction d'inférence de notre modèle prendra la forme d'une composition de fonction comme le montre l'équation,

$$f(\vec{x}, \theta) = (g_l \circ g_{l-1} \circ \cdots \circ g_1)\,(\vec{x}),$$

où à chaque fonction g_k sera associé des poids ${}^k\theta$. Comme plutôt, ${}^k\theta$ peut représenter une combinaison de scalaires, de vecteurs, de matrices ou même de tenseurs à plus haute dimension.

Ce processus d'inférence qui consiste à appliquer la première couche g_1 à l'entrée $\vec{x}$, pour ensuite appliquer la fonction de la seconde couche g_2 à la sortie de la première et ainsi de suite jusqu'à la dernière couche, est nommé propagation avant. Le mécanisme inverse est nommé rétropropagation et est utilisé pour mettre à jour les poids du modèle. Illustrons cela par un exemple. Prenons notre modèle $f(\vec{x}, \theta)$ et supposons que nous souhaitons dériver la moyenne des pertes sur un lot de données $\mathcal{S} \subset D_{\text{train}}$ par rapport à ${}^i\theta$ donnée

$$\nabla_{{}^i\theta}^{\intercal} \mathbb{E}_{\mathcal{S}} \left[\mathcal{L}(f(\vec{x}, \theta), y) \right] = \sum_{(\vec{x}_i, y_i) \in \mathcal{S}} \frac{\delta}{\delta {}^i\theta} \mathcal{L}(f(\vec{x}_i, \theta), y_i), \tag{2.4}$$

par les poids associés à la couche g_i. Comme le montre les calculs ci-dessous, pour simplifier le calcul du gradient de 2.4 par rapport à ${}^i\theta$, il est possible d'utiliser la dérivée en chaîne. Cependant, si nous voudrions par la suite refaire le calcul du gradient de 2.4, mais cette fois-ci par rapport au poids de la couche j avec $1 \leq j < i$, il faudrait refaire tous les calculs de dérivée en chaîne. L'astuce de la rétropropagation consiste à remarquer que dans la dérivée suivante :

$$\begin{aligned}
\frac{\delta}{\delta {}^i\theta} \mathcal{L}(f(\vec{x}_i, \theta), y_i) &= \frac{\delta}{\delta f(\vec{x}_i, \theta)} \mathcal{L}(f(\vec{x}_i, \theta), y_i) \cdot \frac{\delta}{\delta {}^i\theta} f(\vec{x}_i, \theta) \\
&= \frac{\delta}{\delta f(\vec{x}_i, \theta)} \mathcal{L}(f(\vec{x}_i, \theta), y_i) \cdot \frac{\delta}{\delta {}^i\theta} \left(g_l \circ \cdots \circ g_1 \right) (\vec{x}) \\
&= \frac{\delta}{\delta f(\vec{x}_i, \theta)} \mathcal{L}(f(\vec{x}_i, \theta), y_i) \cdot \frac{\delta}{\delta g_{l-1}} g_l \cdot \cdots \cdot \frac{\delta}{\delta g_i} g_{i+1} \cdot \frac{\delta}{\delta {}^i\theta} \left(g_i \circ \cdots \circ g_1 \right) (\vec{x}),
\end{aligned}$$

à l'exception de la dernière dérivée $\frac{\delta}{\delta {}^i\theta} \left(g_i \circ \cdots \circ g_1 \right) (\vec{x})$, tous les calculs de la dérivée en chaîne par rapport ${}^i\theta$ peuvent être réutilisés pour calculer $\nabla_{{}^j\theta}^{\intercal} \mathbb{E}_{\mathcal{S}} \left[\mathcal{L}(f(\vec{x}, \theta), y) \right]$. La rétropropagation consiste donc à calculer le gradient par rapport à la dernière couche du réseau, suivi de celui de la couche précédente et ainsi de suite jusqu'aux poids de la première couche. Cette façon de procéder permet de réutiliser les calculs effectués et de sauver énormément de temps.

Pour résumer, le réseau de neurones multi-couches permet de remplacer le noyau des MVS, qui doit être choisi manuellement, par des couches de neurones qui seront apprises durant l'entrainement. L'utilisation d'une fonction de coût de type entropie croisée permet quant à elle de préserver les avantages des MVS pour permettre les données aberrantes et pour définir un hyperplan de séparation qui est optimale. Le processus de rétro-propagation permet de calculer la dérivée de la fonction de coût par rapport à chacun des vecteurs/matrices de poids de manière efficace et simple.

2.3 Réseaux de neurones à convolution

Le réseau de neurones à convolution est un type d'architecture spécialisé pour les données pour lesquelles il existe une relation spatiale entre les différents éléments. Développé en 1989 par Yan LeCun pour la classification de chiffre écrit à la main [74], ce type de réseaux neuronaux peut aussi être adapté à d'autres types de données, tels que les données vidéo, les données audio ou encore l'imagerie médicale en 3D [46]. Dans cette section, nous expliquerons le fonctionnement de l'opération de convolution ainsi que les concepts de remplissage, de dilatation et de foulée. Pour finir, nous aborderons le ResNet pour d'une part, donner exemple d'une architecture complète et, d'autre part, présenter le modèle qui nous a servi d'inspiration pour presque toutes les architectures utilisées dans notre recherche. Des explications supplémentaires à propos des couches de mise en commun et de normalisation par lot, disponible à la section A.4 de l'annexe A, pourraient être nécessaires.

2.3.1 Fondement et concept de base

Les réseaux de neurones pleinement connectés souffrent de deux problèmes majeurs lorsqu'ils sont appliqués à des tâches de vision par ordinateur. Pour commencer, la dimensionnalité des images implique qu'il faudra plusieurs milliers de poids ${}^{1}\theta_{i,j}$ pour projeter le vecteur d'entrée dans un espace de plus petite dimension. Ce grand nombre de paramètres pourrait mener à des problèmes de sur-apprentissage tout en limitant la profondeur du réseau en raison de mémoire insuffisante. Ensuite, les couches pleinement connectées ne prennent pas en compte la relation spatiale entre

les différents éléments du vecteur d'entrée.

À la base, les réseaux de neurones à convolution ont été développés pour répondre aux besoins suivants [75] :

1. Limiter le nombre de poids du modèle ;
2. Analyser les pixels dans une petite région pour identifier des structures locales ;
3. Permettre d'identifier ces structures n'importe où dans l'image ;
4. Être invariant aux légères translations et rotations des structures par rapport à l'image.

Convolution : En mathématiques, le produit de corrélation $\star : F(\mathbb{X}) \times F(\mathbb{X}) \to F(\mathbb{X})$ est une opération qui prend deux fonctions définies sur un même domaine $\mathbb{X}$ et produit une troisième fonction sur ce domaine. Cette opération est à la fois commutative, associative et bilinéaire et s'applique autant aux fonctions définies sur des domaines continus que discrets. Les équations 2.5 et 2.6 représentent le produit de convolution pour des fonctions définies sur le domaine des réels $\mathbb{R}$ et des entiers $\mathbb{Z}$ respectivement.

$$(f \star g)(x) = \int_{-\infty}^{\infty} f(x-t)g(t)dt \tag{2.5}$$

$$(f \star g)(h) = \sum_{i=-\infty}^{\infty} f(h-i)g(h) \tag{2.6}$$

Cependant, ce n'est que par abus de langage que le terme convolution est utilisé en intelligence artificielle. En réalité, c'est l'opération de corrélation croisée donnée par

$$(f \star g)(h) = \sum_{i=-\infty}^{\infty} f(i+h)g(i),$$

qui est utilisée [46] et qui, bien que similaire à la convolution, ne possède pas la propriété de commutativité. La corrélation croisée peut servir dans plusieurs domaines comme une mesure de similarité entre deux signaux ou deux vecteurs aléatoires. En analyse d'image, la corrélation croisée est plus souvent utilisée pour filtrer une image I à l'aide d'un filtre K plus communément appelé noyau dont la taille correspond le

(a) Image originale (b) Filtre Laplacien (c) Filtre de Sobel

Figure 2.5 – Exemple d'une image filtrée à l'aide de différents noyaux.

plus souvent à la taille du champ récepteur. Le résultat est souvent nommé carte de caractéristiques (*features map*) dans le domaine des réseaux neuronaux. La taille du noyau est souvent beaucoup plus petite que celle de l'image et de dimension impaire. L'équation

$$(I \star K)(h,l) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} I(i+h, j+l) K(i,j), \tag{2.7}$$

représente la corrélation croisée d'un noyau de taille $(2n+1) \times (2m+1)$ appliqué à une image en ton de gris. Lorsque l'image est en couleur et qu'elle possède trois canaux, il y a deux solutions possibles pour filtrer l'image. 1) Si l'image filtrée doit être en couleur, alors il faut donc trois cartes de caractéristiques qui seront produites en appliquant le même filtre sur chaque canal. 2) Si le résultat doit être en ton de gris, alors le noyau doit être en 3D et posséder autant de canaux que l'image. La figure 2.5 donne un exemple d'une image en couleur à laquelle on a appliqué un filtre Laplacien et un filtre de Sobel.

À partir d'ici et pour la suite de ce mémoire, nous utiliserons le terme convolution pour faire référence à la corrélation croisée comme c'est très souvent le cas dans la littérature.

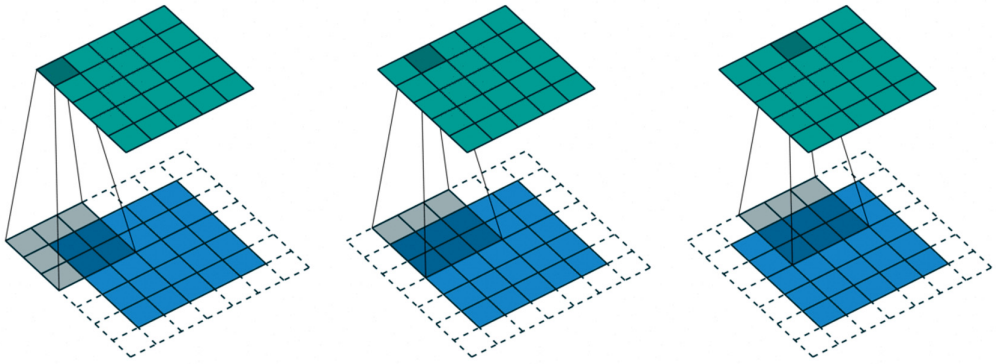


Figure 2.6 – Opération de convolution avec remplissage. La carte en bleu représente l'image, les carrés en pointillés sont les pixels ajoutés et la carte en vert représente le noyau. Ici la zone ombragée sur la carte bleu représente le champ récepteur. Images tirées de [39]

Remplissage, foulée et dilatation : L'opération de convolution telle qu'utilisée en intelligence artificielle présente un peu plus de paramètres que ce qui est décrit plus haut. Les trois principaux paramètres qui n'ont pas été discutés sont le remplissage (*padding*), les foulées (*stride*) et la dilatation.

Pour commencer, nous remarquerons qu'il n'est pas possible de calculer $(I \star K)(0,0)$ avec l'équation 2.7, puisque l'image I n'est pas définie pour les cordonnées $I(-n,-m)$ et ce pour tout $n,m > 0$. Si nous ne faisons rien, il ne sera pas possible de calculer les éléments sur la bordure de l'image et la carte de caractéristiques sera de taille plus petite que l'image originale. Pour obtenir une carte de caractéristiques qui soit de même taille que l'image originale, il faudra ajouter des éléments à la bordure. C'est ce que nous appelons du remplissage. La plupart du temps, ce sont des valeurs 0 qui sont ajoutées à la bordure de l'image, mais plusieurs autres stratégies, comme répéter la valeur du pixel le plus près, existent aussi. Si le noyau est de hauteur et largeur impaires k_1 et k_2 respectivement, alors il faudra ajouter $p_1 = (k_1 - 1)/2$ et $p_2 = (k_2 - 1)/2$ pixels de chaque côté de la bordure au niveau de la hauteur et de la largeur respectivement.

Comme mentionné au début de cette section, la couche de convolution doit aussi avoir pour but d'être insensible aux légères rotations et translations. Pour y arriver, la principale stratégie est de sous-échantillonner l'image et cela peut être fait durant la convolution en augmentant la taille des foulées. La foulée correspond à la taille

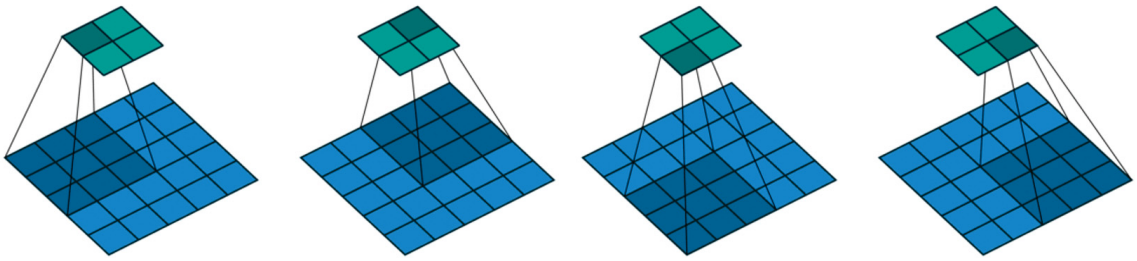


Figure 2.7 – Opération de convolution avec foulée de taille 2. La carte en bleu représente l'image et celle en vert le noyau. Ici la zone ombragée sur la carte bleue représente le champ récepteur. Images tirées de [39]

des pas dans le calcul des cartes de caractéristiques. Une foulée de 1 correspond à une opération de corrélation croisée standard, où en fonction de la gestion des bords, la taille de la carte de caractéristiques sera la même ou presque que celle de l'image originale. Si nous dénotons par h_1 et h_2 la hauteur et la largeur de la carte d'entrée et les foulées verticales et horizontales par s_1 et s_2 respectivement, alors la carte de caractéristiques sera de dimension $f_1 \times f_2$ où $f_i = \lfloor (h_i + 2p_i - (k_i - 1))/s_i \rfloor$.

Un autre point important que doivent remplir ces couches est de limiter le nombre de poids. Or, puisque la taille du noyau correspond à la taille du champ récepteur, augmenter la taille de ce champ revient souvent à augmenter le nombre de poids. Pour pallier ce problème, il est possible d'utiliser le paramètre de dilatation. On pourrait dire que la dilatation correspond à la taille des pas entre deux éléments du noyau. Pour faire simple, si nous dénotons d_1 la dilatation verticale et d_2 la dilatation horizontale, alors la formule de la corrélation croisée est donnée par l'équation

$$(I \star K)(h,l) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} I(i \cdot d_1 + h, j \cdot d_2 + l) K(i,j).$$

De plus, il faudra maintenant ajouter $(k_i - 1) * d_i/2$ pixels de chaque côté de la bordure pour que la carte de caractéristiques soit de même taille que l'image dans le cas où la foulée est de 1. La taille de la carte de caractéristiques sera encore une donnée par f_1 et f_2 où cette fois $f_i = \lfloor (h_i + 2p_i - d_i \cdot (k_i - 1))/s_i \rfloor$.

Couche de convolution : En intelligence artificielle, les couches de convolution représentent un ensemble d'opérations de convolution appliquées à un même signal/i-

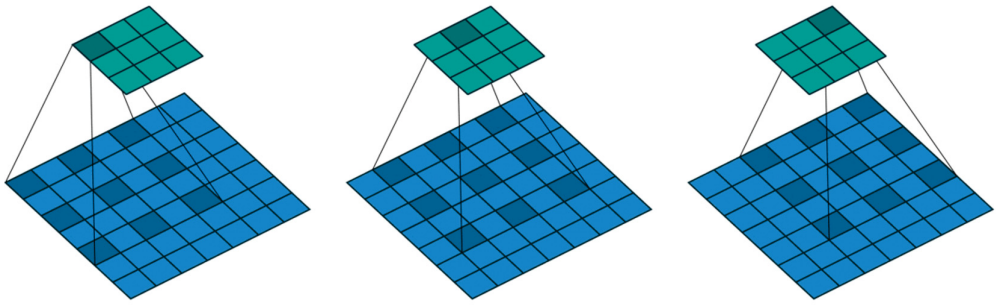


Figure 2.8 – Opération de convolution avec dilatation. La carte en bleu représente l'image et celle en vert le noyau de convolution. Ici la zone ombragée sur la carte bleu délimite le champs récepteur qui lui est de taille 5×5. Images tirées de [39]

mage. De fait, une couche de convolution produira plusieurs cartes de caractéristiques et chacune d'entre elles nécessite un noyau différent. Dans le cas où le réseau de neurones serait appliqué à des images, les noyaux de convolution seront des 3-tenseur (4-tenseur si les images sont en 3D) dont la profondeur sera égale au nombre de canaux de l'image en entrée ou au nombre de cartes de caractéristiques en sortie de la couche précédente. D'autre part, il est possible d'utiliser plusieurs couches de convolution l'une à la suite de l'autre pour augmenter la taille du champ récepteur. Pour finir, les noyaux de convolution seront appris par le modèle et chaque pixel de chaque carte de caractéristiques représentera un neurone.

Pour conclure, les couches de convolutions remplissent les quatre critères énoncés plus haut puisque :

1. Tous les neurones d'une même carte de caractéristiques partagent les mêmes poids qui sont représentés par un noyau, ce qui en plus de limiter le nombre de poids, permet d'identifier une même structure partout dans l'image ;
2. Le champ récepteur de chaque neurone étant limité, cela permet d'identifier des structures locales dans l'image ;
3. Les différentes tailles de foulée permettent au modèle d'être invariant aux légères translations et rotations d'une structure dans l'image.

2.3.2 Couche résiduel et ResNet

De toutes les architectures de réseaux de neurones à convolution qui ont été publiées depuis le début des années 2000, le ResNet [62] est sûrement celle qui a inspiré le plus d'architecture et l'une des plus répandues en classification d'image. On ne compte plus le nombre de variants de ce modèle avec le WideResNet [155], le ResNet de type de type pré-activation [60], le PyramidNet [55] ou encore le ResNext [142] pour n'en nommer que quelques-uns. Facilement adaptable à toutes sortes de problèmes en vision par ordinateur, c'est ce modèle qui nous a servis de base dans la conception des différentes architectures que nous utiliserons dans notre projet. Cette section sera divisée en quatre sous-sections pour aborder les motivations qui ont inspiré ce réseau de neurones, les couches qui le compose, l'architecture globale et un variant du ResNet.

Explosion et disparition du gradient : Avant l'année 2016, deux problèmes très connus affectaient l'entraînement des réseaux de neurones. Ces problèmes étaient l'explosion et la disparition du gradient. Bien que diamétralement opposées, ces complications se produisent lors de la rétropropagation du gradient et elles sont toutes les deux liées à la profondeur du réseau. Comme mentionné dans la section 2.2.2, pour un réseau à l couches, le calcul du gradient des poids ${}^{i}\theta$ de la couche g_i (avec $1 \leq i \leq l$) à l'aide de la dérivée en chaine, implique la multiplication de la dérivée partielle $\frac{\delta}{\delta^{i}\theta} g_i$ avec la dérivée partielle de chacune des couches qui suit par rapport à celle qui l'a précédée $\frac{\delta}{\delta g_{j-1}} g_j$ (avec $i < j \leq l$). C'est dans la multiplication de ces dernières dérivées que résident nos deux problèmes.

Pour commencer, imaginons le cas où $\frac{\delta}{\delta g_{j-1}} g_j >> 1$ pour toutes valeurs de j plus grandes que 1. Alors, le gradient va croître de manière exponentielle au fur et à mesure que celui se rapproche des premières couches ce qui va induire un changement trop important dans la mise à jour des poids du réseau de neurones pouvant ainsi rendre l'entraînement instable, voire même l'arrêter dès les premières itérations si certains poids atteignent la valeur *NaN* (Pas Un Nombre). Ce phénomène qui empêche le modèle de converger se nomme l'explosion du gradient. De nombreuses solutions, telles qu'une meilleure initialisation des poids [45, 61], l'utilisation de couches de normalisation [12, 63, 132, 136] ou encore la coupure de gradient (*gradient clipping*) [46], ont

été proposées. D'autres théories impliquant la souplesse (*smoothness*) de la surface d'optimisation de la fonction de coût ont été avancées plus récemment [121, 154]. Sans aller trop loin dans les détails, il est facile d'imaginer qu'une surface d'optimisation suffisamment hautement non-convexe/chaotique, qui présente des pics élevés et un très grand nombre de minimums locaux, puisse induire un entraînement tout aussi chaotique.

Revenons à la rétropropagation et imaginons que plutôt que d'avoir une série de dérivées supérieures à 0, nous ayons $\frac{\delta}{\delta g_{j-1}} g_j < 1$ pour toutes valeurs de j plus grande que 1. Dans ce cas, le gradient va converger vers 0 très rapidement de la même manière. C'est le problème de la disparition du gradient. À la différence de l'explosion du gradient, cette complication se manifestera plus tard lors de l'entraînement en arrêtant l'apprentissage du modèle. Encore une fois, le fait d'ajouter des couches au modèle ne fait que réduire les performances sur l'ensemble d'entraînement comme sur celui de test [62]. Afin de permettre au gradient de mieux se propager jusqu'aux premières couches du réseau, de nouvelles fonctions d'activation telles que le ReLU [92] ont été proposées. Une autre solution consiste à ajouter des classifieurs auxiliaires à différentes profondeurs du réseau pour injecter davantage de gradient dans les premières couches du réseau. Cette solution adoptée par le GoogLeNet a permis à cette équipe de remporter la compétition *ILSVRC 2014* [119].

Couches résiduelles : Une année après le succès du GoogLeNet et de ses classifieurs auxiliaires, une équipe de Microsoft remporte la compétition *ILSVRC 2015* avec le ResNet [62], une architecture relativement plus simple et plus performante. Le coeur de cette nouvelle architecture réside dans ses nouvelles couches/blocs résiduelles qui forment une solution à la disparition du gradient.

En effet, afin de prévenir le problème de disparition du gradient, les blocs résiduels intègrent des sauts de connexion qui permettent au gradient de garder plus facilement son intensité durant la rétropropagation. De façon simple, l'idée derrière cette nouveauté est que plutôt que d'apprendre la projection optimale $g(x)$, le modèle n'aura qu'à apprendre la carte résiduelle $f(x)$ qui sépare la carte d'entrée de la projection optimale. Ainsi, l'opération peut se résumer comme suit : $g(x) = f(x) + x$. Comme le montre la figure 2.9, la carte résiduelle est produite par un ensemble de plusieurs

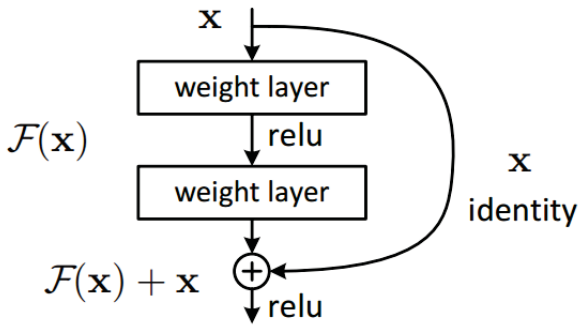


Figure 2.9 – Illustration du concept de couche résiduelle. Image tirée de [62].

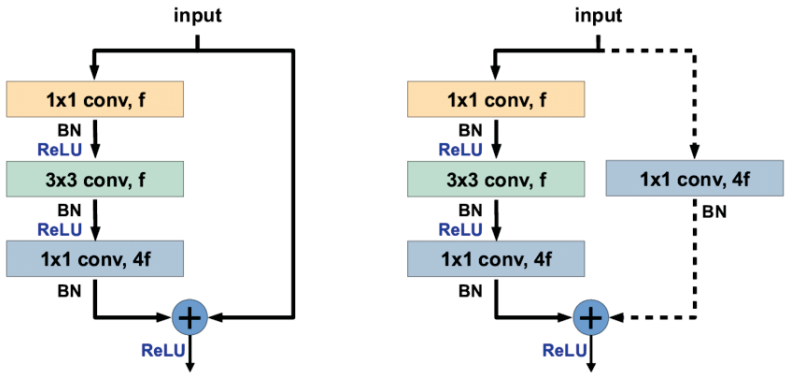


Figure 2.10 – À gauche, un exemple d'un bloc résiduel de type goulot. À droite, un exemple d'un bloc goulot avec sous-échantillonnage. Dans un cas comme dans l'autre, la branche de droite est plus souvent nommée raccourcie. Image tirée de [105].

couches de convolution et une fonction d'activation de type ReLU, d'où pourquoi le terme bloc résiduel est plus souvent utilisé.

Une autre version du bloc résiduel, nommée bloc goulot (*bottleneck block*), a aussi été développée pour les variantes du ResNet qui possèdent plus de couches. En réalité, les véritables blocs résiduels, standard ou goulot, qui composent le ResNet comportent aussi des couches de normalisation par lot avant les $n-1$ premières fonctions d'activation et avant l'addition avec la carte d'entrée. Pour finir, un bloc résiduel peut-être dit avec sous-échantillonnage si la première couche de convolution présente une foulée de 2 plutôt que de 1. Dans ce cas-là, une convolution avec un noyau de taille 1×1 sera appliquée à la carte d'entrée avant l'opération d'addition afin d'ajuster la taille

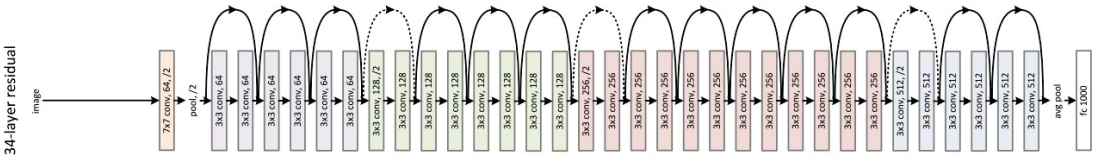


Figure 2.11 – Illustration du ResNet-34 utilisé pour remporter la compétition *ILSVRC 2015*. Les couches d'activations et de normalisation ne sont pas présente dans cette figure, mais elles sont présente dans l'architecture. Image tirée de [62].

des cartes. La figure 2.10 donne un exemple du bloc résiduel de type goulot avec et sans sous-échantillonnage.

ResNet : Dans son ensemble, le ResNet n'est pas seulement composé de blocs résiduels. Pour faire simple, on pourrait décomposer le ResNet en trois sections. La première serait composée d'une couche de convolution, suivie d'une couche de normalisation par lot, d'une couche d'activation et pour finir d'une couche de mise en commun. Dans le cas de la figure 2.11 il s'agit du premier bloc. La seconde section est formée par l'ensemble des blocs résiduels. Par soucis de généralisation et de clarté, cette section est séparée en quatre sous-niveaux qui dans la même figure se distincte par les couleurs mauve, vert, rouge et bleu. À l'exception du premier sous-niveau, le premier bloc de chaque sous-niveau est avec sous-échantillonnage. Le nombre de cartes de caractéristiques est aussi doublé à chacun des changement de sous-niveau. La séparation de cette section en quatre niveaux permet aussi de décrire plus facilement la composition de l'architecture en fonction du nombre couches, en indiquant le nombre de blocs par niveau. Par exemple, la composition du ResNet-18 et du ResNet-34 peuvent être représentées par les vecteurs $[2, 2, 2, 2]$ et $[3, 4, 6, 3]$ respectivement. En général, lorsque l'architecture est composée de plus 50 couches, ce sont des blocs résiduels de type goulot qui sont utilisés. Pour finir, la dernière section est composée d'une couche de type mise en commun moyenne qui va réduire chaque carte de caractéristiques en une carte de taille 1×1. Les cartes résultantes seront ensuite rassemblées pour former un seul vecteur qui sera envoyé à une couche pleinement connectée qui jouera le rôle de classifieur.

ResNet de type pré-activation : Comme mentionné plus haut, le ResNet a été l'inspiration de plusieurs autres architectures. L'une d'entre elles est particulièrement intéressante, car en plus de rester aussi simple et légère, elle a obtenu de meilleurs résultats sur plusieurs jeux de données. Il s'agit du ResNet de type pré-activation [60], une variante du ResNet original qui a été développé par la même équipe que celle qui a créé le ResNet. En bref, en organisant les différentes couches qui composent les blocs résiduels de manière à ce que les couches de normalisation et les fonctions d'activation soient placées avant les couches convolutives, le modèle a moins tendance à souffrir de sur-apprentissage. Ce nouveau type de bloc est nommé bloc résiduel de type pré-activation en raison de l'ordre des couches. La figure 2.12 indique la différence entre les deux types de blocs. Outre le changement de bloc résiduel, l'autre changement important dans cette architecture est qu'il n'y a pas de couche de normalisation et d'activation après la couche de convolution dans la première section.

2.4 Sommaire des avancées en classification des tumeurs rénales

La classification des tumeurs rénales à l'aide de modèle d'apprentissage machine utilisant l'imagerie médicale, est un problème récent dont les premières publications sont en date de 2015 [32, 52]. Cependant, depuis 2019, un nombre d'études si im-

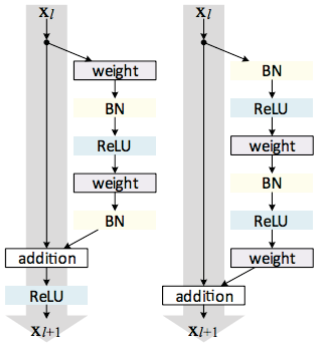


Figure 2.12 – Illustration du bloc résiduel standard (à gauche) et du bloc résiduel de pré-activation. Image tirée de [60].

portant a été fait sur le sujet qu'il nous est impossible de tous les résumer dans un temps raisonnable. Ainsi, nous limiterons nos recherches aux études publiées depuis 2018 et utilisant l'apprentissage profond et des images TC et/ou IRM. Aussi, cette sous-section traitera de la classification de la malignité, du sous-type et du grade seulement. Sans perte de généralité, nous utiliserons le terme justesse pour faire référence à l'*accuracy* ($\frac{VP+VN}{VP+VN+FP+FN}$) et la justesse balancée pour faire référence à $\frac{TPR+TNR}{2}$ dans le cadre d'une classification binaire (voir section 2.1.3 pour plus de détail). Dans le cas, où il y aurait plus de deux classes, la justesse balancée sera calculée en faisant la moyenne de la sensibilité (taux de vrai positif) de chaque classe. Si le lecteur n'a pas été initié à l'apprentissage profond et au transfert d'apprentissage, il est fortement conseillé de se référer respectivement à la section 2 et à l'article de blogue de mwosinski traitant du transfert d'apprentissage[3].

Malignité : En premier, Lee *et al.* utilisaient le transfert d'apprentissage avec un AlexNet préentrainé sur ImageNet [79]. Leurs expériences ont été faites à l'aide d'un ensemble d'images TC avec contraste amélioré regroupant 80 patients et ils ont obtenu une justesse de 76.6%. Coy *et al.* en 2019 [23] et Tanaka *et al.* [128] en 2020 ont procédé de la même manière, mais en utilisant un InceptionV3 et des cohortes de 179 et 159 patients respectivement. Coy *et al.* ont obtenu pour résultats une justesse de 75.4% et 70.6% de justesse balancée, alors que Tanaka *et al.* ont obtenu 88% de justesse et 80% de justesse balancée. Dans une étude regroupant 131 patients, Pan *et al.* [99] utilisaient un modèle multitâche qui effectue la segmentation du rein et de la tumeur rénale et la classification de la malignité à partir d'image TC avec contraste amélioré provenant d'un jeu de données de 131 patients. Leur modèle utilisait pour base un ResNet-50 préentrainé sur ImageNet pour extraire les caractéristiques à partir des tranches 2D de l'image. La moyenne des prédictions de chaque coupe était utilisée pour classifier l'image. Ils ont obtenu une justesse de 100% en test. D'une façon semblable, Zhou *et al.* utilisaient un InceptionV3, préentrainé sur ImageNet, comme base pour extraire des caractéristiques d'une tranche de l'image [163]. Ils utilisaient une couche GRU pour agréger l'information des tranches 2D et prédire la malignité.

3. Disponible à l'url suivant https://medium.com/yosh-ai/transfer-learning-in-a-nutshell-71e1dd0b59da

Ils ont obtenu une justesse de 97%. En 2020 et 2021, Xi *et al.* [146] et Xu *et al.* [145] ont utilisé un ResNet préentrainé sur ImageNet et entrainé sur des images de modalité IRM. Xu *et al.* ont effectué leur étude sur une cohorte de 217 patients et ont obtenu une justesse de 81.8%, alors que Xi *et al.* utilisaient un ensemble de 1162 patients provenant de cinq institutions différentes et ont obtenu 70.0% de justesse.

Sous-type : En 2019, Han *et al.* ont fait usage d'un InceptionNet préentrainé sur ImageNet pour classifier trois sous-types de CCR en utilisant des images TC [53]. À l'aide d'un jeu de données réunissant 169 patients , ils obtiennent en utilisant trois modèles séparés, un pour classifier chaque sous-type, 80.66% de justesse balancée et seulement 45.66% lorsqu'un seul réseau est utilisé pour les trois classes. En 2020, Osowska-Kurczab *et al.* ont entrainé pour la classification de huit sous-types de tumeur rénale [95], à l'aide d'image TC, dix AlexNet préentrainé sur ImageNet. Le modèle d'ensemble résultant a permit d'obtenir 87.1% de justesse, mais obtenaient 89.0% avec une machine à vecteur de support (MVS) entrainée à l'aide de radiomiques texturaux. Leur étude a été faite en utilisant un groupe de 143 patients. Pour finir, Uhm *et al.* ont utilisé les images TC d'une cohorte de 184 patients pour créer un modèle qui fera la segmentation des reins et de la tumeur et la classification parmi cinq sous-types [131]. Chaque patient présente des images de modalité TC 3- ou 4-phases. Le modèle utilisait pour la segmentation est un U-Net 3D, suivi d'un *3D spatial transformer* pour recaler les différentes phases et finalement, c'est un ResNet-101 qui est a été utilisé pour la classification. Ils obtiennent une justesse de 72% et une justesse balancée de 71.98%.

Grade : En 2020, Lin *et al.* ont utilisé des modèles de type ResNet, de profondeur variée (18, 34, 50) et préentrainé sur ImageNet, qu'ils ont entrainé par la suite à l'aide d'image TC 3-phase pour classifier le grade (système de grade Fuhrman) des tumeurs rénales de type CCR et de sous-type cellules claires [86]. Ils ont étudié l'impact du prétraitement des données, en vérifiant les performances avec différentes images rognées de différentes tailles et en appliquant différents niveaux d'attention sur la tumeur à l'aide de la segmentation de cette dernière. Leur meilleur modèle a obtenu une justesse de 73.3%. La même année Zhao *et al.* on utilisé des images de modalité IRM

pour entrainer un modèle concaténant les caractéristiques extraites par deux bases de type ResNet-50 préentrainées sur ImageNet et un perceptron multicouche utilisant des données cliniques telles que l'âge, le sexe et la taille de la tumeur [151]. L'une des deux bases recevait en entré les tranches axial, coronal et sagittal, où le diamètre de la tumeur est le plus grand, de l'image T1 avec contraste amélioré (T1C), alors que l'autre reçoit la même entrée, mais pour la modalité basé sur le contraste T2. Leur étude rassemble une cohorte de 430 patients dont le grade était classifié soit avec le système Fuhrman (353 patients) ou WHO/ISUP (77 patients). Entrainant leur modèle à l'aide des patients classifiés avec le système Fuhrman seulement ils ont obtenu une justesse de 88% pour le système Fuhrman et 83% pour le système WHO/ISUP.

Chapitre 3

Apprentissage multitâche

L'apprentissage multitâche est une branche de l'apprentissage machine qui consiste à utiliser un seul modèle pour effectuer plusieurs tâches simultanément. Le plus souvent, un modèle entrainé à effectuer N tâches apporte plusieurs avantages face à un ensemble de N modèles, qui sont chacun entraînés pour exécuter l'une des N tâches. Pour commencer, lorsque l'architecture multitâche comporte moins de poids et de couches que l'ensemble des N modèles à tâche simple, il en résulte une vitesse d'inférence accrue et une plus petite quantité de mémoires nécessaire. De plus, si les tâches sont suffisamment reliées, le modèle multitâche pourrait présenter une meilleure capacité de généralisation et donc des performances supérieures. Cependant, lors du développment d'un modèle multitâche, il y a beaucoup plus de choses à considéré que pour un modèle à tâche unique. D'abord, alors que l'entropie croisée est maintenant le standard comme choix de fonction de coût pour les modèles à tâche simple, il n'existe pas de tel standard pour les fonctions de coût multitâche. Aussi, il existe des algorithmes d'optimisation qui sont spécifiques aux réseaux neuronaux multitâches et qui sont complémentaires à n'importe quel algorithme de descente [148, 68]. De plus, s'il est parfois difficile de choisir une architecture en temps normal pour des modèles à tâche unique, il faut ici l'adapter à nos différents problèmes. Comme nous le verrons plus bas, il existe différentes façons de procéder. Pour finir, plutôt que de forcer un modèle à exploiter d'hypothétiques relations entre les N tâches, il est parfois plus efficace de simplement déterminer lesquelles doivent être entraînées ensemble à l'aide d'un algorithme de groupement de tâches. Ce chapitre traitera de tous ces problèmes

afin de donner un aperçu global des différentes solutions proposées dans la littérature.

3.1 Fonctionnement et hypothèses

Comme mentionné plus haut, un modèle dispose d'une certaine capacité qui lui permet d'apprendre à extraire des caractéristiques, ou encore à modéliser des variables, qui serviront à prédire l'étiquette $y_i^{(j)}$ associée à la donnée x_i pour la tâche T_j. Parmi les caractéristiques apprises, certaines permettent d'expliquer en partie la distribution Z_j associée à la tâche T_j. Ce sont celles qui permettront de bien généraliser sur l'ensemble de test. Nommons celles-ci les variables explicatives et posons l'ensemble de ces variables $\mathbb{V}_e^{(j)} = \{V_k \mid Z_j | V_k \neq Z_j\}$. Notons toutefois, qu'il est possible que durant l'entraînement, le modèle donne trop d'importance à une variable $V_k \in \mathbb{V}_e^{(j)}$ et qu'il en découle un effet de sur-apprentissage. Nous pourrions expliquer cela par le fait que le modèle ait mal modélisé la relation $Y^{(j)}|V_k$. Le modèle aura aussi tendance à apprendre des caractéristiques qui n'explique pas Z_j, mais qui permettent tout de même de prédire les étiquettes des données d'entraînement. Ce sont ces variables trompeuses qui causent aussi le sur-apprentissage, nommons l'ensemble de ces variables $\mathbb{V}_t^{(j)}$. Il est important de considérer que $\mathbb{V}_t^{(j)}$ n'est qu'un sous-ensemble des variables non explicatives. Puisque la capacité d'un modèle d'apprentissage machine est limitée, nous supposerons qu'il n'est pas possible pour ce dernier d'apprendre à modéliser toutes les variables qui font partie de $\mathbb{V}_e^{(j)} \cup \mathbb{V}_t^{(j)}$. En gardant en tête que le modèle est un réseau de neurones et qu'il est entraîné à l'aide d'un algorithme de type descentes du gradient, nous pouvons supposer qu'à chaque itération, il s'intéressera aux variables les plus faciles à apprendre et qui l'aideront le plus à prédire y_i^j.

Dans le cadre de l'apprentissage multitâche, on considère que deux tâches distinctes T_j et T_l sont liées si l'ensemble $\mathbb{V}_e^{(j,l)} = \mathbb{V}_e^{(j)} \cap \mathbb{V}_e^{(l)} \neq \emptyset$. En d'autres termes, elles sont liées si elles partagent des variables explicatives communes. Bien sûr, plus $\mathbb{V}_e^{(j,l)}$ se rapproche de $\mathbb{V}_e^{(j)} \cup \mathbb{V}_e^{(l)}$ et plus les tâches sont liées. D'un autre côté, rien ne dit que $\mathbb{V}_t^{(j,l)} = \mathbb{V}_t^{(j)} \cap \mathbb{V}_t^{(l)}$ est vide. L'hypothèse raisonnable de l'apprentissage multitâche est que $\mathbb{V}_e^{(j,l)}$ soit prédominant sur $\mathbb{V}_t^{(j,l)}$. C'est-à-dire, qu'une variable qui est pertinente pour deux tâches a plus de chance d'être explicative que trompeuse. Ainsi, il serait possible, en entraînant un modèle sur plusieurs tâches à la fois, d'encoura-

ger ce dernier à apprendre davantage de variables explicatives et moins de variables trompeuses. Plusieurs hypothèses permettent d'expliquer comment cet effet peut se produire ou comment l'apprentissage multitâche peut créer un effet de régularisation.

Écoute clandestine : L'une des premières hypothèses énoncées par Caruana pour expliquer le fonctionnement de l'apprentissage multitâche est l'écoute clandestine (*eavesdropping*) [17]. Il se résume comme ceci : supposons que nous avons une variable explicative $V_k \in \mathbb{V}_e^{(j,l)}$ qui soit difficile à apprendre pour notre modèle lorsque ce dernier est entraîné à modéliser Z_j mais qu'au contraire elle soit facile à identifier lorsqu'il est entraîné pour Z_l. L'entraînement sur ces deux tâches conjointes permettrait à la tâche T_j de profiter de cette caractéristique plus facilement. Toutefois, pour que cet effet se produise, il faut que V_k soit difficile à apprendre pour l'une des deux tâches et facile pour l'autre. Donc si $\mathbb{V}_e^{(j,l)}$ n'est constitué que de variables qui sont faciles à apprendre et que le modèle les avait identifiées en étant entraîné sur seulement l'une des deux tâches, il n'y a pas d'écoute clandestine qui aura lieu. D'autre part, ce mécanisme peut aussi fonctionner pour les variables trompeuses ou encore pour les variables $V_k \in \mathbb{V}_e^j \cap \mathbb{V}_t^k$ et vice-versa. Ainsi, l'écoute clandestine est un couteau à double tranchant pour les performances de notre modèle.

Amplification de données : Il peut arriver que les tâches T_j et T_l n'utilisent pas exactement les mêmes données en entrée. Prenons par exemple notre étude où nous devons classifier la malignité, le sous-type et le grade d'une tumeur rénale. Alors que la classification de la malignité peut être faite pour toutes les lésions rénales, celles du sous-type et du grade ne concernent que les tumeurs malignes (cancéreuses). L'entraînement sur ces trois tâches conjointement permet au modèle de voir davantage de données que s'il avait été entraîné seulement pour la classification du sous-type et/ou du grade, ce qui a généralement pour effet de réduire le sur-apprentissage. Cependant, l'effet bénéfique de l'amplification de données peut être remis en doute. Si certaines caractéristiques sont plus facilement discernables avec les données associées à la tâche T_j que T_l, il pourrait y avoir un effet d'écoute clandestine intéressant. Or, dans certains cas, comme le nôtre par exemple, on est en droit de se demander : est-ce qu'il y a réellement des informations à extraire des tumeurs bénignes qui peuvent être

pertinentes pour la classification du sous-type et du grade des tumeurs malignes ?

Représentation biaisée : Le concept de représentation biaisée des variables, telle que discuté par Caruana dans sa thèse [17], est assez complexe à décrire et à observer. Pour faire simple, supposons que notre modèle découvre une variable explicative $V_k \in \mathbb{V}_e^{(j,l)}$ en s'entraînant pour la tâche T_j seulement. Il se pourrait qu'il n'arrive pas à représenter V_k correctement, car il la modélise de manière à minimiser la perte $\mathcal{L}_j$ sur les données d'entraînement. C'est ce que nous pourrions appeler une représentation biaisée de la variable V_k. L'hypothèse est donc qu'une représentation de V_k qui minimise au maximum les pertes de plusieurs tâches à la fois aura tendance à être moins biaisée et donc à mieux se généraliser sur les données de test.

Compétition pour les ressources : Le phénomène de compétition pour les ressources est en partie expliqué en gardant le concept de représentation biaisée en tête. Comme mentionné plus haut, un modèle possédant trop de capacité/ressources aura tendance à faire du sur-apprentissage. Cependant, réduire la capacité du modèle pourrait empêcher ce dernier d'apprendre toutes les variables explicatives nécessaires pour modéliser la distribution Z_j associée à la tâche T_j. Ainsi, une façon de régulariser le modèle sans réduire les ressources disponibles à l'exécution d'une tâche T_j est de forcer le modèle à partager une partie de sa capacité pour exécuter une seconde tâche T_l. Dans l'idéal, l'architecture sera conçue de manière à ce que certaines ressources/-neurones soient dédiées à chacune des tâches alors que d'autres seront partagées par les deux tâches. Puisqu'il n'y aura pas suffisamment de capacité dans les ressources partagées pour modéliser toutes les variables $V_k \in \mathbb{V}_e^{(j,l)}$ deux fois (une pour chaque tâche), le modèle sera forcé d'utiliser une même représentation pour les deux tâches et de distribuer judicieusement les ressources restantes. Il y a deux problèmes avec ce phénomène. Premièrement, si les deux tâches ne partagent pas suffisamment de variables $V_k \in \mathbb{V}_e^{(j,l)}$ elles risquent toutes les deux de manquer de capacité et donc de subir les effets du sous-apprentissage. Le deuxième problème, est que si la perte pour une tâche est beaucoup plus importante que l'autre, le modèle pourrait favoriser injustement cette tâche au détriment de l'autre en lui accordant davantage de capacité, ce qui pourrait conduire à du sur-apprentissage pour la première tâche et

du sous-apprentissage pour la seconde. Pour que ce deuxième problème n'ait pas lieu, il est important que les fonctions de perte des deux tâches soient balancées.

3.2 Fonction de coût et influence des tâches sur le gradient

L'une des plus grandes difficultés de l'apprentissage multitâche consiste à poser le problème d'optimisation multi-objectifs. Si chaque tâche possède une fonction de coût dérivable qui permet de mesurer la distance entre la prédiction faite et l'étiquette à prédire, il n'est pas clair de quelle façon il faudrait combiner toutes ces fonctions de perte de manière optimale. En effet, on voudrait trouver une fonction de perte totale qui ne favorise pas une tâche plus qu'une autre. Pour cela, il faudrait que tout au long de l'entraînement, la perte de chaque tâche soit environ de la même amplitude que celles des autres. Aussi, il pourrait être intéressant que le gradient soit influencé de manière à peu près égale par chacune des tâches.

Supposons que l'on entraine un modèle sur un ensemble de tâches $\{T_i\}_{i=1}^m$. La première solution pour combiner les fonctions de perte de toutes ces tâches est de définir la fonction de coût multitâche comme étant une combinaison linéaire de ces fonctions de perte comme suit

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{m} \lambda_i \mathcal{L}_i,$$

où les facteurs λ_i qui servent à la pondération sont des hyperparamètres à déterminer.

Déterminer ces hyperparamètres manuellement serait fastidieux et si on les ajoute à notre recherche d'hyperparamètres cette dernière nécessitera encore plus d'itération pour compenser l'augmentation de la dimensionnalité de l'espace de recherche. En plus, le nombre d'hyperparamètres à ajouter est égal au nombre de tâches. Donc si m est trop grand, il devient pratiquement impossible de déterminer correctement la pondération pour chaque tâche. En outre, ces facteurs sont fixes et ne seront peut-être pas adaptés à tout moment durant l'entraînement. Il faut donc une méthode qui permette d'ajuster automatiquement ces facteurs durant l'entraînement. À notre

connaissance, la première solution présentée fût celle de Kendall *et al.* en 2018 [22], où ils proposent d'utiliser l'incertitude homoscédastique (l'incertitude basée sur les tâches) afin de pondérer les différentes fonctions de pertes. Pour faire simple, l'idée derrière cette fonction de perte multitâche basée sur l'incertitude est de laisser le modèle lui-même balancer les fonctions de perte en remplaçant les facteurs λ_i par $e^{-\alpha_i}$ où α_i est un paramètre qui sera appris par descente du gradient durant l'entraînement. Comme le montre l'équation suivante

$$\mathcal{L}_{total} = \sum_{i=1}^{m} \left(e^{-\alpha_i} \mathcal{L}_i + \frac{\alpha_i}{2} \right), \tag{3.1}$$

une pénalité par rapport aux paramètres α_i est ajoutée afin d'éviter que le modèle ne fasse qu'augmenter la valeur de ces paramètres à l'infini pour réduire la perte totale sans rien apprendre.

Une alternative à la fonction de perte basée sur l'incertitude est la méthode de la pondération dynamique des poids (PDP) (*Dynamic Weight Averaging* : DWA) [81]. Ici, plutôt que de laisser le modèle déterminer les facteurs durant l'entraînement, ces derniers seront calculés à partir de l'équation

$$\lambda_i(t) = K \frac{e^{w_i(t-1)/\tau}}{\sum_{k=1}^{m} e^{w_k(t-1)/\tau}}, \quad w_k(t-1) = \frac{\mathcal{L}_k(t-1)}{\mathcal{L}_k(t-2)},$$

qui mesure la variation de la perte pour chacune des tâches afin de donner plus de poids à la tâche dont la perte a la moins diminuée au dernier cycle t. Dans cette équation, K et τ sont des hyperparamètres qui contrôlent respectivement l'amplitude, la perte totale et la distribution des pertes de chaque tâche. Généralement on initialise $w_k(1) = w_k(2) = 1$ puisque ces valeurs ne peuvent être calculées.

Comme mentionné plus haut, l'objectif ultime est de s'assurer que chaque tâche contribue de manière égale à l'entraînement du modèle. Or, le simple fait de balancer les fonctions de pertes ne garantit pas que les gradients des différentes tâches soient balancés. Afin de corriger cette lacune, l'algorithme GradNorm [18] calculera les facteurs λ_i afin de normaliser les gradients de chacune des tâches. Pour éviter d'entrer trop profondément dans les détails mathématiques, à chaque itération t, le GradNorm va calculer pour chaque perte pondérée $\lambda_i(t)\mathcal{L}_i$ la norme du gradient $G_W^{(i)}(t)$

par rapport à un sous-ensemble de poids partagés W, ainsi qu'un ratio $r_i(t)$ qui mesure à quelle vitesse la tâche T_i s'entraine par rapport aux autres, comme le montre la formule

$$r_i(t) = \frac{\tilde{\mathcal{L}}_i(t)}{\mathbb{E}_{task}\left[\tilde{\mathcal{L}}_k(t)\right]}, \quad \tilde{\mathcal{L}}_k(t) = \frac{\mathcal{L}_k(t)}{\mathcal{L}_k(0)}.$$

Par la suite, on défini la fonction de coût

$$\mathcal{L}_{grad}(t, \lambda_i(t)) = \sum_{i=1}^{m} |G_W^{(i)}(t) - \overline{G}_W(t) \cdot [r_i(t)]^{\alpha}|$$

afin de calculer le facteur $\lambda_i(t)$ qui minimise à la fois, la distance entre les gradients et la différence entre les taux d'apprentissage. α représente ici un hyperparamètre qui contrôle l'équilibre entre le poids de la vitesse d'apprentissage et celui de la distance de chaque gradient $G_W^{(i)}(t)$ par rapport au gradient moyen $\overline{G}_W(t)$.

Outre les méthodes qui tentent de balancer l'influence des différentes tâches sur le gradient, certaines s'attaquent plutôt à un problème que l'on pourrait nommer les gradients conflictuels [148, 68]. En apprentissage multitâche, on parle de gradient conflictuel lorsqu'aux moins deux tâches ont des gradients qui sont opposés sur un sous-ensemble de poids W qu'elles partagent. C'est-à-dire que la similarité cosinus entre le gradient de la tâche T_j et celui de la tâche T_k est négative. Pour corriger ce problème, un premier algorithme d'optimisation nommé PCGrad [148] projette le gradient de chacune des tâches sur le plan normal du gradient de l'autre tâche. Une autre technique nommée Rotograd [68] apparaît comme alternative au PCGrad. À la différence de cette dernière, le Rotograd propose d'insérer une couche de rotation après la dernière couche partagée par les différentes tâches. Elle permet d'appliquer une rotation à la représentation latente du modèle de manière à ce que le minimum local de chacune des tâches ne se retrouve plus en direction opposée par rapport à ceux des autres. Les auteurs de ces deux algorithmes émettent l'hypothèse selon laquelle un conflit de gradient entre deux tâches induira du transfert négatif (une perte de performance due à l'apprentissage multitâche), or cela va à l'encontre de l'hypothèse à propos de la représentation biaisée émise par Caruana en 1998 [17] qui indique que ce conflit pourrait plutôt avoir un effet de régularisation. Pour finir, ces deux méthodes

nécessitent l'utilisation d'un algorithme de descente du gradient quelconque et sont compatibles avec les solutions présentées plus haut, soit la fonction de coût basé sur l'incertitude, la pondération dynamique des poids et le GradNorm.

3.3 Format d'architecture multitâche

En apprentissage multitâche, il ne serait pas faux de dire qu'il n'existe pas d'architecture qui soit aussi populaire et reconnue comme l'est le ResNet, mais plutôt qu'il existe des méthodes populaires pour adapter une architecture à tâche unique à un problème multitâche. Ces adaptations ont pour objectif de permettre aux modèles d'exploiter au mieux les relations qui existent entre les différentes tâches. Cette sous-section présentera donc les formats d'architectures multitâches les plus populaires que nous avons rencontrés dans la littérature, c'est-à-dire le modèle de type partage forcé (*hard-sharing*), celui de type partage incité (*soft-sharing*) [1], le réseau d'attention multitâche (RAM) (*MultiTask Attention Network* : MTAN), en plus du Learning-To-Branch, un algorithme de recherche d'architecture.

3.3.1 Partage forcé

Le format d'architecture partage forcé a longtemps été la seule option proposée. Introduit par Caruana en 1993 [15], c'est avec cette architecture en tête qu'il présente ses hypothèses sur le fonctionnement de l'apprentissage multitâche [17]. De manière très large, toutes architectures multitâches pour laquelle il existe une ou plusieurs couches qui sont partagées par plusieurs tâches peut être considérées comme étant de type partage forcé, puisque ce terme signifie simplement qu'une portion des poids sont entraînés pour minimiser la fonction de perte de chacune des tâches. Or le plus souvent, on parlera d'une architecture partage forcé comme d'un modèle qui prend la forme d'un arbre (voir figure 3.1), où le tronc, qui est composé des n premières couches, est entraîné pour toutes les tâches et où après ces n couches, le modèle se sépare en plusieurs branches qui seront entraînées pour des tâches différentes les

1. Puisqu'il n'existe pas de traduction officielle pour les termes *hard-sharing* et *soft-sharing*, nous avons pris la liberté de choisir une traduction qui soit à la fois significative et qui rende honneur à ces termes

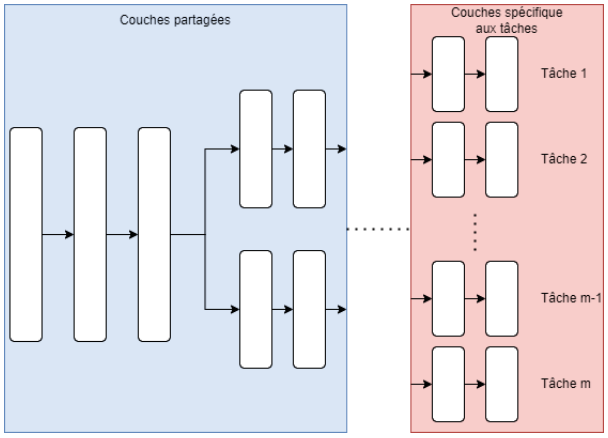


Figure 3.1 – Forme générale d'une architecture multitâche de type partage forcé entraînée pour un ensemble de m tâches.

unes des autres. Bien que ces branches peuvent aussi se séparer par la suite, il faut qu'au final le modèle soit séparé en autant de branches qu'il y a de tâche. Ces dernières branches seront dites spécifiques à une tâche ou spécialisées puisqu'elles ne comportent pas de couche partagée. La façon la plus simple pour adapter une architecture à tâche unique possédant N couches de profondeur en un modèle multitâche est d'utiliser les n premières couches comme tronc et de dupliquer les $N - n$ dernières couches afin de créer m branches qui seront les couches spécialisées. L'avantage de ces modèles est qu'ils sont simples à implémenter, qu'ils sont très personnalisables et qu'ils nécessitent moins de mémoire et de calculs que m modèles indépendants.

3.3.2 Partage incité

Les architectures de type partage incité se différencient de celles de type partage forcé par le fait qu'aucun poids n'est partagé par plusieurs tâches. En fait, dans cette famille d'architectures, chaque tâche à son propre réseau de neurones qui lui est spécifique. Pour profiter de l'apprentissage multitâche, différents mécanismes peuvent être utilisés pour permettre et encourager le partage de connaissance. Par exemple, il est possible de forcer les différents réseaux à avoir des poids qui sont similaires en ajoutant à la fonction de perte totale une pénalité qui est basée sur la distance entre les poids des réseaux [35]. Un autre mécanisme intéressant est l'utilisation de modules

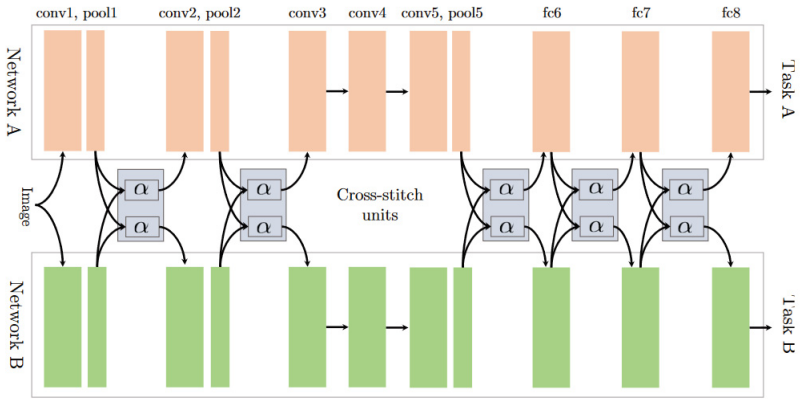


Figure 3.2 – Modèle de type point de croix utilisant deux AlexNet [71] et entrainé sur deux tâches différentes. Image tirée de [91].

de type point de croix (*cross-stitch*) [91]. Comme le montre la figure 3.2, le module de partage connecte les sorties de plusieurs réseaux en effectuant une combinaison linéaire sur les canaux des cartes de sortie. Cela permet à chaque réseau de remplacer l'une de ses cartes de caractéristiques par celle d'un autre réseau. Ils ont pour objectif de permettre aux différents modèles de profiter de l'écoute clandestine sans subir les autres effets de l'apprentissage multitâche. Il est à noter qu'en plus du fait que chaque modèle doit apprendre sa matrice de poids α pour la combinaison linéaire, jamais le gradient de la tâche A ne sera propagé dans le réseau B. Les modules de partage sont apparus plusieurs fois dans le domaine de l'imagerie médicale, que ce soit pour simplement faire de l'apprentissage multitâche [125, 9, 40] ou encore combiner des réseaux utilisant des entrées différentes [137].

3.3.3 Réseau d'attention multitâche

Module d'attention : Les modules d'attention sont basés sur le concept du mécanisme d'attention. Leur rôle est d'analyser une entrée afin d'appliquer un masque à un ensemble de caractéristiques dans le but de mettre en évidence celles qui auront réellement de l'importance pour la prédiction et atténuer les autres. Ils peuvent prendre différentes formes comme le montre les figures 3.3 à 3.5. Le module d'attention sur les canaux [58] (figure 3.3) utilise des couches de mise en commun suivie de couches pleinement connectées afin de créer un vecteur de poids qui multipliera un ensemble

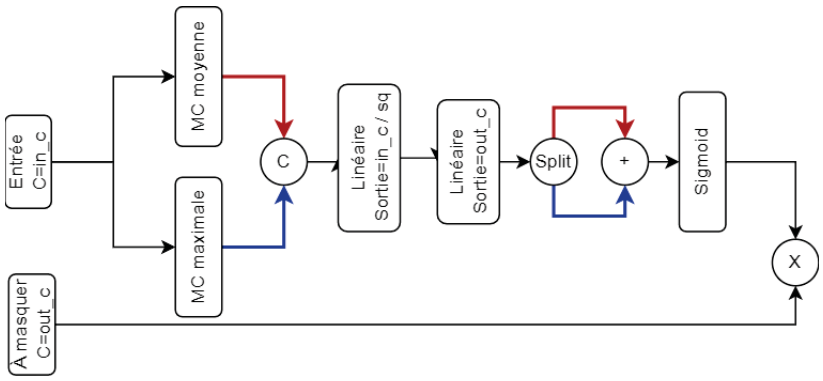


Figure 3.3 – Module d'attention sur les canaux. in_c : nombre de canaux en entrée, out_c : nombre de canaux en sortie, Linear : Couche de type pleinement connectée, sq : taux de compression (*squeeze rate*).

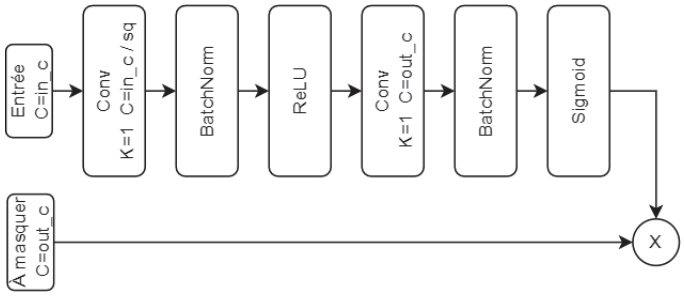


Figure 3.4 – Module d'attention spatial. in_c : nombre de canaux en entrée, out_c : nombre de canaux en sortie, K : taille du noyau de l'opération de convolution.

de cartes de caractéristiques à masquer. De son côté, le module d'attention spatial (figure 3.4) utilise des couches de convolution, de normalisation et d'activation afin de mettre l'emphase sur la région la plus pertinente de chacune des cartes de caractéristiques [141]. Le module de bloc d'attention convolutif (MBAC) (*convolutional attention block module* : CBAM) [26] est simplement la combinaison d'un module d'attention sur les canaux suivis d'un module d'attention spatial.

RAM : Le réseau d'attention multitâche (RAM) est un type d'architecture qui dérive du partage forcé. L'idée est assez simple ; utiliser une architecture à tâche unique, comme un ResNet dont toutes les couches, à l'exception du classifieur, seront partagées par les tâches et ajouter à différent endroit dans l'architecture des modules d'attention qui seront spécialisés. Ces modules d'attention ont pour objectif de per-

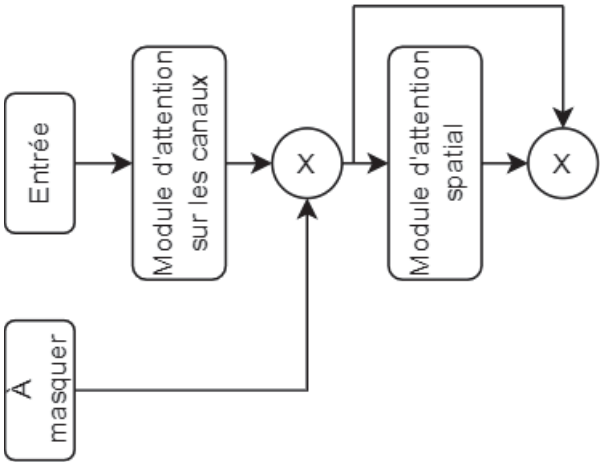


Figure 3.5 – Schéma du module de bloc d'attention convolutif (MBAC).

mettre au modèle d'adapter à la tâche en question la représentation latente produite par les couches du tronc. Il est difficile de décrire davantage la structure du RAM, mais la figure 4.11 de la section 4.3 donne un bon exemple d'un tel modèle adapté à partir d'un ResNet.

3.3.4 Learning-To-Branch

Le Learning-To-Branch (LTB) n'est pas un type d'architecture multitâche en soi, mais plutôt un algorithme de recherche d'architecture pour les modèles de type partage forcé classique. Il permet de définir le branchement optimal pour un ensemble de tâches en un seul entraînement. Pour comprendre le fonctionnement du LTB, il est plus facile de se représenter l'architecture de départ comme étant une série de couches de noeuds, où chaque noeud dans une couche sera nommé enfant et sera connecté à l'ensemble des noeuds de la couche précédente, qui seront nommés parent, à l'aide d'un module de branchement (voir figure 3.6). Chaque noeud peut représenter n'importe quelles combinaisons de couche/bloc : convolution, normalisation, résiduelle, etc. La seule contrainte est que les noeuds d'une même couche doivent tous avoir une sortie qui soit de même dimensionnalité. Durant l'entraînement chaque noeud reçoit en entrée une combinaison linéaire de la sortie des noeuds de la couche précédente à l'aide du module de branchement, qui lui utilise une opération de type gumbel-

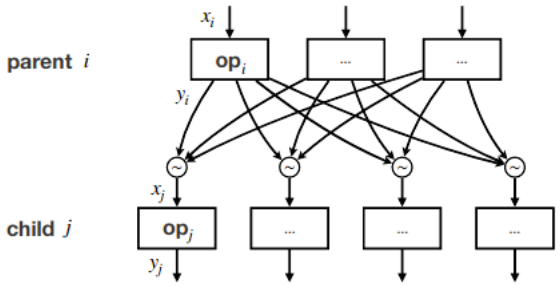


Figure 3.6 – Ici, le module de branchement représenté par les quatre symboles ∼ connecte quatre noeuds enfants à trois noeuds parents durant la phase d'apprentissage en combinant linéairement leur sortie. Durant la phase d'inférence, chaque noeud enfant n'est connecté qu'à un seul parent, celui ayant le plus de poids. Image tirée de [49].

softmax [88, 66]. Le module de branchement possède des poids $\theta_{i,j}$ pour connecter le noeud enfant j au noeud parent i.

Durant l'entrainement le module utilise l'équation

$$\tilde{d}_j = \frac{\exp(\log \theta_{i,j} + \epsilon_i)/\tau}{\sum_k \exp(\log \theta_{k,j} + \epsilon_k)/\tau}$$

pour simuler une fonction softmax et apprendre les poids $\theta_{i,j}$. Dans cette équation ϵ_i est une variable aléatoire suivant une distribution gumbel sur $(0, 1)$ qui va permettre à chaque enfant d'essayer différents parents, alors que τ représente un hyperparamètre de température qui contrôle l'étalement de la distribution du gumbel-softmax. Durant la phase de test, c'est l'équation suivante qui est utilisée afin que chaque noeud enfant j n'ait qu'un seul noeud parent i :

$$d_j = \text{one\_hot}\{\arg\max_i(\log \theta_{i.j})\}. \tag{3.2}$$

Afin d'obtenir de bonne performance avec cet algorithme, il est important de suivre les quatre instructions fournies par les auteurs du LTB [49] :

1. Au départ, les poids des modules de branchement $\theta_{i,j}$ doivent tous être initialisés avec la même valeur ;
2. Il faut attendre quelques cycles avant de commencer l'entraînement des poids dans les modules de branchement ;

3. Au cours de l'entraînement, il est préférable de faire décroître la valeur de l'hyperparamètre τ afin que le modèle ait un comportement moins exploratoire ou moins instable par rapport au changement de noeud parent durant les derniers cycles ;
4. Après avoir entraîné une première fois le modèle, il est important de figer l'architecture trouvée et de réinitialiser les poids du modèle, à l'exception des couches de branchement qui utiliseront désormais l'équation 3.2 autant pour la phase d'inférence que celle de réentraînement.

3.4 Algorithmes de regroupement des tâches

Bien que généralement très performantes, les architectures présentées à la section précédente ne garantissent pas un gain de performance par rapport à l'apprentissage à tâche unique. Cela s'explique simplement par le fait que parfois, certains groupes de tâche ne peuvent profiter de l'apprentissage multitâche lorsque les tâches qui les composent ne partagent pas suffisamment d'affinité. Dans ce cas, il est donc préférable de créer plusieurs sous-ensembles de tâches à l'aide d'un algorithme de sélection de tâches. Cette stratégie peut aussi être utilisée pour sélectionner un sous-ensemble de n tâches auxiliaires parmi un ensemble de N tâches afin d'améliorer les performances au maximum d'une tâche T_j donnée. Cette sous-section traitera donc deux algorithmes de groupement de tâches qui montrent globalement les progrès qui ont été dans ce domaine durant les dernières.

Partage de caractéristiques pleinement adaptatif : Notre premier algorithme permet de créer un ensemble de modèles de type partage forcé classiques optimaux pour un ensemble de tâches donné. C'est le partage de caractéristiques pleinement adaptatif (PCPA) (*fully-adaptive feature sharing*) [83]. Cet algorithme permet en un seul entraînement de créer une architecture en arbre à l'aide d'une statistique qui mesure l'affinité entre les différentes tâches en se basant sur l'hypothèse selon laquelle deux tâches sont reliées si, pour chaque donnée x_i, elles ont environ autant de difficulté ou facilité à prédire y_i. Pour faire simple, au départ on crée un modèle de partage forcé où toutes les couches, à l'exception des classifieurs, sont partagées. Après n

cycles d'entrainement, l'affinité entres les tâches est calculée à l'aide des données d'entraînement afin de séparer les m tâches en g groupes à l'aide d'un algorithme de regroupement quelconque. Une fois les g groupes créés, la dernière couche partagée est dupliquée afin que seules les tâches d'un même groupe puissent la partager. Par la suite, l'entraînement se poursuit pendant encore n cycles avant de mesurer une nouvelle fois l'affinité, mais cette fois à l'intérieur des différents groupes de tâches. Cette mesure est alors encore une fois utilisée pour regrouper des groupes de tâches et diviser le modèle en dupliquant l'avant-dernière couche partagée. Le processus est ainsi répété jusqu'à la première couche, créant plusieurs modèles multitâches entrainés pour effectuer chacun un groupe de tâches différent. L'avantage du PCPA est qu'il joue à la fois le rôle d'un algorithme de recherche d'architecture comme le LTB, en plus de celui d'un algorithme de groupement de tâches.

Groupement de tâches par affinité (GTA) : Le groupement de tâches par affinité (*task-affinity grouping*) [41] est un algorithme publié en octobre 2021 qui, en plus de pouvoir s'adapter à n'importe quelle architecture multitâche, est assez facile à programmer. À la différence du PCPA qui mesure l'affinité entre deux tâches en considérant leur difficulté à classifier les mêmes données, le GTA se base sur une hypothèse beaucoup plus intuitive. Si la rétro-propagation calculée à partir de la perte d'une tâche T_j seulement permet de réduire la perte d'une tâche T_k, alors c'est que la tâche T_k possède une affinité pour la tâche T_j. Bien sûr, la relation d'affinité décrite ici n'est pas une relation symétrique.

L'algorithme GTA fonctionne comme suit ; On commence avec un groupe de m tâches $\{T_j\}_{j=1}^{m}$ que l'on veut séparer en sous-groupe et une architecture multitâche que l'on peut entraîner sur ces m tâches à la fois. Durant l'entraînement, on va calculer l'affinité inter-tâche à toutes les itérations t tel que $t \mod p = 0$, où p est un hyperparamètre qui contrôle la fréquence à laquelle on mesure l'affinité. Ainsi, pour chaque tâche T_j, on va effectuer une copie du modèle pour laquelle on va effectuer une rétro-propagation, en ne considérant que la perte de la tâche T_j afin de mettre à jour les poids partagés $\theta_{s|j}^{t+1}$. Par la suite, pour chaque k tel que $k \neq j$, on va mesurer l'affinité entre T_j et T_k en considérant l'impact sur la perte $\mathcal{L}_k$ qu'a eu la mise à jour

des poids comme le montre l'équation ci-dessous :

$$Z^t_{j\to k} = 1 - \frac{\mathcal{L}_k(x, \theta^{t+1}_{s|j}, \theta^t_k)}{\mathcal{L}_k(x, \theta^t_s, \theta^t_k)}.$$

Ici, $Z^t_{j\to k}$ prend une valeur positive si la tâche T_j a eu un impact bénéfique pour la tâche T_k et négatif dans le cas contraire. Une fois l'entraînement terminé, l'affinité inter-tâche finale entre chaque pair de tâche est calculée à l'aide de l'équation suivante :

$$\tilde{Z}_{j\to k} = \frac{1}{\lfloor T/p \rfloor} \sum_{t=1}^{\lfloor T/p \rfloor} Z^{p\cdot t}_{j\to k}.$$

Ensuite, les tâches sont séparées en g groupes de manière à maximiser l'affinité inter-tâche. Pour finir, il ne reste plus qu'à entrainer les g réseaux de neurones multitâche résultant.

3.5 État de l'art de l'apprentissage multitâche en imagerie médicale

Cette section traitera des différentes applications de l'apprentissage multitâche en imageries médicales. Nous ne résumerons que les cas présentant au moins une tâche de classification et qui utilise l'apprentissage profond.

Pour commencer, Wang *et al.* en 2018 [140], Chen *et al.* en 2019 [19] et Zhou *et al.* en 2021 [150] ont développé chacun un modèle de segmentation et de classification en utilisant une version modifiée du UNet/VNet pour la segmentation et en utilisant les caractéristiques extraites dans l'espace latent de ce modèle[2]. Wang *et al.* ont travaillé sur la segmentation et la classification des os à partir d'échographie 2D, Chen *et al.* se concentraient sur la segmentation de l'artère gauche et sa classification pre/post-ablation. Alors que Zhou *et al.* ont utilisé leur modèle pour la segmentation et la classification des tumeurs à partir d'échographie mammaire 3D.

2. Nous définissons ici l'espace latent d'un modèle UNet ou VNet comme étant la sortie du dernier bloc de l'encodeur et/ou celle du premier bloc du décodeur.

De son côté, Wang *et al.* en 2021 ont créé une architecture pour la segmentation et la classification des lésions pulmonaires pour le diagnostique de la covid-19 à partir d'image TC [137]. Leur modèle complexe utilisait un module point de croix [91], un UNet3D, un décodeur 3D et plus encore. Sinon, Wang *et al.* en 2020 ont développé un modèle semi-supervisé pour la détection du glaucome à partir d'image de tomographie par cohérence optique (TCO) [135]. À la différence des autres, ils ont utilisé des tâches de régression, la prédiction de multiple mesures du champ visuel, comme tâches auxiliaires afin d'augmenter les performances de la tâche de classification.

Du côté des études portant exclusivement sur de multiples tâches de classification, Chen *et al.* en 2019 ont étudié la prédiction de quatre caractéristique de dégénérescences maculaires liées à l'âge à partir d'image de fond de l'oeil [29]. Ils ont utilisé une architecture de type partage forcé qui utilise des blocs dense et des blocs de type InceptionV3. Wang *et al.* qui ont travaillé sur un sujet similaire en 2019, créé un modèle permettant l'identification de 36 maladies de l'oeil différentes pour trois régions [138]. Il se décompose en trois parties : 1) Un YoloV3 qui sert à localiser trois régions du fond de l'oeil (disque du nerf optique, la macula et l'oeil en entier). 2) trois sous-réseaux pour extraire les caractéristiques de chaque région. 3) trois classifieurs utilisant les caractéristiques extraites par un ou plusieurs sous-réseaux pour la détection des maladies de chacune des régions. Pour finir, Goncharov *et al.* en 2021 ont utilisé un UNet3D pour effectuer deux tâches de classification pour des images TC de poumons [50]. La première étant de déterminer si le patient est atteint de la COVID-19 et la deuxième est de quantifier la sévérité. Contrairement aux autres, ils ont utilisé la sortie du UNet pour extraire les caractéristiques utiles à la classification.

Pour conclure, la plupart des études portant sur l'apprentissage multitâche appliqué à l'imagerie médicale s'intéressent davantage à des tâches de segmentation et de classification. Certains auteurs confondent même l'apprentissage multitâche et la segmentation multiclasses [54]. Nous n'avons trouvé que très peu d'études portant exclusivement sur des tâches de classifications.

Chapitre 4

Ensemble de données et méthodologie d'entrainement

Comme mentionné au chapitre 1, notre étude porte sur le développement d'un système de support d'aide à la décision pour de multiples tâches de classification de tumeurs rénales. Plus précisément, notre objectif principal est d'évaluer le potentiel de l'apprentissage multitâche pour prédire la malignité d'une tumeur rénale, ainsi que le sous-type et le grade de celles qui sont malignes. Dans ce chapitre, nous présenterons en quatre sections distinctes la méthodologie mise en place pour mener à bien notre recherche. Nous accorderons nos deux premières sections à l'analyse de l'ensemble de données et à la manière dont nous avons prétraité ces dernières. La section suivante détaillera les modèles utilisés pour nos expériences et expliquera comment nous avons adapté notre architecture à tâche simple pour créer nos modèles multitâches. La dernière section illustrera le déroulement de nos différentes expérimentations.

4.1 Ensemble de données

Pour réaliser ce projet, nous avons à notre disposition les images IRM en pondération T1 amélioré et en pondération T2, avec les tumeurs segmentées, ainsi que les données cliniques d'un ensemble de 1082 patients présentant une tumeur au rein. Les données proviennent de cinq institutions différentes : l'hôpital de l'université de

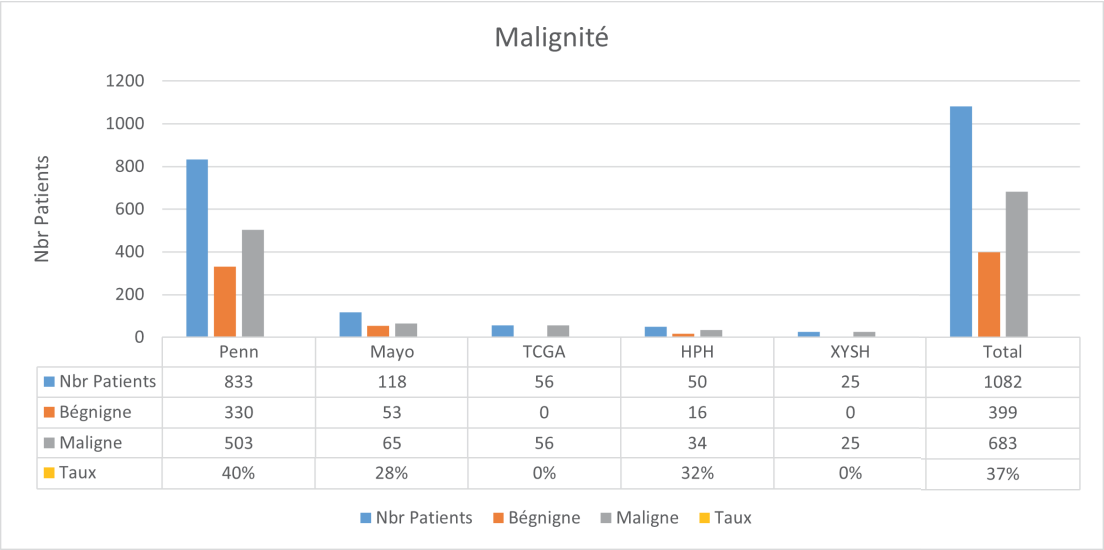
Pennsylvanie (*Penn*), l'hôpital du peuple de la province de Hunan (*HPH*), l'hôpital clinique Mayo (*Mayo*), la seconde hopital de Xiangya de l'université du centre sud (*XYSH*) et l'atlas du génome du cancer (*TCGA*) [1]. Parmi ces données, 1082 pouvaient être utilisées pour la classification de la malignité (lésion bénigne vs lésion maligne) des tumeurs et parmi celles qui sont maligne 599 pour la classification du sous-type (papillaire vs cellules claires) et 593 pour la classification du grade (bas vs élevé). Les graphiques 4.1 à 4.3 montrent la distribution des données en fonction de leur classe et de leur institution.

Afin de mesurer les performances de nos différents modèles, nous avons séparé les données en trois sous-ensembles. Le premier ensemble de test formé à l'aide d'un échantillonnage stratifié de 10% parmi les données. Cet ensemble, que nous nommerons l'ensemble de retenue ou l'ensemble de test final, ne sera utilisé qu'à la fin de l'étude afin de valider nos résultats. Ensuite, un autre ensemble de test stratifié contenant 20% des données restantes et finalement un ensemble d'entraînement. Les données sont stratifiées par rapport à leur étiquette de classe pour chaque classe (malignité, sous-type et grade), afin que les proportions de chaque classe soient environ les mêmes dans chacun des ensembles, et ce pour chaque tâche.

4.2 Prétraitement des données

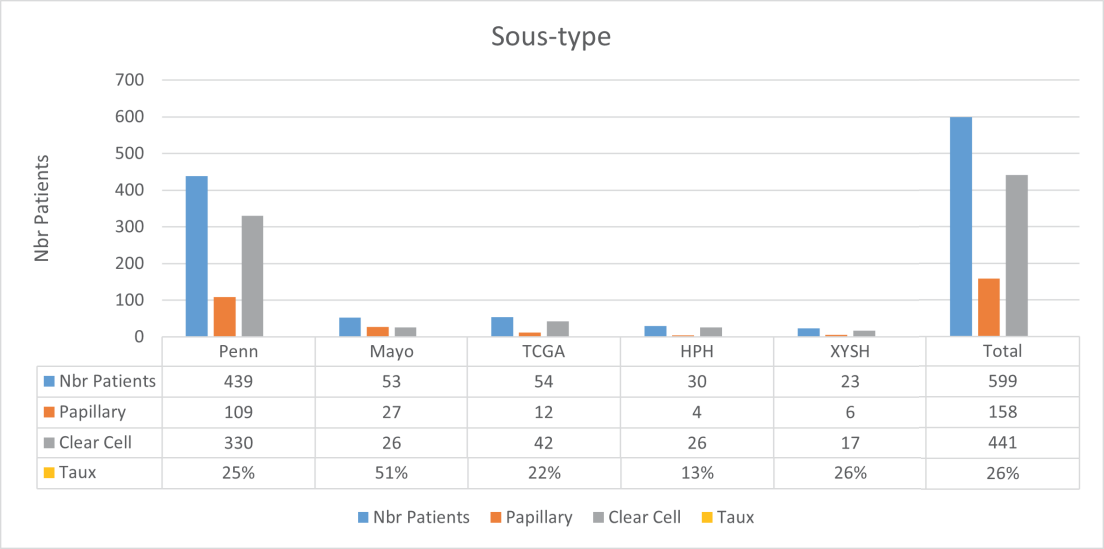
Dans la section qui suit, nous présenterons les détails concernant le prétraitement des données tabulaires et imagées, l'extraction des caractéristiques radiomiques, ainsi que les méthodes d'augmentation de données qui ont été utilisées durant l'entrainement.

1. Les acronymes anglophone sont ici utilisée pour simplifier la tâche des lecteurs qui auront reçu l'autorisation d'accéder aux données et qui souhaitent inspecter notre jeux de données après le prétraité. Les noms anglais sont dans l'ordre : *Hospital of the University of pennsylvania* (*Penn*), *Poeple's Hospital Of Hunan Province* (*HPH*), *Mayo Clinic Hospital* (*Mayo*), *The Second Xiangya Hospital Of Central South University* (XYSH) et *The Cancer Genome Atlas Program* (*TCGA*).



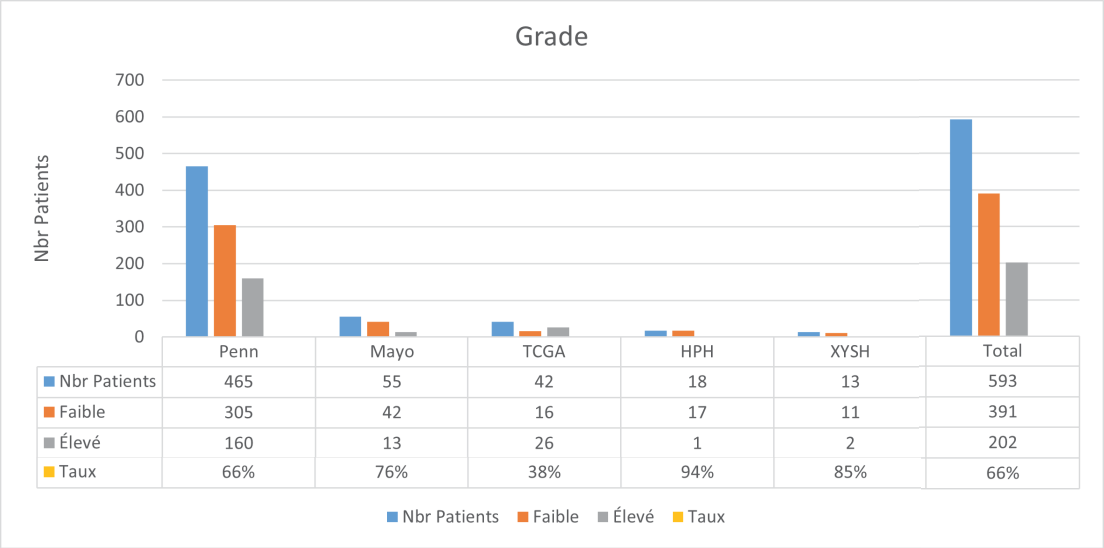
	Penn	Mayo	TCGA	HPH	XYSH	Total
Nbr Patients	833	118	56	50	25	1082
Bégnigne	330	53	0	16	0	399
Maligne	503	65	56	34	25	683
Taux	40%	28%	0%	32%	0%	37%

Figure 4.1 – Distribution des patients en fonction de leur institution et de la malignité de leur tumeur.



	Penn	Mayo	TCGA	HPH	XYSH	Total
Nbr Patients	439	53	54	30	23	599
Papillary	109	27	12	4	6	158
Clear Cell	330	26	42	26	17	441
Taux	25%	51%	22%	13%	26%	26%

Figure 4.2 – Distribution des patients en fonction de leur institution et du sous-type de leur tumeur.



	Penn	Mayo	TCGA	HPH	XYSH	Total
Nbr Patients	465	55	42	18	13	593
Faible	305	42	16	17	11	391
Élevé	160	13	26	1	2	202
Taux	66%	76%	38%	94%	85%	66%

Figure 4.3 – Distribution des patients en fonction de leur institution et du grade de la tumeur.

4.2.1 Analyse et traitement des données cliniques

A ce niveau, le prétraitement des données a été effectué en deux parties : la première partie concerne les données cliniques et la seconde les images IRM. Pour les données cliniques, nous avons d'abord changé les variables catégoriques en variable numériques à l'aide de l'encodage 1 parmis n (*one-hot encoding*). Ensuite, nous avons retiré la variable *M stage* puisque pour certains patient il pouvait y avoir deux valeurs différentes. Nous avons aussi retiré la variable *traitement*, car nous croyons qu'en vue des objectifs de notre recherche et d'une utilisation future, cette variable ne devrait pas être disponible lors de l'utilisation clinique de notre modèle. En effet, la finalité même d'un tel projet est de fournir un outil qui aiderait les médecins dans le choix du traitement approprié à un patient donné. Nous concevons donc ainsi notre modèle pour un usage de prétraitement et non de posttraitement.

Ensuite, pour la variable pT, les valeurs supérieures à 3 ont été rassemblées. Pour les variables pN et pT nous avons imputé la valeur 0 aux données inconnues, puisque les valeurs 1 et 2 étaient déjà utilisées. Dans le cas des variables *Métastase*, *VonHippleLindel disease* et *Renal vein invasion*, nous avons aussi imputé la valeur de 0 aux données manquantes. Afin de sélectionner les variables les plus pertinentes, nous avons

mesuré, sur l'ensemble d'apprentissage, le coefficient de corrélation de *Spearman* de chaque variable avec chacune des issues cliniques (malignité, sous-type, grade). Les mesures de corrélation sont notées dans les tableaux 4.1 à 4.3.

À la suite de l'analyse des différents tableaux de corrélations, nous avons décidé de garder seulement les variables *Âge*, *Sexe* et *Taille* pour la classification de la malignité. Pour le problème du sous-type nous avons choisi les variables *Âge*, *Sexe*, *Taille*, *Renal vein invasion*, *Métastase*, pT et pN. Pour la classification du grade, nous avons choisi les variables *Sexe*, *Taille*, *Renal vein invasion*, *VonHippleLindel disease*, *Métastase*, pT et pN. Nous avons fait le choix de ne pas sélectionner la variable *Race*, car cette variable semblait trop fortement corrélée avec les institutions et donc certaines valeurs auraient pu être sous-représentées dans certaines institutions. Pour finir, les variables pN et pT ont été mises sous forme vectorielle 1 parmis n afin de faciliter l'apprentissage de nos modèles.

Tableau 4.1 – Analyse des corrélations entre les variables cliniques et la variable de classification de la malignité.

Variable	Bénigne	Maligne	Spearman	p
Âge (moyenne, écart-type)	55.02 (15.74)	60.39 (11.85)	0.19	****
Sexe			-0.327	****
0	120	406		
1	239	199		
Latéralité			0.029	
0	181	287		
1	178	318		
Emplacement			-0.022	
0	113	211		
1	148	230		
2	98	164		
Taille en mm (moyenne, écart-type)	2.92 (2.11)	4.22 (3.04)	0.225	****

$p < 0.001$: ****, $p < 0.01$: ***, $p < 0.05$: **, $p < 0.1$: *

Tableau 4.2 – Analyse des corrélations entre les variables cliniques et la variable de classification du sous-type

Variable	Papillaire	Cellules claires	Spearman	p
Âge (moyenne, écart-type)	62.73 (10.13)	59.93 (11.79)	-0.108	**
Sexe			0.112	***
0	107	256		
1	32	137		
Latéralité			0.042	
0	72	185		
1	67	208		
Emplacement			-0.082	*
0	44	139		
1	46	158		
2	49	96		
Taille en mm (moyenne, écart-type)	3.69 (2.65)	4.42 (3.02)	0.109	**
Renal vein invasion			0.115	***
0	134	349		
1	5	44		
VonHippleLindel disease			0.052	
0	139	389		
1	0	4		
Métastase			0.124	***
0	139	371		
1	0	22		
pT			0.173	****
0	25	37		
1	105	284		
2	9	72		
pN			0.128	***
0	25	37		
1	109	320		
2	3	30		
3	2	6		

$p < 0.001$: ****, $p < 0.01$: ***, $p < 0.05$: **, $p < 0.1$: *

Emplacement : 0 : bas 1 : centre 2 : haut

Tableau 4.3 – Analyse des corrélations entre les variables cliniques et la variable de classification du grade

Variable	Grade bas	Grade élevé	Spearman	p
Âge (moyenne, écart-type)	60.02 (11.84)	61.77 (10.92)	0.072	
Sexe			-0.102	**
0	225	132		
1	124	45		
Latéralité			-0.002	
0	167	85		
1	182	92		
Emplacement			-0.017	
0	122	61		
1	131	73		
2	96	43		
Taille en mm (moyenne, écart-type)	3.44 (2.50)	5.54 (3.42)	0.331	****
Renal vein invasion			0.28	****
0	339	143		
1	10	34		
VonHippleLindel disease			-0.076	*
0	343	177		
1	6	0		
Métastase			0.184	****
0	344	161		
1	5	16		
pT			0.279	****
0	38	10		
1	289	115		
2	22	52		
pN			0.225	****
0	38	10		
1	300	139		
2	11	22		
3	0	6		

$p < 0.001$: ****, $p < 0.01$: ***, $p < 0.05$: **, $p < 0.1$: *

Emplacement : 0 : bas 1 : centre 2 : haut

4.2.2 Analyse et traitement des images

Pour le prétraitement des images IRM T1C, T2 et de leur masque de segmentation associé, nous avons choisi de rogner les images 3D autour des tumeurs (voir figure 4.4) afin de forcer notre modèle à concentrer son attention sur les tumeurs. Dans le but de redimensionner nos images, nous avons mesuré les dimensions des différentes images et des tumeurs ainsi que la dimension des voxels qui composent ces images. Les résultats de cette analyse sont inscrits dans le tableau 4.4.

Puisque pour 946 des 1086 patients les images étaient orientées en coupe axiale, nous avons choisi de tourner nos images pour les orienter dans la direction R,A,S [2]. Aussi, à la suite de l'analyse des données de ce tableau nous avons redimensionné les voxels pour qu'ils soient de dimensions $(1.03 \times 1.00 \times 2.90)$ mm. La dimension des voxels a été calculée de manière à ce que nous ayons 96x96x32 voxels et cette zone puisse contenir 95% des régions d'intérêt. Le nombre de voxels le long de chaque dimension est dû à une restriction des modèles que nous utilisions et aux résultats de certaines expérimentations préliminaires. De plus, les images ont été centrées sur le centre de masse de la tumeur calculé à partir de leur masque de segmentation

Tableau 4.4 – Analyse des régions d'intérêt

Variable	P_5	P_{95}	Moyenne$_5$	P_{10}	P_{90}	Moyenne$_{10}$
Taille RI (mm)						
x	13.28	98.51	36.28	15.44	76.62	31.52
y	13.28	95.61	35.71	15.30	72.73	34.00
z	11.84	92.74	32.25	15.00	70.08	30.00
Taille RI (voxel)						
x	11	96	33.21	13	73	31.52
y	7	90	30.87	11	67	29.36
z	1.95	34	1.32	2	21.1	6.45
Taille voxel (mm)						
x	0.66	1.56	1.09	0.70	1.56	1.09
y	0.66	3	1.16	0.70	1.64	1.14
z	1.18	94	5.32	2	8.40	5.33

$P_x = x^{\text{e}}$ percentile, $Moyenne_x$ = Moyenne coupée au x^{e} percentile.

2. LeftToRight, PosteriorToAnterior, InferiorToSuperior (standard nibabel)

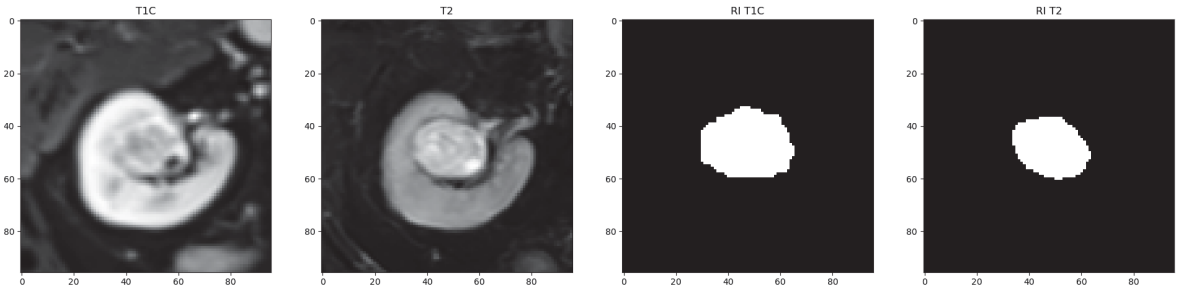


Figure 4.4 – Exemple d'image T1C, T2WI et leur région d'intérêt respective rognées. Pour chaque modalité, la région extraite est centrée au centre de masse de la tumeur défini par le masque de segmentation. L'exemple ci-dessus est une vue en coupe axial.

respectif. Ces deux opérations ont été effectuées à l'aide de la librairie *Antspy*[3]. De plus, les patients dont la tumeur était trop mince et dont la segmentation disparaissait après le rééchantillonnage ont aussi été retirés (deux patients). Pour finir, les images ont été normalisées en soustrayant la moyenne de l'intensité des voxels et en divisant par leur écart-type.

Noter qu'afin de préserver au mieux les textures qui seront par la suite mesurées à l'aide de caractéristiques radiomiques, nous n'avons pas appliqué l'algorithme de correction de biais N4 [126] sur nos images de pondération T1C et de pondération T2 bien qu'il aurait possiblement corrigé le contraste des différents tissus.

Suite à des expérimentations préliminaires (voir annexe B.1.2), nous avons déterminé qu'il était préférable de ne pas fusionner les deux masques de segmentation. Ainsi, les images de ces deux modalités et leurs masques de segmentation respectifs ont plutôt été concaténés le long de l'axe canaux afin de créer une seule image à quatre canaux par patient.

Comme illustré au niveau de la figure 4.5, nous avons retiré les patients dont au moins une image ou un masque était manquant (trois patients). Nous avons aussi retiré les patients dont l'une des images était trop bruitée pour être utilisée (55 patients). Seize patients ont aussi été retirés car la segmentation de la tumeur pour l'image T1C et T2 était inconsistante avant ou après avoir redimensionné les images. 15 autres patients ont été retirés car l'une des images présentait trop de distorsion ou

3. https ://github.com/ANTsX/ANTsPy, consulté le 2 mars 2023

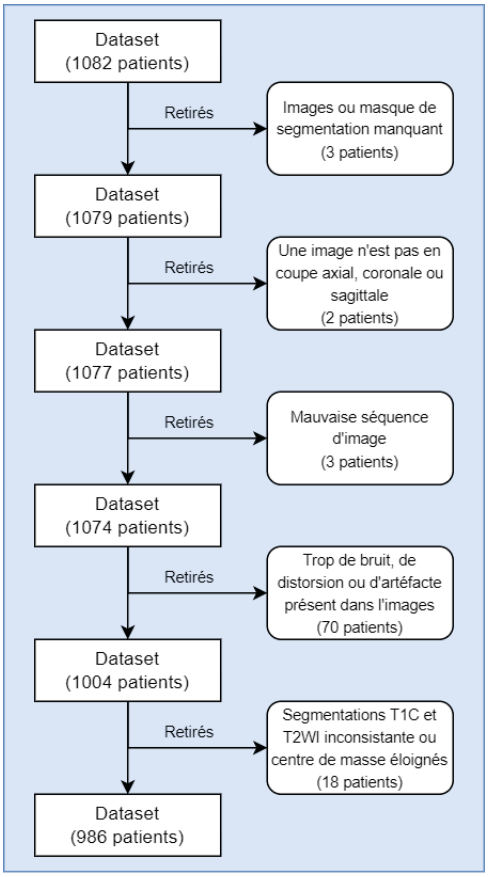


Figure 4.5 – La liste des raisons pour lesquelles un patient peut être retiré de notre jeu de données et le nombre de patients retirés.

des artéfacts trop importants. Quelques images semblaient aussi être dans la mauvaise séquence (trois patients). Les patients dont la distance entre le centre de masse de la tumeur de l'image de pondération T1C et celui de l'image de pondération T2 était trop grande ont aussi été retirés (quatre patients). Nous avons aussi rejeté les patients dont au moins une image n'était pas orientée en coupe axiale, coronale ou sagittale (deux patients). Bien qu'il aurait été possible de corriger ces images, nous avons jugé que notre temps serait mieux investi dans le développement de nos modèles plutôt que dans la récupération de ces deux patients. Au total, il y a 104 patients qui ont été retirés de notre jeu de données.

4.2.3 Extraction des caractéristiques radiomiques

Nous avons extrait les caractéristiques radiomiques à l'aide de la librairie *radiomics-develop* disponible en ligne[4]. Pour calculer les caractéristiques, les voxels ont été redimensionnés pour qu'ils soient de tailles $(1.03 \times 1.00 \times 2.90)$ mm à l'aide d'une interpolation trilinéaire. Pour retirer les valeurs aberrantes durant la resegmentation, nous avons utilisé l'algorithme de Collewet *et al.* [31]. En d'autres termes, toutes les voxels dont l'intensité n'est pas incluse dans l'intervalle $[\mu - 3\sigma, \mu + 3\sigma]$, où μ et σ représentent respectivement la moyenne et l'écart-type des intensités du masque, ont été retirés du masque de segmentation. Pour la discrétisation des intensités, nous avons choisi de fixer le nombre de valeurs possible. Ainsi, pour le calcul des histogrammes d'intensités et des histogrammes d'intensité-volumes, le nombre de niveaux de gris possible était de 64 et de 1000 respectivement, alors que pour le calcul des caractéristiques texturales, ce nombre se limitait à 16.

4.2.4 Augmentation de données

Comme discuté à la section A.1.2, l'augmentation de données est une méthode efficace pour prévenir le sur-apprentissage. Vu le nombre limité de données dont nous disposons, en particulier pour les tâches en lien avec le sous-type et le grade, il nous a paru impossible de ne pas en faire usage lors de nos expérimentations comportant des réseaux neuronaux ou même pour balancer les classes en ce qui concerne le problème de classification du sous-type avec les données tabulaires. Les modèles de type MVS ayant à la base une capacité très limitée, nous n'avons eu besoin que de l'algorithme Adasyn pour générer de nouvelles données d'entraînement que pour la classification du sous-type. Selon quelques expériences, les autres tâches ne semblaient pas profiter de l'augmentation de données avec ce type de modèle.

À l'inverse de ces derniers modèles, les réseaux neuronaux ne sont pas limités en capacité et des méthodes d'augmentation de données plus agressive ont dû être utilisées. Ainsi, pour les images nous avons appliqué les méthodes d'augmentation de données suivantes : retournement aléatoire, changement de l'échelle d'intensité

4. La version utilisée est disponible au lien suivant : https://github.com/mvallieres/radiomics-develop

aléatoire (pour les images T1C et T2 seulement et non les masques), rotation aléatoire, déchirement aléatoire, translation aléatoire, recadrage aléatoire et zoom aléatoire. Lorsque nécessaire, des voxels contenant la valeur 0 ont été ajoutés sur les bords des images afin de respecter les dimensionnalités mentionnées plus haut, soient $96 \times 96 \times 64$ voxels. Les méthodes d'augmentation de données appliquées aux images ont toutes nécessité l'usage de la librairie MONAI (*Medical Open Network for Artificial Intelligence*) [5]

4.3 Modèles employés

Cette section présentera les différentes architectures utilisées lors de nos expérimentations. De plus, nous illustrerons les différents blocs utilisés avant d'énumérer les différentes configurations que nous avons créées pour chaque modèle.

4.3.1 Modèle à tâche unique

Pour les modèles à tâche unique, nous n'avons que deux modèles à présenter. Le premier est une MVS qui nous servira pour les données tabulaires et le second est une version modifiée du ResNet que nous nommerons simplement ResNet3D.

MVS

La machine à vecteur de support telle que décrite à la section 2.2.1 est un algorithme de classification qui utilise des données tabulaires en entrée. Pour l'implémenter, nous avons utilisé la libraire *scikit-learn* [6].

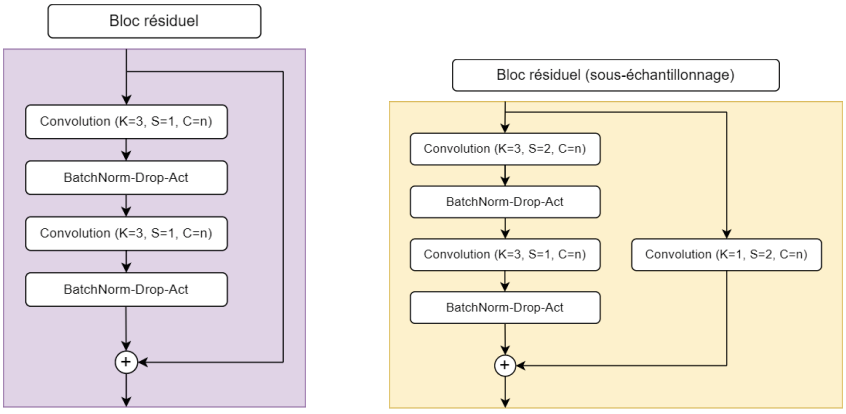
ResNet 3D

Notre principal modèle de réseau de neurones à convolution est le ResNet. Nous avons adapté ce modèle pour en créer une version 3D qui peut utiliser à la fois des blocs résiduels de type post-activation et de type pré-activation. Afin de suivre un standard autant que possible, nous avons implémenté nos blocs résiduels post-activation (voir

5. La version 0.4.0 de MONAI a été utilisé pour effectuer nos expériences.
6. https ://scikit-learn.org/stable/, consulté le 2 mars

figure 4.6) à l'aide de la librairie *MONAI*, dont la formation des blocs résiduels diffère de la version originale. En effet, la dernière couche d'activation est appliquée juste avant l'opération d'addition plutôt qu'après, en plus d'avoir ajouté quelques couches de d'extinction des poids [113]. Pour ce qui est des blocs de type pré-activation (voir figure 4.7), nous les avons implémenté à l'aide de la librairie Pytorch [97]. Comme mentionné à la section 2.3.2, le ResNet est divisible en quatre niveaux qui consistent chacun en une suite de blocs résiduels. Chaque niveau, à l'exception du premier, commence par un bloc de sous-échantillonnage qui réduit la taille des cartes de caractéristiques et double leur nombre. Le nombre de blocs par niveau dépend de la profondeur totale du réseau. Dans le cas du ResNet34, la configuration est la suivante : 3-4-6-3, 3 blocs au niveau 1 ($N_1 = 2$), 4 blocs au niveau 2 ($N_2 = 3$), 5 blocs au niveau 3 ($N_3 = 5$) et 3 blocs au niveau 4 ($N_4 = 2$). La configuration du ResNet18 est 2-2-2-2 ($N_1 = N_2 = N_3 = N_4 = 1$). La figure 4.8 donne un exemple d'un ResNet utilisant seulement des blocs résiduels de type pré-activation.

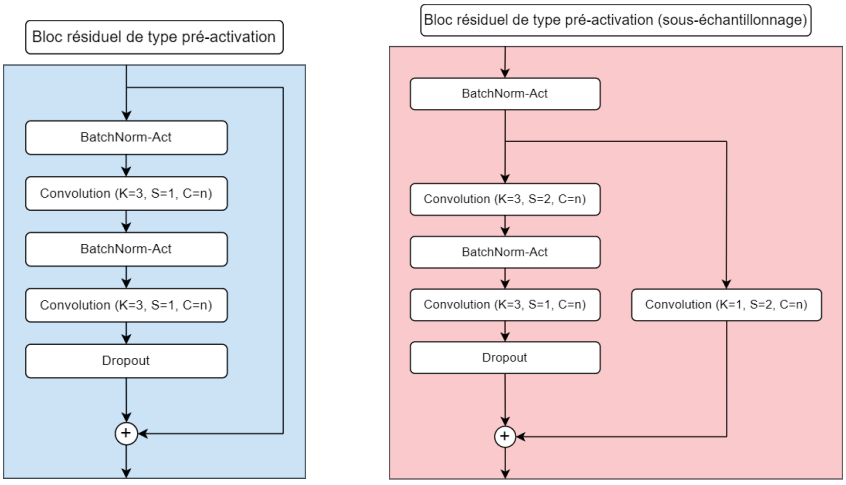
Dans nos expérimentations, nous avons créé quatre configurations différentes de cette architecture en utilisant des combinaisons de blocs différentes. Les configurations 1 et 2 n'utilisent respectivement que des blocs de type pré-activation et post-activation. La configuration suivante est composée de blocs post-activation pour les



(a) Bloc résiduel

(b) Bloc résiduel avec sous-échantillonage

Figure 4.6 – Schémas des blocs résiduels de type post-activation. C : Nombre de canaux en sortie, K : Taille du noyau de convolution, S : foulée.



(a) Bloc résiduel de type pré-activation

(b) bloc résiduel de type pré-activation avec sous-échantillonnage

Figure 4.7 – Schémas des blocs résiduels de type pré-activation. C : Nombre de canaux en sortie, K : Taille du noyau de convolution, S : foulée.

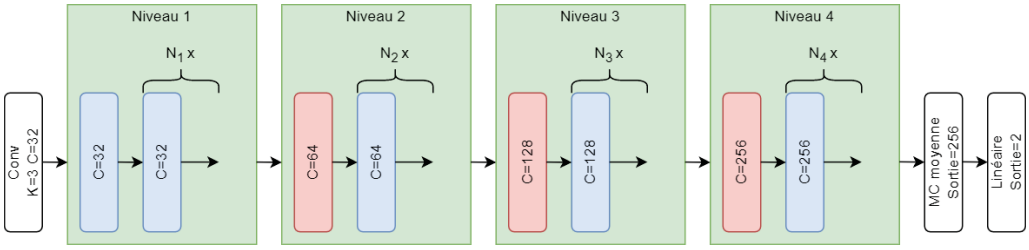


Figure 4.8 – La forme générale du ResNet de type pré-activation. C : Nombre de canaux en sortie, K : Taille du noyau de convolution. MC : Couche de mise en commun. S : foulée .

deux premiers niveaux et de blocs pré-activation pour le reste. À l'inverse de la 3e configuration, la 4e utilise d'abord des blocs pré-activation pour les deux premiers niveaux suivis de blocs post-activation pour les niveaux suivants.

4.3.2 Modèles multitâches

Avant de détailler les caractéristiques de chacune des architectures utilisées pour nos expérimentations portant sur l'apprentissage multitâche, il est important de mentionner quelques détails quant à la fonction de coût multitâche utilisée et les modèles qui seront employés. Afin de garder notre code propre et facile à lire, nous avons choisi d'utiliser une seule fonction de coût multitâche pour tous nos modèles en nous basant sur des expérimentations préliminaires (voir annexe B.2). Étant donné que l'impact du PCGrad sur les résultats était mitigé la plupart du temps et que son utilisation était coûteuse en calcul et nous limitait dans nos futures expériences (voir annexe B.2), nous avons choisi d'utiliser la fonction de coût basée sur l'incertitude des tâches (équation 3.1). Pour les expérimentations impliquant une tâche principale et m tâches auxiliaires la fonction de perte utilisée est

$$\mathcal{L}_{total} = \mathcal{L}_{\text{main}} + \lambda_{\text{aux\_coeff}} \cdot \sum_{i=1}^{m} \left(e^{-\alpha_i} \mathcal{L}_i + \frac{\alpha_i}{2} \right),$$

où $\lambda_{\text{aux\_coeff}}$ est un hyperparamètre qui permet de contrôler l'impact des tâches auxiliaires dans l'entraînement du modèle.

En ce qui a trait aux modèles utilisés pour notre problème de classification multiple, nous avons testé les modèles de type partage forcé, partage incité, RAM et LTB. Pour ce qui est des expérimentations qui portent sur l'utilisation d'une ou plusieurs tâches principales (classification de la malignité, du grade et/ou du sous-type) et de plusieurs tâches auxiliaires (prédiction de caractéristiques radiomiques), seules les architectures de type partage forcé et LTB on pu être testé sans dépasser nos limites matérielles.

Modèles de type partage forcé

Dans le cadre de l'apprentissage multitâche, nous avons défini une première version partage forcé de notre ResNet en nous basant sur les résultats de nos expériences préliminaires (voir annexe B.1.1). Ces expérimentations faites avec des modèles à tâche unique ont montré qu'il était préférable d'utiliser la configuration 1 (seulement des blocs de type pré-activation) pour la classification de la malignité et du grade, alors que pour la classification du sous-type, la configuration 4 (pré-act x2 - post-act x2) obtenait les résultats les plus favorables. Comme le montre la figure 4.9, notre modèle multitâche tente donc de respecter au maximum ces contraintes. Cependant, lorsque les trois premiers niveaux sont partagés par toutes les tâches, la configuration 4 sera favorisée si le modèle est entraîné pour la classification du sous-type. C'est-à-dire que le sous-réseau utilisé pour la classification de la malignité pourrait avoir la forme pré-actx2 - post-act - pré-act. Toutefois, lorsque le modèle multitâche n'est entraîné que pour la classification de la malignité et du grade, l'architecture sera inspirée de la configuration 1 du modèle à tâche unique.

Nous avons également défini une autre version en nous inspirant des résultats de la recherche d'hyperparamètres des modèles à tache unique. Ainsi, l'architecture multitâche est inspirée de la configuration 3 du ResNet3D (post-act x2 - pré-act x2). Nous avons nommé ces deux architectures les configurations A et B respectivement. Pour les expérimentations qui incluent des tâches auxiliaires, la configuration A n'est pas disponible.

Pour finir, d'autres expérimentations nous ont permis de découvrir que la classification du grade pouvait nécessiter une architecture plus profonde. Ainsi, nous avons créé une architecture multitâche à profondeur mixte pour ce modèle, c'est-à-dire que les niveaux utilisés spécifiquement pour la classification du grade sont formés de manière à imiter une architecture à 34 couches de profondeur alors que les niveaux partagés et ceux qui sont utilisés par les autres tâches imitent une configuration à 18 couches de profondeur.

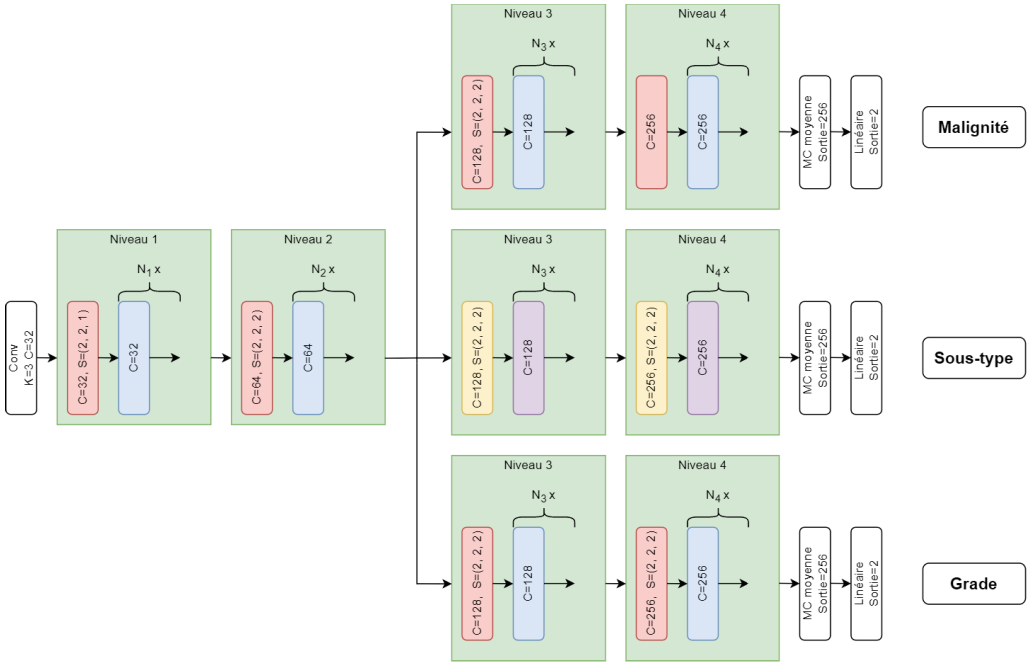


Figure 4.9 – La forme générale du modèle de type partage forcé adapté du ResNet. C : Nombre de canaux en sortie, K : Taille du noyau de convolution. MC : Couche de mise en commun. S : foulée.

Modèles de type partage incité

Comme mentionné plus haut, nous avons choisi d'utiliser les modules de partage présent dans le modèle point de croix (*Cross-Stitch*) [91] pour expérimenter les modèles de partage incité. Des solutions différentes comme les modules de type écluse (*Sluice*) [100] auraient pu être utilisées, mais par souci de simplicité, nous avons préféré expérimenter seulement le premier type de module. L'idée derrière cette couche est de permettre au modèle de faire travailler ensemble différents sous-réseaux en combinant leur représentation latente à l'aide d'une combinaison linéaire sur les canaux, c'est-à-dire qu'elle transforme le j^e canal du réseau i en combinant les n^e canaux de chaque réseau ensemble. En termes plus mathématiques, soit le module de partage $P = \{p_{ijk}\}_{i,k\in\{1,\dots,m\},j\in\{0,\dots,n\}}$ (m = nombre de tâches et n = nombre de canaux) et $x_{ij}^{(l)}$ la carte de caractéristique j en sortie du sous-réseau i à la couche l, si le module de partage se trouve après la couche l, nous aurons que $x_{ij}^{(l^*)} = \sum_{k=1}^{m} p_{ijk} \cdot x_{kj}^{(l)}$. N'ayant pas trouvé dans la littérature de méthode formelle pour initialiser les poids du tenseur

P, nous avons choisi d'initialiser les poids selon une distribution bimodale :

$$p_{ijk} \sim \begin{cases} \delta[c] & \text{if } i = k \\ N\left(\frac{1-c}{m-1}, \frac{d}{m-1}\right) & \text{else} \end{cases} . \tag{4.1}$$

Dans cette équation, c est le paramètre de conservation, d le paramètre de dispersion, $N(\mu, \sigma^2)$ une distribution gaussienne de moyenne μ et de variance σ^2 et $\delta[c]$ représente la distribution de Dirac centrée en c.

Comme pour le partage forcé, nous nous sommes inspirés de la recherche d'hyperparamètres des modèles à tâche unique afin de réduire le nombre d'hyperparamètres. C'est-à-dire que les trois sous-réseaux utilisés correspondent à la configuration 3 du ResNet3D (post-act x2 - pré-act x2). De plus, ce type d'architecture permet l'utilisation de sous-réseaux de profondeur différente. Donc, la profondeur mixte signifie ici que le sous-réseau utilisé pour la classification du grade possède 34 couches de profondeurs alors que les autres n'en possèdent que 18.

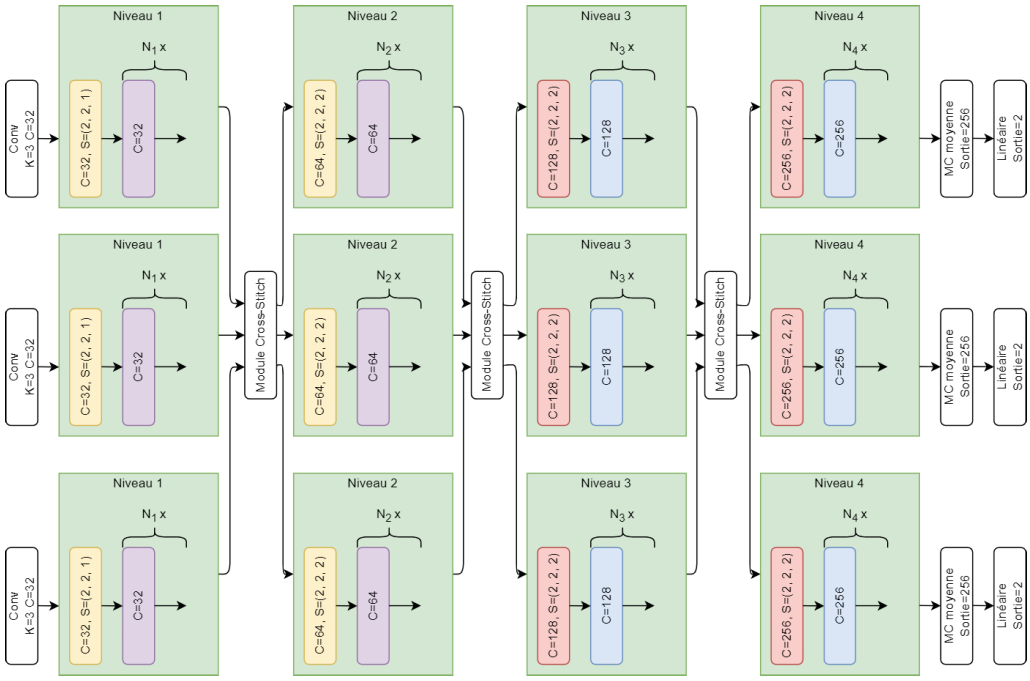


Figure 4.10 – La forme générale du modèle de type partage incité. Le modèle utilise la configuration des modules #2. C : Nombre de canaux en sortie. K : Taille du noyau de convolution. MC : Couche de mise en commun. S : foulée.

Comme le montre la figure 4.10, nous avons placé les modules de partage entre chaque niveaux du modèle. Nous avons créé quatre configurations différentes où les modules sont placés à différents endroits. La première utilise des modules de partage après chaque niveau, la seconde en utilise à la fin de chacun des trois premiers niveaux seulement, la suivante utilise des modules après les niveaux 1 et 2 et la dernière en utilise seulement après les niveaux 2 et 3. Nous avons utilisé un facteur régularisation L2 différent pour les modules de point de croix du reste du réseau.

Modèle d'attention multitâche

Le réseau d'attention multitâche (RAM), tel que présenté à la section 3.3.3, est une architecture où seuls les modules d'attention et les classifieurs ne sont pas partagés par les différentes tâches. Bien qu'il ait été conçu pour des tâches segmentations et de prédiction de la profondeur, nous l'avons adapté aux tâches de classifications. Comme le montre la figure 4.11, notre version utilise la partie convolutive de notre ResNet3D en tant que tronc.

Comme expliqué précédemment, le tronc du ResNet est séparé en quatre niveaux qui sont eux séparés en N_i blocs résiduels qui peuvent être de type post-activation ou de type pré-activation. Dans le cas du RAM, chaque niveau possède un module d'attention (de type spatial, sur canaux ou MBAC voir 3.4, 3.3 et 3.5 respectivement). Le premier module d'attention reçoit en entrée la sortie de l'avant-dernier bloc résiduel de ce niveau afin de produire un masque qui sera ensuite multiplié à la sortie du dernier bloc résiduel. Afin que les cartes de caractéristiques soient de même dimension que la sortie du prochain niveau, une couche de convolution partagée par toutes les tâches est utilisée pour sous-échantillonner ces cartes de caractéristiques. Le fonctionnement est identique pour les modules d'attention suivants, à la différence que leurs entrées correspondent à la sortie de l'avant dernier bloc résiduel de leur niveau, concaténé au résultat sous-échantillonné du niveau précédent. Les cartes de caractéristiques produites par le dernier bloc d'attention seront sous-échantillonnées à l'aide d'une couche de type mise en commun moyenne.

Finalement, puisque le RAM utilise un ResNet 3D comme tronc, les quatre configurations de niveau présentées précédemment sont toujours disponibles. Cependant, dû à une erreur d'inattention de notre part, le modèle a été fixé à la configuration 4,

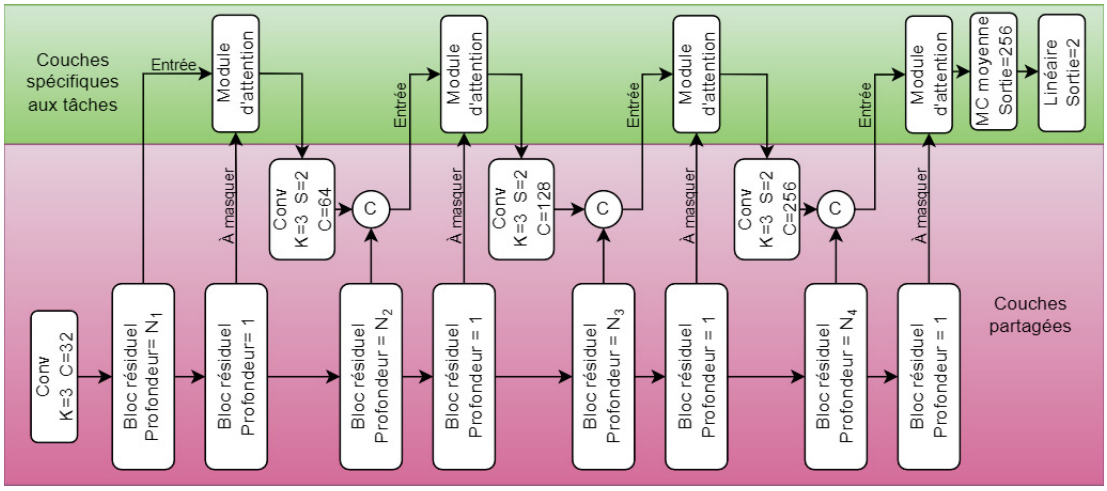


Figure 4.11 – La forme générale du réseau d'attention multitâche (RAM) tel qu'utilisé dans nos expérimentations. MC : Couche de mise en commun.

c'est-à-dire pré-act x2 - post-act x2.

Learning-to-Branch

Le LTB est davantage un algorithme de recherche d'architecture multitâche en arbre plutôt qu'une architecture en soit. Pour rappel, il utilise des modules de branchement qui servent à connecter au maximum un noeud parent à chaque noeud enfant (voir section 3.3.4). Par exemple, supposons que nous avons n sous-réseaux parallèles et que nous plaçons un module de branchement entre les couches $l-1$ et l de chaque sous-réseau. À la fin de la première phase d'apprentissage, l'algorithme sera capable de déterminer quelle sortie de la $l-1^e$ couche de quel sous-réseau devra être utilisée comme entrée de la l^e couche du k^e sous-réseau. Ainsi, l'algorithme pourra décider combien de couches partagent chacune des tâches avec les autres.

Pour nos expérimentations, nous avons décidé que la recherche d'architecture utiliserait des ResNet 3D utilisant la configuration 3 (post-act $\times 2$ - pré-act $\times 2$). Le choix de la configuration a été motivé par la recherche d'hyperparamètres des modèles à tâche uniques. Afin de donner une flexibilité maximale à notre algorithme, nous avons placé un module de branchement après chaque bloc résiduel de chaque sous-réseau. Pour ces modules, nous avons utilisé un taux d'apprentissage dix fois

plus élevé que les autres couches ainsi qu'un taux de régularisation que nous avons fixé à $1 \cdot 10^{-6}$. Encore une fois, basés sur des expériences préliminaires (voir annexe B.4), nous avons décidé que le paramètre de température τ utilisé pour entraîner les blocs de branchement serait fixé à dix. Pour finir, dans le cas où il n'y avait pas de tâche auxiliaire, le nombre de sous-réseaux est égal au nombre de tâches. Dans le cas contraire, nous avons choisi d'utiliser quatre sous-réseaux lorsque la tâche principale était la classification de la malignité ou du grade et seulement trois sous-réseaux pour la classification du sous-type.

4.4 Déroulement des expérimentations

La première étape de nos expériences, visant à évaluer les performances de la classification de tumeurs rénales à l'aide d'IRM dans un contexte d'apprentissage multitâche, fut d'identifier manuellement les patients à retirer de notre jeu de données. Par la suite, nous avons échantillonné de notre jeu de donnée 10% des données afin de créer notre ensemble de retenue qui ne serait utilisé qu'à la dernière étape de nos expérimentations pour l'évaluation finale. Nous retirons les patients identifiés plus tôt seulement après cette étape, afin de pouvoir en réinsérer une partie plus tard si nécessaire. Par la suite, pour chaque architecture nous avons procédé à une phase d'évaluation afin de sélectionner le meilleur ensemble d'hyperparamètres. Les architectures de type partage forcé et LTB ont subi une seconde phase d'évaluation, mais cette fois-ci en utilisant des caractéristiques radiomiques comme tâches auxiliaires. Après cela, nous avons, cinq fois, entrainé le modèle final de chaque architecture en utilisant une séparation aléatoire entrainement/validation 80 :20 afin de mesurer les performances sur l'ensemble de test final.

4.4.1 Phase de validation

La phase d'évaluation des architectures a servi à identifier les hyperparamètres qui allaient être utilisés pour former le modèle final. Elle est principalement composée d'une recherche d'hyperparamètre qui maximise la justesse balancée sur un ensemble de test créé aléatoirement en échantillonnant 20% de notre jeu de données avant

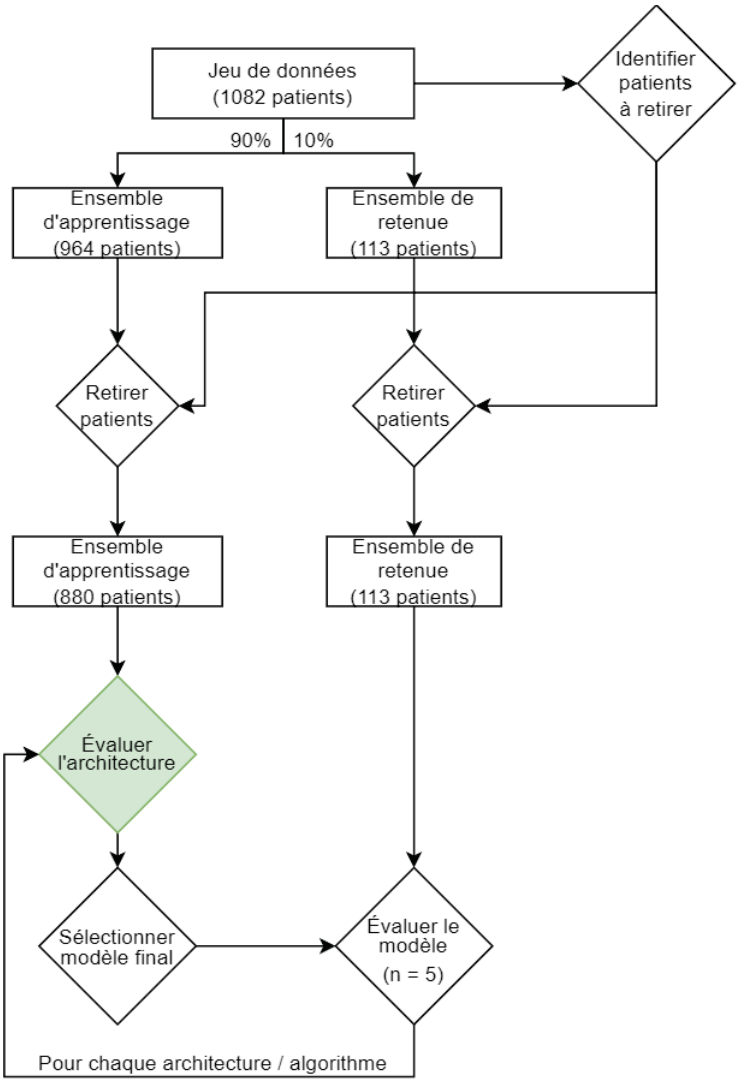


Figure 4.12 – Plan du déroulement des expérimentations. n : Nombre de fois où l'expérimentation est répétée.

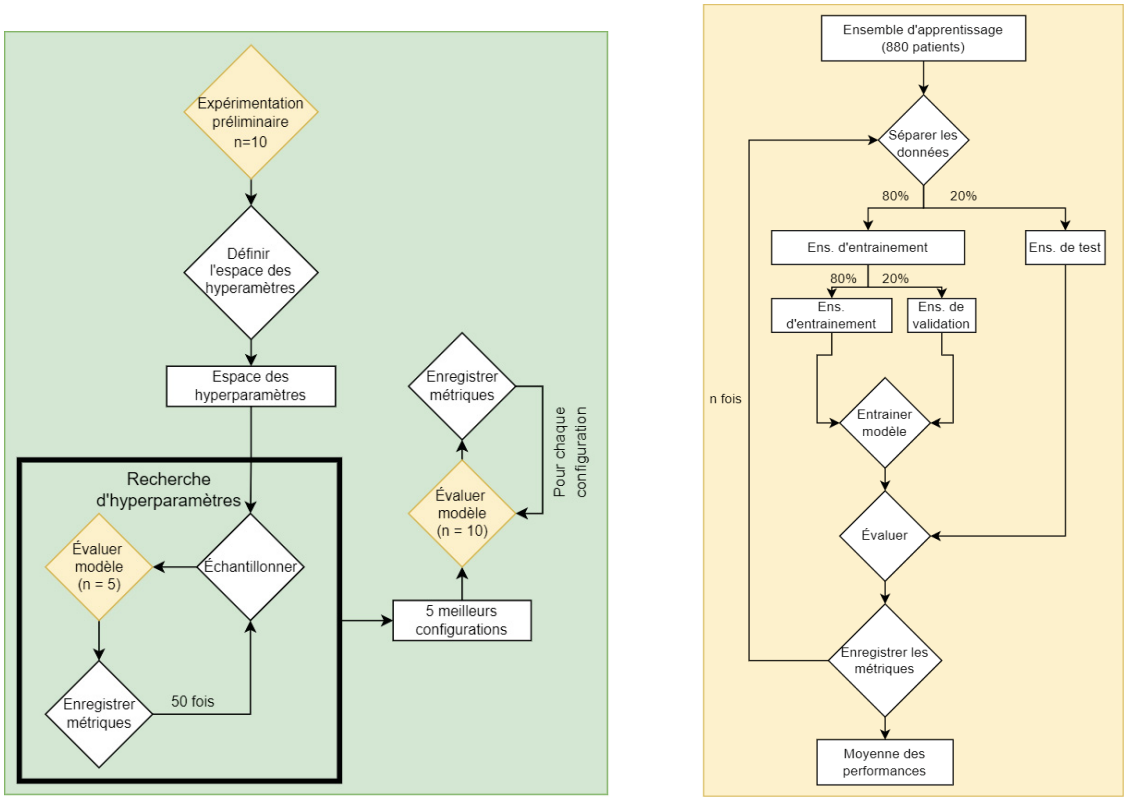
chaque entrainement.

Pour les MVS, nous avons effectué une recherche d'hyperparamètres de 250 itérations en utilisant en parallèle quatre algorithmes de recherche d'hyperparamètre, soit la recherche aléatoire, l'arbre d'estimateur de parzen et le processus gaussien en utilisant les fonctions probabilité maximale d'amélioration (PMA) et d'amélioration espérée (AE). Pour ce faire, nous avons utilisé le code d'un projet nommé HyperPara qui est disponible sur la plateforme GitHub [7].

Pour les modèles de type réseau neuronal, la phase de validation est un peu plus complexe. Comme le montre la figure 4.13, pour chaque architecture, nous avons commencé par évaluer la sensibilité de nos hyperparamètres, c'est la phase des expérimentations préliminaires disponible en partie à l'annexe B. Cette étape consistait à évaluer indépendamment différents hyperparamètres afin de sélectionner ceux qui feraient partie de la recherche d'hyperparamètres. Par la suite, pour chaque architecture, à l'exception du Learning-To-Branch, nous avons fait une recherche d'hyperparamètres de 50 itérations à l'aide de l'algorithme de recherche aléatoire. Ce choix est motivé par le temps d'entrainement conséquent de nos modèles qui nous force à limiter le nombre d'itérations. À la fin de la recherche d'hyperparamètres, les cinq meilleures configurations ont été sélectionnées pour une seconde phase de validation avant de sélectionner le meilleur modèle. L'absence de recherche d'hyperparamètres pour le LTB est due à son temps d'entrainement qui de deux à quatre fois plus long que les autres modèles. Nous avons préféré définir manuellement une configuration à tester en nous basant sur les résultats de la recherche d'hyperparamètres du partage forcé.

Pour finir, il est important de mentionner quelques détails à propos de l'entraînement de nos réseaux neuronaux. Tous nos modèles, à l'exception du LTB, ont été entraînés pendant un total de 100 cycles d'entraînement. Pour sa part, le Learning-To-Branch utilise 200 cycles pour la recherche d'architecture et 100 cycles supplémentaires pour réentraîner le modèle avec l'architecture optimale. Afin de maximiser les performances, nous avons utilisé la méthode de l'arrêt prématuré avec un critère de sauvegarde utilisant la moyenne géométrique des rappels afin de sauvegarder le

7. Disponible à la page https://github.com/AleAyotte/HyperPara, consulté le 2 mars 2023.



(a) Évaluation d'une architecture

(b) Évaluation d'un modèle

Figure 4.13 – Déroulement de la phase d'évaluation des architectures de type réseau neuronal.

modèle au meilleur cycle. Pour les modèles multitâches, le critère de sauvegarde est basé sur la moyenne géométrique des performances sur les tâches principales. Le choix du critère d'arrêt est basé sur les résultats d'expériences préliminaires disponibles à l'annexe B.1.3. À la fin de l'entraînement, nous avons optimisé le seuil de décision pour chaque tâche afin de maximiser la justesse balancée sur l'ensemble de validation.

4.4.2 Sélection des tâches auxiliaires

Avec plus de 300 caractéristiques radiomiques pouvant être utilisées comme tâches auxiliaires pour nos expérimentations, il nous était impossible de tester les $2^{300} - 1$ sous-ensemble de tâches pour chacun de nos problèmes de classification. Afin de former le sous-ensemble de tâches optimales, nous avons en premier lieu tenté d'utiliser

l'algorithme GTA, mais comme il sera discuté à la section 6.3, cet algorithme a échoué à identifier les caractéristiques radiomiques les plus propices à améliorer les performances de nos modèles pour chacune de nos tâches principales. Nous avons donc créé manuellement différents groupes de tâches radiomiques.

Pour chaque tâche principale, 13 groupes de tâches auxiliaires ont ainsi été créés. Les neuf premiers groupes correspondent à neuf des 11 familles de caractéristiques radiomiques (voir section 1.3.2). Les familles d'intensité local et de statistiques basées sur l'intensité, n'ont pas été considérées, car elles ne peuvent pas être calculées avec les modalités d'image que nous utilisons. Par la suite, pour chaque tâche de classification, nous avons formé un groupe de cinq tâches auxiliaires et un autre de dix, en sélectionnant les caractéristiques radiomiques dont la corrélation point-bisériale avec la tâche principale était la plus élevée. Deux autres groupes de tâches de tailles similaires ont été créés à l'aide d'un algorithme qui utilise le critère *minimal-redundancy-maximal-relevance* [98] pour sélectionner les caractéristiques les plus liées à la tâche principale. Le choix du sous-ensemble de tâche à utiliser est donc devenu un hyperparamètre qui a été fixé avant la recherche d'hyperparamètres.

Chapitre 5

Résultats

Ce chapitre présente les résultats de tous nos modèles l'un à la suite de l'autre. Pour chaque modèle, nous commencerons par résumer les conclusions que nous avons retirées de nos expérimentations préliminaires lorsqu'il y en avait, nous listerons les différents hyperparamètres choisis et nous présenterons les résultats obtenus par la configuration finale sur l'ensemble de retenue.

5.1 Modèles à tâche simple

Au chapitre 1, nous avons établi que notre premier objectif était de vérifier si les modèles de type réseau de neurones sont mieux adaptés qu'un modèle plus simple utilisant seulement les données cliniques. Cette section nous permet aussi, plus simplement, de déterminer si les données clinique et/ou imagées sont pertinentes pour la classification de la malignité des tumeurs rénales, et la classification du sous-type et du grade des tumeurs malignes.

5.1.1 Données cliniques et machine à vecteur de support

Comme mentionné à la section 4.4.1, les modèles utilisant les données tabulaires n'ont pas eu besoin d'expérimentations préliminaires. Vu le peu d'hyperparamètres disponibles nous n'avions pas besoin d'en fixer certains à l'avance. Deux expérimen-

Tableau 5.1 – Liste des hyperparamètres utilisés pour la machine à vecteur de support

Hyperparamètre	Distribution	Min	Max	Pas	Catégorie
C	Log-uniforme	1e-8	1e0	N/A	N/A
Gamma	Log-uniforme	1e-8	1e0	N/A	N/A
Noyau	Catégorique	N/A	N/A	N/A	RBF, Sigmoid
K*	Discrète	1	35	1	N/A
Taux*	Uniforme	0.5	0.9	N/A	N/A

C : paramètre de régularisation. K : Nombre de voisin pour l'algorithme des K plus proche voisins (Adasyn). Taux : Contrôle le ratio (classe minoritaire / classe majoritaire) (Adasyn). * : Pour la classification du sous-type seulement.

tations ont été faites pour déterminer si nos différents problèmes de classification pouvaient bénéficier de l'utilisation de l'algorithme Adasyn (voir section A.1.2), un algorithme d'augmentation de données qui vise à corriger le problème de débalancement des classes. Il nous est apparu que seule la classification du sous-type en profitait. Comme le montre le tableau 5.1, mises à part les hyperparamètres en lien avec Adasyn, nous n'avions que trois hyperparamètres ; Le choix du noyau, le paramètre de régularisation C et le paramètre gamma qui affecte les noyaux RBF et sigmoid.

Les résultats obtenus par nos machine à vecteur de support (MVS) sur l'ensemble de test final qui sont présentés dans le tableau 5.2, montrent que seul celui entraîné pour la classification de la malignité semble obtenir des performances intéressantes. Avec une justesse balancée de 72.33% et une surface sous la courbe (SSC) de 0.722, ce dernier est capable d'effectuer la tâche demandée bien mieux que les deux autres MVS.

Tableau 5.2 – Résultats des modèles de type machine à vecteur de support sur l'ensemble de retenue

Tâche	SSC	Justesse balancée	Score F1	CCM
Malignité	0.722	72.33%	0.837	0.469
Sous-type	0.586	58.61%	0.559	0.156
Grade	0.610	61.00%	0.400	0.156

SSC : Surface sous la courbe. CCM : Coefficient de corrélation de Matthews.

Les modèles entraînés pour la classification du sous-type et du grade n'obtiennent pour leur part qu'une justesse balancée de 58.61% et 61.00% respectivement. Avec un coefficient de corrélation de Matthews (CCM) de 0.156, soit trois fois inférieurs à celui obtenu pour le premier modèle, il est clair que les données cliniques utilisées ne sont pas suffisantes pour résoudre ces deux tâches.

5.1.2 Réseau de neurones à tâche unique

Les expérimentations préliminaires effectuées pour nos réseaux de neurones à tâche unique nous ont permis non seulement de définir un espace d'hyperparamètres pour ce type de modèle, mais aussi de poser les bases pour la construction de toutes nos architectures multitâches. Comme décrit à l'annexe B.1, les expérimentations préliminaires les plus pertinentes portaient sur trois points : 1) Les architectures hybrides, 2) la fusion des masques de segmentations et 3) le critère de sauvegarde à utiliser durant l'entraînement (voir section 4.3.1 à B.1.3 respectivement). Nous en avons tiré les conclusions suivantes :

1. Il peut être bénéfique d'utiliser à la fois des blocs résiduels de type post-activation et pré-activation dans une même architecture. Toutefois, la meilleure configuration pourrait varier d'une tâche à l'autre et mérite de devenir un hyperparamètre.
2. Bien que la fusion des masques de segmentation des images T1C et T2 puisse améliorer les performances pour la classification du grade, il y a aussi une perte de performance importante pour les deux autres tâches. Les masques ne seront donc pas fusionnés.
3. La meilleure mesure de performance à utiliser comme critère de sauvegarde est la moyenne géométrique du rappel de chaque classe. Ce paramètre sera fixé.

En raison de ces résultats et quelques expérimentations moins importantes ou concluantes, nous avons défini l'espace des hyperparamètres tel que d'écrit dans le tableau 5.3

Comme l'indique le tableau 5.4, les résultats obtenus par nos modèle utlisant des données imagées sont supérieurs à ceux utilisant des données tabulaires. De toutes les

Tableau 5.3 – Liste des hyperparamètres utilisés pour le modèle à tâche unique

Hyperparamètre	Distribution	Min	Max	Pas	Catégorie
Configuration des niveaux	Catégorique	N/A	N/A	N/A	1, 2, 3, 4
Taux d'extinction des neurones	Uniforme	0.1	0.7	N/A	N/A
Nombre de canaux	Discrète	16	64	1	N/A
Profondeur	Discrète	18	34	16	N/A
Régularisation L2	Log-uniforme	1e-8	1e-2	N/A	N/A
Taille des lots	Discrète	16	48	2	N/A
Taux d'apprentissage	Log-uniforme	1e-5	1e-2	N/A	N/A
ϵ	Log-uniforme	1e-6	1e-1	N/A	N/A

Nombre de canaux : Nombre de canaux à l'entrée de tous les blocs du premier niveau ($N1$).
ϵ : Hyperparamètre lié à l'algorithme d'optimisation *Adam*.

Tableau 5.4 – Résultats des modèles de type réseaux de neurone à tâche unique sur l'ensemble de retenue

Tâche	SSC	Justesse balancée	Score F1	CCM
Malignité	0.825 ± 0.036	$74.58\% \pm 3.66\%$	0.811 ± 0.045	0.482 ± 0.083
Sous-type	0.792 ± 0.035	$76.44\% \pm 4.77\%$	0.810 ± 0.025	0.473 ± 0.084
Grade	0.753 ± 0.020	$69.50\% \pm 3.66\%$	0.615 ± 0.024	0.386 ± 0.050

Moyenne des performances sur 5 entrainements. SSC : Surface sous la courbe. CCM : Coefficient de corrélation de Matthews.

tâches, c'est pour la classification du sous-type que la différence est la plus flagrante avec un gain absolu de 17.83% pour la justesse balancée et de 0.317 pour le CCM. À l'inverse, la classification de la malignité montre des gains beaucoup plus modestes avec un gain de seulement 2.25% pour la justesse balancée, de 0.103 pour le SSC et même une perte de 0.026 au niveau du score F1. Pour le grade, toutes les mesures enregistrent un gain, en particulier le score F1 (+0.215) et le CCM (+0.230).

5.2 Modèles multitâches

Le second objectif de notre recherche est de déterminer si l'apprentissage multitâche peut être bénéfique pour les problèmes de classification de tumeurs rénales à l'aide de réseaux neuronaux. La section ci-dessous présentera nos résultats pour les modèles de type partage forcé, partage incité et RAM, ainsi que ceux de l'algorithme Learning-To-Branch.

5.2.1 Modèle de type partage forcé

Nos expérimentations sur la fonction de coût multitâche (disponible à l'annexe B.2), a été la plus importante de l'étape préliminaire de notre architecture de type partage forcé. Elle nous a permis de choisir la fonction de coût basée sur l'incertitude comme choix final pour tous les modèles qui suivront dans cette section. Les autres expérimentations préliminaires, qui portaient principalement sur la profondeur des sous-réseaux et le nombre de couches partagées, n'ont pas été suffisamment

Tableau 5.5 – Liste des hyperparamètres utilisés pour les modèles de type partage forcé

Hyperparamètre	Distribution	Min	Max	Pas	Catégorie
Coefficient des tâches auxiliaire*	Uniforme	0.1	0.8	N/A	N/A
Configuration	Catégorique	N/A	N/A	N/A	A†, B
Taux d'extinction des neurones	Uniforme	0.1	0.7	N/A	N/A
Nbr canaux	Discrète	16	64	1	N/A
Profondeur	Catégorique	N/A	N/A	N/A	18, 34, Mixte†
Régularisation L2	Log-uniforme	1e-8	1e-2	N/A	N/A
Niveau de séparation	Discrète	2	4	1	N/A
Taille des lots	Discrète	16	48	2	N/A
Taux d'apprentissage	Log-uniforme	1e-5	1e-2	N/A	N/A
ϵ	Log-uniforme	1e-6	1e-1	N/A	N/A

Nombre de canaux : Nombre de canaux à l'entrée de tous les blocs du premier niveau ($N1$). Niveau de séparation : Premier niveau à être dupliqué en plusieurs branches. ϵ : Hyperparamètre lié à l'algorithme d'optimisation Adam. † : La configuration A et la profondeur mixte ne sont disponibles que pour les expériences qui n'incluent pas de tâches auxiliaires. * : Pour les expériences incluant des tâches auxiliaires seulement.

concluantes pour pouvoir fixer un autre hyperparamètre, ou même être incluses dans notre annexe. Suite à cela, nous avons défini l'espace des hyperparamètres pour les modèles de type partage forcé avec et sans tâches auxiliaires, tel que décrit dans le tableau 5.5. Pour rappel, les choix de configurations A et B correspondent à des combinaisons de blocs résiduels utilisées pour créer l'architecture.

Les résultats de nos expériences sur l'ensemble de retenue, qui sont affichés dans les tableaux 5.6a et 5.6b, montrent que le modèle de type partage forcé peut atteindre des performances comparables à notre modèle à tâche simple pour au moins deux tâches sur trois. En effet, pour la classification de la malignité cette architecture atteint en moyenne, par rapport à la combinaison de tâches choisie (moyenne de chaque colonne), une justesse balancée de 75.17% pour un CCM de 0.484 et un SSC de 0.818. Pour ce qui est du grade, la justesse balancée atteint un maximum de 74.81%, soit 5.31% de plus que notre modèle à tâche unique. En moyenne, les performances pour cette tâche

Tableau 5.6 – Résultats du modèle de type partage forcé sur l'ensemble de retenue

(a) Résultats pour les mesures de surface sous la courbe (SSC) et de justesse balancée (JB)

	Malignité		Sous-type		Grade	
Tâches	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
M + ST + G	0.798 ± 0.071	72.17 ± 3.06	0.796 ± 0.037	75.56 ± 3.73	0.785 ± 0.023	74.81 ± 3.84
M + ST	0.800 ± 0.010	75.04 ± 2.00	0.790 ± 0.022	75.33 ± 4.96	–	–
M + G	0.856 ± 0.017	78.30 ± 2.40	–	–	0.756 ± 0.021	68.52 ± 2.67
ST + G	–	–	0.761 ± 0.018	70.92 ± 4.61	0.772 ± 0.033	70.55 ± 4.92

(b) Résultats pour les mesures de Score F1 et de Coefficient de corrélation de Matthews (CCM)

	Malignité		Sous-type		Grade	
Tâches	Score F1	CCM	Score F1	CCM	Score F1	CCM
M + ST + G	0.792 ± 0.028	0.429 ± 0.060	0.796 ± 0.046	0.460 ± 0.070	0.678 ± 0.051	0.502 ± 0.070
M + ST	0.796 ± 0.034	0.480 ± 0.044	0.792 ± 0.061	0.456 ± 0.096	–	–
M + G	0.826 ± 0.020	0.543 ± 0.048	–	–	0.608 ± 0.035	0.372 ± 0.056
ST + G	–	–	0.772 ± 0.051	0.377 ± 0.088	0.632 ± 0.058	0.408 ± 0.096

La combinaison de tâches employée est indiquée dans la colonne Tâches. Moyennes des performances sur 5 entrainements. M : Malignité. ST : Sous-type. G : Grade.

sont moins élevées avec une justesse balancée de 71.29%, un score F1 de 0.639 et un CCM de 0.427, malgré un SSC moyen de 0.771. À l'inverse, la classification du sous-type montre une perte de performance, peu importe la combinaison de tâche utilisée, avec une justesse balancée maximale de 75.56%, un score F1 de 0.796 et un CCM de 0.460. Pour finir, nous ferons remarquer au lecteur que le choix de la combinaison de tâches fait beaucoup varier les résultats des différentes tâches.

5.2.2 Modèle de type partage incité

Les expérimentations préliminaires du modèle, disponibles à l'annexe B.3, n'ont porté que sur la nécessité de pré-entraîner ou non les sous-réseaux qui forment le modèle. Pour faire bref, le pré-entraînement des sous-réseaux semblait plutôt nuisible aux performances du modèle final, ce qui contredit l'article original qui proposait cette architecture [91]. Nous croyons que cela est dû au fait que nos différentes tâches n'apprennent pas à la même vitesse et qu'il était difficile d'arrêter l'entraînement

Tableau 5.7 – Liste des hyperparamètres utilisés pour le modèle de type partage incité

Hyperparamètre	Distribution	Min	Max	Pas	Catégorie
c	Uniforme	0.5	1.0	N/A	N/A
Taux d'extinction des neurones	Uniforme	0.1	0.7	N/A	N/A
Nombre de canaux	Discrète	16	64	1	N/A
Profondeur	Catégorique	N/A	N/A	N/A	18, 34, Mixte
Régularisation L2	Log-uniforme	1e-8	1e-2	N/A	N/A
Régularisation L2 (Module de partage)	Log-uniforme	1e-8	1e-2	N/A	N/A
Configuration des modules	Catégorique	N/A	N/A	N/A	1, 2, 3, 4
Dispersion	Uniforme	0.01	0.30	N/A	N/A
Taille des lots	Discrète	16	48	2	N/A
Taux d'apprentissage	Log-uniforme	1e-5	1e-2	N/A	N/A
ϵ	Log-uniforme	1e-6	1e-1	N/A	N/A

c : Correspond au paramètre de conservation c des modules de partage. Nombre de canaux : Nombre de canaux à l'entrée de tous les blocs du premier niveau ($N1$). Dispersion : Correspond au paramètre de dispersion d des modules de partage. ϵ : Hyperparamètre lié à l'algorithme d'optimisation Adam.

avant que ces sous-réseaux ne commencent à sur-apprendre.

L'espace des hyperparamètres (voir tableau 5.7) est très semblable à celui du partage forcé et du réseau de neurones à tâche unique. Nous avons remplacé l'hyperparamètre en lien avec la configuration des blocs par un autre qui contrôle la position des modules de partage. Nous avons aussi ajouté deux hyperparamètres, nommées c et dispersion, qui contrôlent l'initialisation des modules de partages.

Pour ce qui est des résultats sur l'ensemble de retenue, notre modèle de type parage incité obtient des résultats qui sont beaucoup moins sensibles au choix de la combinaison de tâches (voir tableaux 5.8a et 5.8b). Encore une fois, la classification de la malignité semble profiter de cette architecture multitâche avec une justesse balancée moyenne de 76.54%, un SSC de 0.849 et un CCM de 0.512. La classification du sous-type affiche encore des résultats légèrement inférieurs avec en moyenne une justesse balancée de 75.79%, un SSC de 0.777 et un score F1 de 0.792. Enfin, pour

Tableau 5.8 – Résultats du modèle de type partage incité sur l'ensemble de retenue

(a) Résultats pour les mesures de surface sous la courbe (SSC) et de justesse balancée (JB)

	Malignité		Sous-type		Grade	
Tâches	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
M + ST + G	0.844 ± 0.024	77.73 ± 3.44	0.764 ± 0.033	74.71 ± 1.88	0.717 ± 0.014	64.42 ± 3.33
M + ST	0.859 ± 0.013	75.43 ± 3.49	0.787 ± 0.043	76.36 ± 2.61	–	–
M + G	0.843 ± 0.035	76.46 ± 2.14	–	–	0.722 ± 0.034	65.48 ± 2.43
ST + G	–	–	0.781 ± 0.046	76.31 ± 4.36	0.726 ± 0.028	67.20 ± 1.53

(b) Résultats pour les mesures de Score F1 et de Coefficient de corrélation de Matthews (CCM)

	Malignité		Sous-type		Grade	
Tâches	Score F1	CCM	Score F1	CCM	Score F1	CCM
M + ST + G	0.823 ± 0.020	0.535 ± 0.056	0.792 ± 0.018	0.442 ± 0.030	0.556 ± 0.036	0.288 ± 0.071
M + ST	0.821 ± 0.018	0.495 ± 0.063	0.795 ± 0.027	0.470 ± 0.047	–	–
M + G	0.810 ± 0.015	0.506 ± 0.038	–	–	0.566 ± 0.027	0.308 ± 0.051
ST + G	–	–	0.789 ± 0.024	0.466 ± 0.075	0.577 ± 0.026	0.353 ± 0.030

La combinaison de tâches employée est indiquée dans la colonne Tâches. Moyennes des performances sur 5 entrainements. M : Malignité. ST : Sous-type. G : Grade.

la classification du grade, les résultats montrent une perte de performances avec une justesse balancée moyenne de 65.70% et de même pour le score F1 et le CCM qui s'élèvent respectivement à 0.567 et 0.316.

5.2.3 Réseau d'apprentissage multitâche

Pour le RAM, aucune de nos expérimentations préliminaires n'a obtenue de résultats concluant ou suffisamment intéressant pour figurer dans notre annexe. Elles portaient principalement sur le choix du module d'attention à utiliser. Ainsi, nous avons décidé de l'ajouter à notre espace des hyperparamètres (voir tableau 5.9). Il est à noter que la configuration des niveaux qui contrôlent la combinaison de blocs résiduels utilisés pour notre architecture n'a finalement pas fait partie de la recherche d'hyperparamètres à cause d'une erreur de programmation [1]. Ainsi, seulement la configuration 4 a été utilisée durant nos expériences (voir 4.3.1) puisque nous aurions dû recommencer 1 mois de calculs GPU (voir section 6.4.1 pour plus d'explication).

À la différence de nos modèles multitâches présentés plutôt, si l'on regarde seulement la justesse balancée, le RAM a obtenu des résultats qui sont supérieurs à notre

Tableau 5.9 – Liste des hyperparamètres utilisés pour le réseau d'attention multitâche

Hyperparamètre	Distribution	Min	Max	Pas	Catégorie
Configuration des niveaux	Catégorique	N/A	N/A	N/A	1, 2, 3, 4
Taux d'extinction des neurones	Uniforme	0.1	0.7	N/A	N/A
Nombre de canaux	Discrète	16	64	1	N/A
Module d'attention	Catégorique	N/A	N/A	N/A	Sur canaux, Spatiaux, CBAM
Profondeur	Catégorique	N/A	N/A	N/A	18, 34
Régularisation L2	Log-uniforme	1e-8	1e-2	N/A	N/A
Taille des lots	Discrète	16	48	2	N/A
Taux d'apprentissage	Log-uniforme	1e-5	1e-2	N/A	N/A
ϵ	Log-uniforme	1e-6	1e-1	N/A	N/A

Nombre de canaux : Nombre de canaux à l'entrée de tous les blocs du premier niveau ($N1$).
ϵ : Hyperparamètre lié à l'algorithme d'optimisation Adam.

1. Cette erreur de code a été corrigé avant de procéder à la validation de nos résultats (voir section 6.4.3

Tableau 5.10 – Résultats du réseau d'attention multitâche sur l'ensemble de retenue

(a) Résultats pour les mesures de surface sous la courbe (SSC) et de justesse balancée (JB)

	Malignité		Sous-type		Grade	
Tâches	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
M + ST + G	0.828 ± 0.033	78.15 ± 1.69	0.779 ± 0.090	79.08 ± 4.41	0.735 ± 0.065	68.47 ± 4.40
M + ST	0.769 ± 0.077	74.59 ± 3.18	0.790 ± 0.024	76.58 ± 4.68	–	–
M + G	0.879 ± 0.024	78.86 ± 1.06	–	–	0.726 ± 0.038	70.24 ± 2.65
ST + G	–	–	0.809 ± 0.024	78.50 ± 2.96	0.724 ± 0.068	71.05 ± 2.92

(b) Résultats pour les mesures de Score F1 et de Coefficient de corrélation de Matthews (CCM)

	Malignité		Sous-type		Grade	
Tâches	Score F1	CCM	Score F1	CCM	Score F1	CCM
M + ST + G	0.828 ± 0.024	0.543 ± 0.037	0.807 ± 0.028	0.515 ± 0.078	0.604 ± 0.057	0.367 ± 0.088
M + ST	0.810 ± 0.032	0.478 ± 0.065	0.797 ± 0.048	0.474 ± 0.086	–	–
M + G	0.835 ± 0.019	0.559 ± 0.022	–	–	0.625 ± 0.031	0.403 ± 0.055
ST + G	–	–	0.811 ± 0.033	0.509 ± 0.051	0.642 ± 0.031	0.412 ± 0.058

La combinaison de tâches employée est indiquée dans la colonne Tâches. Moyennes des performances sur 5 entrainements. M : Malignité. ST : Sous-type. G : Grade.

réseau de neurones à tâche unique. En effet, pour cette mesure, les résultats moyens sont de 77.20%, 78.05% et 69.92% pour la classification de la malignité, du sous-type et du grade respectivement. Cependant, si l'on regarde les autres mesures, les résultats sont plus modestes avec un score F1 moyen respectif de 0.824, 0.805 et 0.623, ainsi qu'un CCM moyen respectif de 0.527, 0.499 et 0.394. Encore une fois, les résultats pour ce modèle ne semblent pas trop être affectés par la combinaison de tâches choisie.

5.2.4 Learning-To-Branch

Pour nos expérimentations faites avec l'algorithme Learning-To-Branch, il n'y a pas eu de recherche d'hyperparamètres, puisqu'il nous fallait en moyenne trois fois plus de temps pour entraîner un modèle avec cet algorithme que nos modèles de type partage forcé. Pour palier à ce manque, nous avons fait une expérimentation

préliminaire pour fixer à l'avance la valeur de l'hyperparamètre de température τ, dont les détails se trouvent à l'annexe B.4. Pour ce qui est des autres hyperparamètres, nous avons utilisé des valeurs qui étaient communes aux expérimentations préliminaires de nos différents modèles. Malgré l'absence de recherche d'hyperparamètres, les résultats du LTB ne sont pas en reste comparativement à nos autres architectures multitâches. Par rapport à la combinaison de tâches choisies, le SSC moyen pour la classification de la malignité, du sous-type et du grade est respectivement de 0.859, 0.814 et de 0.743, alors que la justesse balancée moyenne est de 77.03%, 76.98% et 67.66%. Pour ce qui est du score F1 et du CCM, la classification de la malignité obtient respectivement 0.814 et 0.520, alors que pour le sous-type ces mesures s'élèvent à 0.807 et 0.485 et pour finir les performances pour la classification du grade atteignent 0.599 et 0.350 respectivement. Comme le montre les tableaux 5.11a et 5.11b, les résultats semblent légèrement sensibles au choix de la combinaison de tâches lorsqu'on observe la justesse

Tableau 5.11 – Résultats de l'algorithme Learning-To-Branch sur l'ensemble de retenue

(a) Résultats pour les mesures de surface sous la courbe (SSC) et de justesse balancée (JB)

	Malignité		Sous-type		Grade	
Tâches	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
M + ST + G	0.844 ± 0.007	74.03 ± 0.73	0.830 ± 0.036	77.51 ± 4.53	0.747 ± 0.031	69.07 ± 3.63
M + ST	0.876 ± 0.017	79.15 ± 4.14	0.796 ± 0.041	77.17 ± 4.31	–	–
M + G	0.857 ± 0.020	77.89 ± 2.78	–	–	0.737 ± 0.034	67.61 ± 3.16
ST + G	–	–	0.817 ± 0.027	76.26 ± 3.10	0.744 ± 0.015	66.29 ± 5.77

(b) Résultats pour les mesures de Score F1 et de Coefficient de corrélation de Matthews (CCM)

	Malignité		Sous-type		Grade	
Tâches	Score F1	CCM	Score F1	CCM	Score F1	CCM
M + ST + G	0.798 ± 0.017	0.462 ± 0.014	0.816 ± 0.044	0.497 ± 0.088	0.610 ± 0.042	0.382 ± 0.075
M + ST	0.832 ± 0.030	0.560 ± 0.076	0.793 ± 0.042	0.482 ± 0.079	–	–
M + G	0.812 ± 0.031	0.537 ± 0.049	–	–	0.598 ± 0.037	0.349 ± 0.065
ST + G	–	–	0.812 ± 0.035	0.475 ± 0.059	0.588 ± 0.056	0.321 ± 0.115

La combinaison de tâches employée est indiquée dans la colonne Tâches. Moyennes des performances sur 5 entrainements. M : Malignité. ST : Sous-type. G : Grade.

balancée des tâches en lien avec la malignité et le grade des tumeurs, puisque ces deux problèmes montrent respectivement un écart-type de 2.18% et de 1.14%.

5.3 Utilisation des caractéristiques radiomiques comme tâches auxiliaires

Le dernier objectif de notre recherche est de déterminer si les caractéristiques radiomiques sont de bon candidats pour former des tâches auxiliaires qui pourraient aider à l'entraînement sur nos tâches de classification des tumeurs rénales. Pour des raisons matérielles, ces expérimentations n'ont été faites qu'avec le modèle multitâche de type partage forcé et l'algorithme Learning-To-Branch. Dans nos expériences, nous avons entraîné chaque tâche principale avec un ensemble de tâches auxiliaires et finalement, nous avons entraîné un modèle à effectuer nos trois tâches principales et un groupe de tâches auxiliaires.

Comme indiqué à la section 4.4.2, des expérimentations préliminaires ont été effectuées afin de déterminer le meilleur ensemble de tâches auxiliaires pour chaque tâche principale et pour les trois tâches en même temps. Pour sauver du temps, cette étape préalable n'a été faite qu'avec le modèle de type partage forcé, tout comme la recherche d'hyperparamètres qui a suivi. Ainsi, les ensembles de tâches auxiliaires que nous avons sélectionnés furent la famille de caractéristiques morphologique et le groupe de caractéristiques texturales MDVNG pour la classification de la malignité et du grade respectivement. Pour la classification du sous-type, l'ensemble des cinq caractéristiques radiomiques dont la corrélation point-bisériales était le plus élevé a été choisi comme groupe de tâches auxiliaires. Finalement, c'est la famille de radiomiques MDVNG qui semblait la plus adaptée pour la dernière expérimentation impliquant les trois tâches principales à la fois.

5.3.1 Modèle de type partage forcé

Les résultats de nos expériences pour le modèle de type partage forcé entraîné sur une seule tâche principale avec un groupe de tâches auxiliaires semblent indiquer que cette stratégie n'a été bénéfique que pour la prédiction de la malignité. En effet,

comme l'indique le tableau 5.12a, les performances de notre modèle atteignent une justesse balancée de 78.01%, soit un gain de 3.49 par rapport au modèle entrainé seulement pour la tâche principale, un SSC de 0.879 et un CCM de 0.540 pour cette tâche. Pour la classification du sous-type et du grade, les résultats sont très similaires à ceux des modèles à tâche unique. En effet, pour cette première tâche, seul le score F1 affiche un certain écart avec une valeur de 0.786, ce qui représente une perte de 0.024. Pour la seconde tâche, c'est le CCM qui est le plus intéressant avec une valeur de 0.413, un gain de 0.027.

Pour ce qui est de nos expériences avec le modèle entraîné à la fois sur les trois tâches de classification et un ensemble de tâches auxiliaires, les résultats sont moins bons pour les trois tâches principales. De fait, le tableau 5.12b montre que les performances sont en baisse, par rapport à nos modèles entraînés pour une tâche principale et un ensemble de tâches auxiliaires, pour toutes les mesures et tâches à l'exception

Tableau 5.12 – Résultats du modèle de type partage forcé avec les tâches auxiliaires radiomiques

(a) Modèle de type partage forcé entrainé à effectuer à la fois l'une des 3 tâches de classification et un ensemble de tâches de régression liées à la prédiction de caractéristiques radiomiques

Tâche	SSC	Justesse balancée	Score F1	CCM
Malignité	0.879 ± 0.029	$78.01\% \pm 3.44\%$	0.826 ± 0.038	0.540 ± 0.069
Sous-type	0.795 ± 0.014	$76.32\% \pm 2.85\%$	0.786 ± 0.056	0.471 ± 0.055
Grade	0.745 ± 0.015	$69.88\% \pm 2.89\%$	0.614 ± 0.031	0.413 ± 0.072

(b) Modèle de type partage forcé entrainé à effectuer à la fois les tâches 3 principale et un ensemble de tâches de régression liées à la prédiction de caractéristiques radiomiques

Mesures	Malignité	Sous-type	Grade
SSC	0.851 ± 0.020	0.748 ± 0.047	0.714 ± 0.017
Justesse Balancée	$76.13\% \pm 4.05\%$	$70.78\% \pm 4.65\%$	$66.08\% \pm 2.74\%$
Score F1	0.829 ± 0.023	0.784 ± 0.036	0.590 ± 0.021
CCM	0.512 ± 0.023	0.378 ± 0.078	0.315 ± 0.057

Moyenne des performances par tâche de classification sur l'ensemble de retenue pour 5 entrainements. SSC : Surface sous la courbe CFR. CCM : Coefficient de corrélation de Matthews.

du score F1 de la prédiction de la malignité qui atteint 0.829, soit une augmentation anecdotique de 0.003. Si l'on compare par rapport au modèle à tâche unique, il y a tout de même une amélioration pour cette dernière tâche avec une justesse balancée de 76.13%, une amélioration de 1.51%. À l'inverse, nous notons une perte conséquente au niveau de cette même mesure pour la prédiction du sous-type et du grade de 5.66% et 3.42% respectivement.

5.3.2 Learning-To-Branch

Le tableau 5.13a affiche les résultats du LTB pour les groupes d'une tâche principale avec un groupe de tâches auxiliaires. Pour la malignité, notre algorithme obtient une justesse balancée de 75.46%, ainsi qu'un SSC, un score F1 et un CCM de 0.822, 0.802 et 0.487 respectivement. Pour la prédiction du grade, les résultats sont un peu en baisse avec un SSC de 0.738, une justesse balancée de 68.09%, un score F1 de 0.600

Tableau 5.13 – Résultats du Learning-To-Branch avec les tâches auxiliaires radiomiques

(a) Algorithme Learning-To-Branch entrainé à effectuer à la fois l'une des 3 tâches de classification et un ensemble de tâches de régression liées à la prédiction de caractéristiques radiomiques

Tâche	SSC	Justesse balancée	Score F1	CCM
Malignité	0.822 ± 0.030	$75.46\% \pm 2.06\%$	0.802 ± 0.017	0.487 ± 0.036
Sous-type	0.825 ± 0.022	$79.17\% \pm 2.84\%$	0.820 ± 0.022	0.521 ± 0.050
Grade	0.738 ± 0.033	$68.09\% \pm 3.24\%$	0.600 ± 0.036	0.367 ± 0.080

(b) Algorithme Learning-To-Branch entrainé à effectuer à la fois les tâches 3 de classification et un ensemble de tâche de régression liées à la prédiction de caractéristiques radiomiques

Mesures	Malignité	Sous-type	Grade
SSC	0.858 ± 0.022	0.800 ± 0.021	0.760 ± 0.042
Justesse Balancée	$75.96\% \pm 2.47\%$	$75.74\% \pm 3.84\%$	$69.57\% \pm 2.80\%$
Score F1	0.801 ± 0.020	0.793 ± 0.033	0.619 ± 0.038
CCM	0.490 ± 0.042	0.460 ± 0.067	0.385 ± 0.053

Moyenne des performances sur l'ensemble de retenue pour 5 entrainements distinct par tâche classification. SSC : Surface sous la courbe CFR. CCM : Coefficient de corrélation de Matthews.

et un CCM 0.367. À l'inverse, la classification du sous-type affiche une nette amélioration. La justesse balancée atteint 79.17%, soit 2.73% de plus que notre modèle à tâche unique, un SSC de 0.825, un score F1 de 0.820 et un CCM de 0.521.

À la section 5.3.1, nous avions pu remarquer une baisse des performances lorsque le modèle était entraîné à effectuer trois tâches principales en même temps qu'un ensemble de tâches secondaires. Le tableau 5.13b montre une conclusion très différente pour le LTB. Pour la prédiction de la malignité et du grade, nous obtenons de meilleurs résultats en utilisant l'algorithme sur les trois tâches principales à la fois plutôt que sur une seule. La justesse balancée passe ainsi de 75.46% à 75.96% pour la classification de la malignité et de 68.09% à 69.57% pour le grade. Pour la classification du sous-type, notre modèle enregistre une perte de 3.43%, devenant ainsi moins bon que le modèle à tâche unique avec un écart de 0.7%. Il en va de même pour les autres mesures qui passe de 0.825 à 0.800 pour le SSC, de 0.820 à 0.793 pour le score F1 et de 0.521 à 0.460 pour le CCM. Globalement, cette configuration du LTB s'en sort tout de même mieux que nos réseaux de neurones à tâche unique pour au moins deux des trois tâches de classification.

5.4 Résumé des résultats

Dans cette dernière section du chapitre 4, nous résumerons les résultats de nos différents modèles multitâches pour les comparer à notre réseau de neurones à tâche unique. Nous utiliserons une mesure spécifique à l'apprentissage multitâche et présentée par Vandenhende et al. [133]. Cette mesure donnée par l'équation suivante :

$$\Delta_{\text{MT}} = \frac{1}{K}\sum_{k=1}^{K}(-1)^{l_k}\frac{m_k - u_k}{u_k}, \tag{5.1}$$

a pour objectif de comparer les performances d'un modèle multitâche entraîné pour K tâches différentes par rapport aux modèles à tâche unique. Dans cette équation, m_k représente la performance du modèle multitâche pour la tâche k, alors que u_k représente celle du modèle à tâche unique. La constante l_k est égale à 1 si la mesure utilisée pour la tâche k est meilleure lorsqu'elle est plus basse, comme l'erreur moyenne au carré, et 0 lorsqu'il s'agit d'une mesure que l'ont veut maximiser, comme

Tableau 5.14 – Résumé des meilleurs résultats obtenus pour chaque type de réseaux neuronaux

Modèle	Malignité	Sous-type	Grade	Δ_{MT}
ResNet (tâche unique)	74.58% (0.00%)	76.44% (0.00%)	69.50% (0.00%)	0.000
Partage forcé	78.30% (▲ 3.58%)	75.56% (▼ 0.88%)	**74.81%** (▲ **5.31%**)	**+0.038**
Partage incité	77.73% (▲ 3.15%)	76.36% (▼ 0.08%)	67.20% (▼ 2.30%)	+0.003
RAM	78.86% (▲ 4.28%)	79.08% (▲ 2.64%)	71.05% (▲ 1.55%)	**+0.038**
LTB	**79.15%** (▲ **4.59%**)	77.51% (▲ 1.07%)	69.07% (▼ 0.43%)	+0.023
Partage forcé (avec radiomiques)	78.01% (▲ 3.43%)	76.32% (▼ 0.12%)	69.88% (▲ 0.38%)	+0.017
LTB (avec radiomiques)	75.96% (▲ 1.38%)	**79.17%** (▲ **2.73%**)	69.57% (▲ 0.07%)	+0.018

Afin de simplifier la comparaison, seules la justesse balancée et la mesure multitâche sont utilisées. Moyennes des performances sur 5 entraînements.

c'est le cas avec la justesse balancée. Dans notre cas, nous avons choisi de calculer les performances multitâches en utilisant la justesse balancée comme mesure pour chacune des tâches.

La table 5.14 montre les meilleurs résultats obtenus par chacun de nos modèles en n'observant que la justesse balancée. Pour la classification de la malignité, c'est l'algorithme Learning-To-Branch qui obtient la meilleure justesse balancée avec 79.15%, soit une amélioration de 4.59% par rapport à notre modèle à tâche unique. Pour la classification du sous-type, c'est le LTB utilisant des tâches auxiliaires qui montre le plus grand gain, avec un score de 79.17%, soit 2.73% de plus. Pour la prédiction du grade, c'est notre modèle de partage forcé qui obtient de loin le résultat le plus élevé avec 74.81%, portant le gain à 5.31%. Nous aimerions faire remarquer au lecteur que le RAM a obtenu la seconde place pour les trois tâches de classification. En effet, la justesse balancée maximale atteinte par ce modèle est de 78.86%, 79.08% et 71.05% pour la classification de la malignité, du sous-type et du grade respectivement. Ce sont les modèles partage forcé et RAM qui obtiennent la meilleure mesure multitâche, soit un Δ_{MT} de 0.038 dans les deux cas.

Les résultats pouvant varier considérablement en fonction de la combinaison de tâches choisies, nous avons décidé de résumer la moyenne des justesses balancées obtenue par nos modèles multitâches dans le tableau 5.15. Comme on peut le voir, ces

Tableau 5.15 – Résumé de la moyenne des résultats pour chaque type de réseaux neuronaux

Modèle	Malignité	Sous-type	Grade	Δ_{MT}
ResNet (tâche unique)	74.58% (0.00%)	76.44% (0.00%)	69.50% (0.00%)	0.000
Partage forcé	75.17% (▲ 0.59%)	73.94% (▼ 2.50%)	**71.29%** (▲ **1.76%**)	+0.000
Partage incité	76.54% (▲ 1.96%)	75.79% (▼ 0.65%)	65.70% (▼ 3.80%)	−0.012
RAM	**77.20%** (▲ **2.62%**)	**78.04%** (▲ **1.60%**)	69.92% (▲ 0.42%)	**+0.021**
LTB	77.02% (▲ 2.44%)	76.98% (▲ 0.44%)	67.66% (▼ 1.84%)	+0.004
Partage forcé (avec radiomiques)	77.07% (▲ 2.49%)	73.55% (▼ 2.89%)	67.98% (▼ 1.52%)	−0.009
LTB (avec radiomiques)	75.71% (▲ 1.13%)	77.46% (▲ 1.02%)	68.83% (▼ 0.67%)	+0.006

Pour chacun des types de réseaux neuronaux étudiés, la moyenne des résultats est calculée par rapport aux combinaisons de tâches. Afin de simplifier la comparaison, seules la justesse balancée et la mesure multitâche sont utilisées. Moyennes des performances calculées à partir des tableaux 5.4, 5.6a, 5.8a, 5.10a, 5.11a et 5.12a à 5.13b.

résultats très différents changent considérablement la façon d'analyser nos modèles. Ici, c'est le RAM qui obtient le plus souvent les meilleurs résultats avec une justesse balancée de 77.20%, de 78.04% et de 69.92% pour la prédiction de la malignité, du sous-type et du grade respectivement. C'est aussi celui qui obtient la meilleure mesure multitâche avec 0.021. En fait, il s'agit de la seule architecture à avoir enregistré un gain pour les trois tâches, même s'il arrive en deuxième place pour la classification du grade. La première place étant toujours occupée par le modèle de type partage forcé avec 71.29%. Le deuxième modèle qui réussi à s'en sortir le mieux est le LTB qui, avec ou sans l'utilisation de caractéristiques radiomiques comme tâches auxiliaires, arrive à proposer de meilleures performances pour deux des trois tâches. Plus spécifiquement, c'est lorsque cet algorithme est utilisé avec les tâches secondaires qu'il est le plus intéressant, puisqu'il obtient un gain moyen de 0.59% par tâche, alors que sa contrepartie sans tâches auxiliaires montre un gain moyen de 0.35%. Toutefois, sa performance multitâche selon la mesure 5.1 est de seulement 0.006, soit loin derrière le RAM.

Chapitre 6

Discussion

Le chapitre qui suit présentera l'analyse des résultats obtenus au chapitre précédent ainsi que des expériences supplémentaires. En premier lieu, nous ferons un retour sur nos modèles à tâche unique et la pertinence de l'imagerie médicale pour classification de tumeur rénale. Dans un second temps, nos modèles multitâches entraînés sans tâches auxiliaires seront comparés entre eux. En troisième lieu, nous discuterons des caractéristiques radiomiques comme tâches auxiliaires dans un contexte d'apprentissage multitâche. Pour finir, nous aborderons l'effet de régularisation de l'apprentissage multitâche et l'instabilité de nos résultats.

6.1 Modèles à tâches uniques

À la section 5.1, nous avons évalué les performances des modèles de type MVS utilisant des données cliniques et des modèles de type réseaux neuronaux utilisant les IRM abdominales de modalités T1C et T2 des patients, ainsi que les masques segmentant les tumeurs. Pour rappel, les trois tâches de classifications étaient les suivantes :

1. Déterminer si la tumeur rénale est maligne ou bénigne.
2. Déterminer le sous-type des tumeurs malignes (cellules claires ou papillaires)
3. Déterminer le grade des tumeurs malignes (grade faible ou grade élevé)

Puisqu'il s'agit de problèmes de classification binaire, nos modèles ne devraient pas obtenir une justesse balancée inférieure à 50%. Ainsi, par les résultats obtenus avec nos MVS, soit 72.33%, 58.61% et 61.00% pour la classification de la malignité, du sous-type et du grade respectivement, nous pouvons en déduire que les données tabulaires ne semblent pas très appropriées pour ces deux dernières tâches, alors que la prédiction de la malignité semble obtenir un score suffisant pour former un modèle de base en utilisant seulement trois variables (l'âge, le sexe et la taille de la tumeur).

En prenant en compte les tableaux 4.1 à 4.3, qui montrent la corrélation entre chaque tâche et les variables cliniques, ces résultats ne sont guère surprenant. Alors qu'il existe des corrélations de l'ordre de ± 0.3 entre la classification de la malignité et les variables cliniques, elles atteignent à peine ± 0.16 pour la classification du sous-type. Pour la classification du grade, la valeur absolue de la corrélation entre cette tâche et les variables cliniques utilisées varie entre 0.11 et 0.36. Nous pensions que de meilleur résultats auraient pu être obtenus en n'utilisant que les variables dont la corrélation variait entre 0.28 et 0.32, mais des expériences supplémentaires ont montré que la justesse balancée descendait à 0.57% en procédant ainsi.

6.1.1 Usage de données imagées pour la classification des tumeurs rénales

Le premier objectif que nous nous étions fixé pour cette étude était de déterminer si les données imagées étaient mieux adaptées que les données cliniques pour la classification des tumeurs rénales. Les résultats obtenus par nos réseaux de neurones inspiré du ResNet semblent confirmer cette hypothèse. Comme mentionné à la section 5.1.2, nos modèles arrivent à une justesse balancée de 76.44% (+17.83%) pour la classification du sous-type et de 69.50% (+8.50%) pour celle du grade. Ce qui démontre l'efficacité de ce type de modèle et de l'imagerie. Pour la classification de la malignité, nous n'avons obtenu qu'un gain modeste au niveau de la justesse balancée avec 74.58% (+2.25%).

Nos réseaux de neurones possèdent toutefois un désavantage face aux MVS : les résultats obtenus sont instables. Tel que mentionné à l'annexe B.1, nos résultats varient énormément d'un entraînement à l'autre ; en particulier pour la prédiction

du sous-type. Même en gardant la même séparation entraînement/validation/test, les résultats sur l'ensemble de test pouvaient varier de ±10%. Par exemple, pour nos expérimentations sur l'ensemble de retenue, l'écart-type de la justesse balancée était de ±3.66%, ±4.77% et ±3.66% pour la prédiction de la malignité, du sous-type et du grade respectivement. C'est cette observation qui nous a amené à toujours faire la moyenne des performances sur dix entraînements durant nos expérimentations préliminaires et cinq fois durant notre recherche d'hyperparamètres et nos expériences finales.

6.1.2 Utilisation conjointe de l'imagerie et des données tabulaires

Si l'on considère que la taille de la tumeur fait partie des trois variables cliniques utilisées par notre MVS pour la prédiction de la malignité, nous pouvons émettre l'hypothèse selon laquelle notre ResNet a appris à extraire cette caractéristique des images IRM et que cette architecture s'en est elle aussi servi pour classifier les tumeurs. Toutefois, il serait beaucoup plus surprenant que le modèle soit capable d'estimer l'âge et le sexe à partir des images IRM T1C et T2 rognées autour de la tumeur du patient. Si ce n'est pas le cas, alors le modèle a réussi à apprendre d'autres caractéristiques lui permettant de mieux classifier la tumeur.

Pour vérifier si c'est le cas ou non, nous avons ajouté les variables cliniques à notre réseau de neurones afin de voir s'il y a un gain de performance significatif. Dans le nouveau modèle, ces variables ont été concaténées aux caractéristiques apprises par

Tableau 6.1 – Résultats de l'utilisation des données cliniques pour la prédiction de la malignité sur l'ensemble d'apprentissage

Avec données cliniques	SSC	Justesse balancée	Score F1	CCM
Non	0.797 ± 0.017	$73.38\% \pm 2.42\%$	0.771 ± 0.031	0.455 ± 0.048
Oui	0.814 ± 0.022	$74.75\% \pm 2.18\%$	0.786 ± 0.019	0.484 ± 0.040

Des hyperparamètres de base ont été utilisés. Moyenne des résultats pour 10 entrainements avec une séparation entrainement/validation/test aléatoire. SSC : Surface sous la courbe. CCM : Coefficient de corrélation de Matthews.

les couches de convolutions, c'est-à-dire, juste avant la couche pleinement connectée qui sert de classifieur. Les résultats du tableau 6.1 montrent qu'il y a un gain certain à ajouter les données cliniques à notre réseau de neurones. Cependant, l'amélioration reste très modeste, ce qui nous laisse trois hypothèses pour expliquer ce phénomène.

1. L'utilisation des données tabulaires cause davantage de sur-apprentissage.
2. Notre modèle identifie déjà des caractéristiques à partir des images qui sont très corrélées avec l'âge et le sexe.
3. L'âge et le sexe du patient n'ont que peu d'importance pour la prédiction de la malignité.

Nous pouvons écarter la première hypothèse puisque la justesse balancée sur l'ensemble d'entraînement est de 81.45% pour notre modèle utilisant aussi les données cliniques, soit 2.62% de moins que pour le modèle utilisant seulement les images. Il ne reste donc que les hypothèses 2 et 3 qui sont plausibles. Si le modèle identifie déjà des caractéristiques qui sont très corrélées avec l'âge et le sexe, comme c'est probablement le cas avec la taille de la tumeur, cela ne les empêches pas d'être un peu utile. De cette manière, en les retirants, nous devrions voir une baisse de performance sans descendre plus bas que ceux de notre modèle n'utilisant pas de variable clinique. Si au contraire, l'âge et le sexe n'ont que peu d'importance, alors les retirer ne devrait pas affecter les résultats. On pourrait donc penser qu'une expérience supplémentaire où ces variables seraient exclues permettrais de départager les deux hypothèse. mais l'instabilité des résultats tels que mentionné un peu plus haut ainsi que l'écart déjà tenu entre notre modèle utilisant aussi les données et celui n'en n'utilisant pas, rendrait les analyses difficile. Il serait donc inapproprié de tirer des conclusions ou même d'émettre des suppositions de cette expérience et c'est pourquoi elle n'a pas été faite.

6.2 Architectures et algorithme multitâche

La section 5.2 plus haute présente les résultats que nous avons obtenus avec nos différents modèles multitâches lorsqu'ils sont entraînés pour des combinaisons d'au moins deux tâches principales. Nous aborderons dans cette section trois points importants concernant ces résultats. Soit, la variance des résultats par rapport à la

combinaison de tâches choisie, les raisons qui expliquent les performances du réseau d'attention multitâche (RAM) et pour finir, la pertinence de l'apprentissage multitâche dans le cas où le modèle n'est entraîné que pour deux des trois tâches.

6.2.1 Influence du choix de combinaison des tâches.

Lors de nos expérimentations avec les modèles multitâches, nous avons fait notre recherche d'hyperparamètres seulement avec des modèles entraînés sur les trois tâches principales afin de sauver du temps. Ensuite, la meilleure configuration trouvée a été utilisée pour entraîner nos modèles sur toutes les combinaisons d'au moins deux tâches principales (quatre combinaisons au total). En produisant les résultats sur l'ensemble de retenue, nous avons remarqué que certains de nos modèles étaient très sensibles à la combinaison de tâches choisie. Par exemple, comme on peut le voir dans le tableau 6.2, bien que notre modèle de partage forcé ait obtenu en moyenne une justesse balancée de 71.29% pour la classification du grade, cette valeur variait entre 68.52% et 74.81% produisant ainsi un écart-type de 2.62%. La stabilité des résultats est donc très importante ici afin d'éviter de devoir effectuer quatre recherches d'hyperparamètres différentes par modèles.

Si l'on fait la moyenne des écarts-types pour chaque tâche, le modèle de type partage forcé est de loin le plus instable avec 2.42%. Sans surprise, c'est notre modèle partage incité qui est le moins dépendant avec un écart-type moyen de 0.95%. Ce résultat est logique puisqu'aucune couche n'est partagée dans l'architecture de partage incité. Pour les deux autres modèles, l'écart-type moyen est de 1.34% et 1.28% pour

Tableau 6.2 – Moyenne et écart-type des résultats sur l'ensemble de retenue obtenus par les modèles multitâches

Modèle	Malignité	Sous-type	Grade
Partage forcé	75.17% (±2.50%)	73.94% (±2.14%)	71.29% (±2.62%)
Partage incité	76.54% **(±0.94%)**	75.79% (±0.77%)	65.70% (±1.14%)
RAM	77.20% (±1.87%)	78.04% (±1.07%)	69.92% **(±1.08%)**
LTB	77.02% (±2.18%)	76.98% **(±0.53%)**	67.66% (±1.14%)

Moyennes des performances présentées dans les tableaux 5.6a, 5.8a, 5.10a et 5.11a.

le RAM et le LTB respectivement.

6.2.2 Réseau d'attention multitâche et modules d'attentions

Les résultats obtenus à la section 5.4 du chapitre précédent nous laissent penser que le RAM est un bon candidat parmi les modèles testés. Toutefois, ce type d'architecture possède une particularité : il emploie des modules d'attention. Il est donc possible que le gain de performance par rapport à l'architecture à tâche unique puisse être expliqué par l'utilisation de ces modules plutôt que par l'apprentissage multitâche. Pour vérifier, nous avons décidé de créer une version à tâche unique du RAM, le réseau d'attention à tâche-unique (RATU). Pour comparer les résultats sur l'ensemble de retenue, nous aurions eu besoin d'effectuer une recherche d'hyperparamètres pour chaque tâche, ce qui aurait demandé trop de temps de calcul. Nous avons plutôt décidé d'utiliser les hyperparamètres de base (voir annexe C) pour comparer les deux architectures de réseau de neurones à tâche unique.

Les résultats présentés dans le tableau 6.3 n'infirment pas notre hypothèse. Le RATU obtient de meilleurs résultats pour la classification de la malignité et du sous-type. La justesse balancée pour ces tâches respectives est de 75.53% et 75.02%, soit une amélioration de 2.15% et de 1.82%. Le CCM (coefficient de corrélation de Matthews) affiche aussi des gains avec 0.500 plutôt 0.455 pour la malignité et 0.451 au lieu de

Tableau 6.3 – Résultats du réseau d'attention à tâche unique sur l'ensemble de test

Module d'attention	tâche	SSC	Justesse balancée	Score F1	CCM
Non	Malignité	0.797 ± 0.017	$73.38\% \pm 2.42\%$	0.771 ± 0.031	0.455 ± 0.048
	Sous-type	0.812 ± 0.037	$73.20\% \pm 3.32\%$	0.788 ± 0.055	0.417 ± 0.060
	Grade	$\mathbf{0.704 \pm 0.063}$	$\mathbf{68.23\% \pm 5.34\%}$	$\mathbf{0.580 \pm 0.071}$	$\mathbf{0.362 \pm 0.106}$
Oui	Malignité	$\mathbf{0.822 \pm 0.023}$	$\mathbf{75.53\% \pm 2.83\%}$	$\mathbf{0.792 \pm 0.034}$	$\mathbf{0.500 \pm 0.057}$
	Sous-type	$\mathbf{0.815 \pm 0.047}$	$\mathbf{75.02\% \pm 4.80\%}$	$\mathbf{0.807 \pm 0.050}$	$\mathbf{0.451 \pm 0.090}$
	Grade	0.680 ± 0.056	$65.58\% \pm 4.23\%$	0.560 ± 0.049	0.303 ± 0.083

Des hyperparamètres de base ont été utilisés. Moyenne des résultats pour 10 entrainements avec une séparation entrainement/validation/test aléatoire. SSC : Surface sous la courbe. CCM : Coefficient de corrélation de Matthews.

0.417 pour le sous-type. Pour ce qui est de la prédiction du grade, le RATU s'en sort moins bien que sa contrepartie qui n'emploie pas de module d'attention. La justesse balancée passe ainsi de 68.23% à 65.58%, alors que le CCM passe de 0.362 à 0.303. Suite à ces résultats, il n'est donc plus clair si les gains de performances, pour la classification de la malignité et du sous-type, obtenus par le RAM sont dus en partie à l'apprentissage multitâche ou totalement à l'utilisation de modules d'attention.

6.3 Utilisation des caractéristiques radiomiques comme tâches auxiliaires synthétiques

La dernière hypothèse que nous voulions explorer dans ce mémoire était l'utilisation des caractéristiques radiomiques pour créer des tâches auxiliaires qui pourraient bénéficier à l'apprentissage d'une ou plusieurs tâches principales. Dans cette section, nous aborderons les difficultés rencontrées à l'identification des ensembles de tâches auxiliaires optimales. Nous terminerons sur les bénéfices que présentait cette stratégie.

6.3.1 Identification des caractéristiques radiomiques à utiliser

Avec plus de 300 caractéristiques radiomiques par patient (150 par image), il était pratiquement impossible de tester toutes les combinaisons existantes. Pour pallier ce problème, nous avons dans un premier temps tenté d'employer l'algorithme de groupement de tâches par affinité (GTA), avant de nous tourner vers une solution utilisant l'algorithme LTB. Aucune de ces deux stratégies ne nous a permis d'identifier de manière fiable un ensemble de tâches auxiliaires optimales pour nos tâches principales.

Identification des tâches auxiliaires à l'aide du GTA

L'algorithme de GTA a été notre premier espoir de pouvoir identifier efficacement les caractéristiques radiomiques à utiliser comme tâches auxiliaires pour améliorer les performances de nos modèles. Pour l'utiliser, nous n'avons eu qu'à faire des change-

ments mineurs dans l'algorithme. En effet, plutôt que de mesurer l'affinité entre toutes les tâches, nous avions seulement besoin de mesurer l'affinité entre la tâche principale et toutes les tâches secondaires, en vérifiant l'écart entre la fonction de coût de la tâche principale avant et après mise à jour des poids du modèle par rapport à une tâche auxiliaire. Nous avons suivi les recommandations de l'article original [41] en mesurant l'affinité inter-tâche sur l'ensemble d'entraînement, et ce seulement une fois toutes les cinq itérations, afin d'accélérer l'entraînement du modèle. Nous avons entraîné chaque tâche principale avec toutes les caractéristiques radiomiques en alternant entre les modalités (T1C et T2) afin d'éviter d'avoir plus 300 tâches auxiliaires. Nous avons utilisé un modèle de type partage forcé pour identifier les tâches auxiliaires les plus bénéfiques pour chaque problème de classification.

Il est important de noter que l'algorithme était très lent puisque le calcul de l'affinité inter-tâche nécessitait autant de propagation avant et arrière qu'il y avait de tâches auxiliaires. L'exécution de l'algorithme pouvait prendre entre trois et cinq heures. Afin de vérifier la stabilité des résultats, nous avons utilisé cet algorithme dix fois pour chaque tâche principale.

Comme le montre le tableau 6.4, pour les trois tâches de classification il n'est jamais arrivé qu'une tâche auxiliaire partage une affinité positive avec une tâche principale plus de trois fois sur dix. Ainsi, nous pouvons dire que selon cet algorithme, toutes les tâches auxiliaires ont davantage de chance de nuire aux performances que

Tableau 6.4 – Répartition des tâches auxiliaires avec le groupement de tâche par affinité

(a) Affinité mesurée jusqu'au meilleur cycle d'entrainement

	Malignité		**Sous-type**		**Grade**	
Fréquence des affinités positives	Effectif	Fréquence relative	Effectif	Fréquence relative	Effectif	Fréquence relative
0	0	0%	5	3.42%	41	28.08%
10%	4	2.74%	63	43.15%	84	57.53%
20%	124	84.93%	66	45.20%	21	14.38%
30%	18	12.33%	12	8.22%	0	0.00%
40% et +	0	0.00%	0	0.00%	0	0.00%

Tableau 6.4 – Répartition des tâches auxiliaires avec le groupement de tâche par affinité

(b) Affinité mesurée jusqu'au dernier cycle d'entrainement

	Malignité		**Sous-type**		**Grade**	
Fréquence des affinités positives	Effectif	Fréquence relative	Effectif	Fréquence relative	Effectif	Fréquence relative
0	146	100%	118	80.82%	146	100%
10%	0	0.00%	28	19.18%	0	0.00%
20%	0	0.00%	0	0.00%	0	0.00%
30%	0	0.00%	0	0.00%	0	0.00%
40% et +	0	0.00%	0	0.00%	0	0.00%

Distribution de 146 tâches auxiliaires de type prédiction de caractéristiques radiomiques selon la fréquence de leur affinité positive avec chaque tâche principale sur 10 entrainements.

d'aider. De plus, parmi les tâches auxiliaires partageant le plus souvent une affinité positive avec la classification de la malignité, une seule d'entre elles correspondait à la prédiction d'une caractéristique radiomiques de type morphologique. Cela contredit les résultats obtenus lors de notre étape de validation, qui nous indiquait que l'ensemble des caractéristiques radiomiques morphologiques était le plus adapté pour améliorer les performances du modèle pour cette tâche. Pour être certains, nous avons aussi mesuré l'affinité inter-tâche jusqu'à la fin de l'entrainement et comme le montre le tableau 6.4b, c'est encore pire. Au vu de ces résultats, nous avons décidé de ne pas poursuivre avec l'algorithme GTA pour créer l'ensemble de tâches auxiliaires optimales pour chaque tâche principale.

Identification des tâches auxiliaires par l'analyse du LTB

Comme mentionné plus haut, nous avons décidé d'analyser l'architecture apprise par l'algorithme LTB afin d'identifier les tâches auxiliaires qui sont les plus liées avec nos tâches de classification. En partant de l'hypothèse populaire selon laquelle deux tâches faiblement liées se nuiront mutuellement en se battant pour les ressources du réseau de neurones, nous croyons que l'algorithme LTB sépara le plus possible les tâches qui se nuisent le plus. Ainsi, dans l'architecture finale, les tâches auxiliaires

qui partageront le plus de couches avec la tâche principale seront celles qui auront la plus grande affinité avec cette dernière. Notre stratégie consiste donc à analyser la fréquence à laquelle une tâche auxiliaire partage $n-1$ et $n-3$ couches avec la tâche principale.

Pour cause de limite de ressources de calcul, nous n'avons pu évaluer que des groupes de dix tâches auxiliaires à la fois. Pour éviter le cas où une tâche auxiliaire soit en relation avec une seule tâche principale, mais que les tâches principales soient trop liées entre elles, nous n'avons utilisé qu'une seule tâche de classification à la fois. Chaque caractéristique radiomique ne sera associée qu'à une seule des deux modalités pour réduire de moitié le nombre de tests nécessaires. Dans nos expériences nous avons utilisé un hyperparamètre de température τ de 5.0 et un taux d'apprentissage pour les couches branchement et les autres couches de $1e-4$. L'architecture utilise six couches en parallèle afin de permettre à l'algorithme de bien séparer les tâches qui ne partagent pas d'affinité. Pour finir, nous ne présenterons les résultats que pour les tâches de prédictions du sous-type et du grade et que pour le groupe de caractéristiques morphologiques afin de resté à l'essentiel.

Comme on peut le voir dans le tableau 6.5a, la classification du sous-type semble partager davantage d'affinité avec les tâches de prédiction du volume, du déplacement du centre de masse (*Center of Mass*), de la platitude (*flatness*) et de l'élongation (*Elongation*). De plus, en dix entraînements, la fréquence à laquelle une tâche auxiliaire partage $n-1$ et $n-3$ couches avec cette tâche principale est de 26 et 17 respectivement. De son côté, la prédiction du grade semble davantage reliée aux caractéristiques de surface (*Area*), de la première mesure de compacité (*Compacity 1*) et la caractéristique radiomique *a dens conv hull*.[1] La fréquence à laquelle la classification du grade partage $n-1$ et $n-3$ avec une tâche auxiliaire est respectivement de 21 et 9.

De cet ensemble de dix tâches auxiliaires, nous en avons remplacé deux afin de vérifier la stabilité des résultats trouvés. Nous avons remplacé la prédiction du volume par celle de la longueur de l'axe majeur (*pca major*), ainsi que la prédiction du déplacement du centre de masse avec celle de l'asphéricité. Le tableau 6.5b montre des changements très importants dans la relation entre les tâches principales et les

1. Pour plus d'information, voir le manuel de l'IBSI[158].

Tableau 6.5 – Analyse d'affinité entre les tâches à l'aide de l'algorithme Learning-To-Branch.

(a) Affinité entre 2 des tâches de classification et les tâches auxiliaires du premier groupe

	Sous-type		**Grade**	
	Nombre de couches partagées		**Nombre de couches partagées**	
Tâche	$n-1$	$n-3$	$n-1$	$n-3$
T1C *Volume*	4	2	1	1
T2 *Area*	3	1	3	1
T1C *Compacity* 1	2	1	3	1
T2 *Sphericity*	2	1	0	1
T1C *Center of Mass*	4	2	0	2
T2 *Elongation*	3	3	3	0
T1C *Flatness*	4	3	2	1
T2 *v dens aabb*	1	2	1	2
T1C *a dens mvee*	2	1	3	0
T2 *a dens conv hull*	1	1	5	0
Total	26	17	21	9

Tableau 6.5 – Analyse d'affinité entre les tâches à l'aide de l'algorithme Learning-To-Branch.

(b) Affinité entre 2 des tâches de classification et les tâches du second groupe

	Sous-type		**Grade**	
	Nombre de couches partagées		**Nombre de couches partagées**	
Tâche	$n-1$	$n-3$	$n-1$	$n-3$
T1C *pca major*	1	2	0	1
T2 *area*	2	3	2	1
T1C *Compacity 1*	2	1	3	2
T2 *sphericity*	5	1	4	0
T1C *asphericity*	2	1	2	2
T2 *elongation*	1	1	0	2
T1C *flatness*	0	4	3	1
T2 *v dens aabb*	0	1	2	2
T1C *a dens mvee*	2	1	2	0
T2 *a dens conv hull*	1	1	0	0
Total	16	16	17	14

Analyse d'affinité entre les tâches de classification du sous-type et du grade et 10 tâches auxiliaires de prédiction de caractéristiques radiomiques morphologiques. Fréquence sur 10 entraînements à laquelle chaque tâche partage $n-1$ et $n-3$ couches avec chaque tâche principale. Le nom des caractéristiques radiomiques est laissé en anglais afin de permettre au lecteur de se référer plus facilement au manuel de référence de l'IBSI.

tâches secondaires. Pour commencer, la prédiction de la platitude et de l'élongation ne semble plus partager autant d'affinité avec la prédiction du sous-type. De même pour la caractéristique radiomique *a dens conv hull* avec la classification du grade. À l'inverse, la prédiction de la sphéricité semble beaucoup plus liée à nos deux tâches principales.

Pour ce qui est des performances sur l'ensemble de test, le changement de tâches auxiliaires a été négative pour la justesse balancée de la prédiction du sous-type, qui passe de 74.7% à 72.5%. C'est tout le contraire pour la classification du grade qui est passé de 62.4% à 63.8%. Ces changements dans les résultats étaient toutefois

assez prévisibles puisque, durant nos premières expériences, les deux tâches auxiliaires retirées semblaient partager beaucoup d'affinité avec la classification du sous-type, mais peu avec celle du grade.

Malgré tout, nous avons choisi de ne pas poursuivre avec cette méthode pour l'identification des tâches auxiliaires à utiliser, puisque l'affinité, tel que nous la mesurions en analysant le LTB, semble très sensible à l'ensemble des tâches auxiliaires utilisé. En d'autres termes, la relation entre une tâche A et une tâche B n'est peut-être pas aussi bien exploitée lorsque la tâche D est présente que lorsque cette dernière est remplacée par la tâche C. Donc, si l'on faisait l'expérience plus haut pour chaque famille de caractéristiques radiomiques et que prenions les dix caractéristiques partageant le plus souvent $n-1$ couches avec chaque tâche de classification, il n'est pas garanti que l'affinité entre ces tâches auxiliaires et la tâche principale soit préservée.

6.3.2 Bénéfice de l'utilisation de tâches auxiliaires

Nous avons pu voir au chapitre précédent que l'utilisation des caractéristiques radiomiques comme tâches auxiliaires pouvait être bénéfique pour de multiples tâches de classification des tumeurs rénales. L'idée derrière notre stratégie est basé sur les résultats des travaux de Caruana [21], qui démontre que certaines caractéristiques sont plus utiles comme tâches auxiliaires que comme variables d'entrées. Les caractéristiques radiomiques forment un ensemble de plus de 150 tâches auxiliaires qui pourrait être potentiellement bénéfique. L'hypothèse est donc que l'apprentissage de la bonne combinaisons de tâches auxiliaires permettrais à notre modèle de découvrir des caractéristiques pouvant aider à la classification de la malignité, du sous-type et du grade. Par exemple, forcer un modèle à apprendre les caractéristiques morphologique d'une tumeur pourrait l'amener à mieux décortiquer la structure de la région d'intérêt et ainsi à prédire plus facilement la malignité.

Nos expériences ont montré que ce sont surtout les tâches de prédiction de la malignité et du sous-type qui bénéficiaient des tâches auxiliaires avec des gains de performances de 1 à 3% (voir section 5.4). Il faut toutefois rappeler que l'objectif principal de ce travail était de proposer un modèle multitâche pouvant effectuer au moins deux des trois tâches de classification en même temps. Ainsi, il pourrait être

légitime de remettre en cause la pertinence des modèles entraînés à effectuer un ensemble de tâches radiomiques et une seule tâche principale.

Si l'on se concentre seulement sur nos modèles qui ont été entraînés à effectuer à la fois les trois tâches de classification et un ensemble de tâches auxiliaires, on se rappelle que les résultats étaient bien moins bons. Par exemple, pour le modèle partage forcé, la justesse balancée des trois tâches principales était affectée négativement, avec des pertes allant jusqu'à 5.54% pour celle du sous-type. Pour les modèles obtenus à l'aide de l'algorithme LTB, seule la classification du sous-type voyait sa justesse balancée amoindrie de 3.43%, alors que les deux autres tâches montraient des performances légèrement meilleures. Ces résultats ne font que tout simplement indiquer qu'il est beaucoup plus difficile de trouver un ensemble de tâches auxiliaires optimales pour les trois tâches en même temps qu'indépendamment. Malgré tout, la stabilité des résultats obtenus par le LTB ne fait que montrer, une fois de plus, la capacité de cet algorithme à s'adapter, ainsi que sa supériorité face au simple modèle de type partage forcé.

6.4 Effet de régularisation de l'apprentissage multitâche

Le développement de nos modèles multitâches s'est montré plus ardu qu'il n'y parait. Lorsque nous travaillions sur l'ensemble d'apprentissage, nous n'arrivions jamais à obtenir de meilleurs résultats avec les modèles multitâches alors que ceux des modèles à tâche unique semblaient imbattables. Les résultats sur l'ensemble de test final, nous ont à la fois surpris et rendus perplexes. Nous souhaitons donc prendre quelques lignes pour mettre en évidence les éléments qui nous ont fait douter de la fiabilité de nos résultats.

6.4.1 Résultats de la seconde validation

Ce qui nous a la puce à l'oreille en premier, ce sont probablement les résultats de notre seconde étape de validation. Pour rappel, l'instabilité de nos résultats faisait en sorte que nous avions besoin d'effectuer une seconde étape de validation après la re-

Tableau 6.6 – Résumé des meilleurs résultats pour chaque type de réseaux neuronaux sur l'ensemble d'apprentissage

Modèle	Malignité	Sous-type	Grade	Δ_{MT}
ResNet (tâche unique)	**76.17% (0.00%)**	77.42% (0.00%)	66.95% (0.00%)	0.000
Partage forcé	73.32% (▼ 2.85%)	73.49% (▼ 3.93%)	68.86% (▲ 1.91%)	−0.020
Partage forcé (radiomiques)	76.03% (▼ 0.14%)	76.64% (▼ 0.78%)	**69.47% (▲ 2.52%)**	**+0.008**
Partage incité	74.30% (▼ 1.87%)	**77.97% (▲ 0.55%)**	67.36% (▲ 0.41%)	−0.003
RAM	74.78% (▼ 1.39%)	74.19% (▼ 3.23%)	65.21% (▼ 1.74%)	−0.029

Afin de simplifier la comparaison, seules la justesse balancée et la mesure multitâche sont utilisées. Moyennes des performances sur 10 entrainements.

cherche d'hyperparamètres. Cette seconde étape de validation consistait à reprendre les cinq meilleures configurations trouvées à l'aide de notre recherche d'hyperparamètres et de faire dix entraînements pour chaque configuration, plutôt que cinq, et de les comparer en faisant la moyenne des résultats. À chaque fois, cela changeait le classement des performances. Nous avons fait cette seconde étape de validation pour toutes les architectures à l'exception de l'algorithme LTB, qui lui a été entraîné en n'utilisant que les hyperparamètres de base.

Le tableau 6.6 montre les résultats de notre seconde étape de validation. Comme l'on peut le remarquer, les résultats sont beaucoup moins encourageants que ceux de la section 5.4. Ici, la plupart de nos modèles échouent à obtenir de meilleurs résultats que notre architecture à tâche unique. Plus particulièrement, le RAM est celui qui obtient en moyenne la justesse balancée la plus basse, soit 71.39%, ne réussissant pas à dépasser le modèle à tâche unique pour aucune des trois tâches. C'est d'ailleurs ce modèle qui obtient la mesure multitâche la plus basse avec −0.029. À l'inverse, le modèle de type partage forcé est le seul a obtenir une valeur positive avec la mesure multitâche, soit 0.008. Comme l'on peut le remarquer, cela est principalement dû à l'écart de performance pour la prédiction du grade. L'architecture de type partage incité, pour sa part, est celle qui arrive le plus souvent à obtenir des gains, avec deux tâches sur trois, mais qui malheureusement ne réussit pas à compenser suffisamment les pertes sur le problème de classification de la malignité, obtenant ainsi un score de −0.003 pour la mesure multitâche.

Ces résultats contrastent énormément avec ceux obtenus au chapitre précédent. Ici, le RAM n'est plus du tout le modèle ayant les meilleurs résultats. Toutefois, comme mentionné à la sous-section 5.2.3, il y a eu une erreur au niveau de la recherche d'hyperparamètres pour ce modèle. En effet, seulement une seule configuration de blocs a été testée. Il s'agit de la configuration dont les deux premiers niveaux sont composés de blocs de type pré-activation et dont les blocs des deux derniers niveaux sont de type post-activation. Pour les modèles à tâche unique, la configuration de bloc optimale pour les trois tâches était l'inverse. Soit, des blocs de type post-activation suivie de ceux de type pré-activation. Cela pourrait expliquer les mauvais résultats du RAM sur l'ensemble d'apprentissage, mais les bons résultats sur l'ensemble de test nous ont fait renoncer à l'idée de recommencer la recherche d'hyperparamètre et la seconde phase de validation pour ce modèle.

Malgré tout, cela ne peut expliquer les bons résultats du RAM et des autres modèles sur l'ensemble de test. Le fait que durant la recherche d'hyperparamètre et la seconde phase de validation, les modèles multitâches, à l'exception de ceux utilisant des caractéristiques radiomiques comme tâches auxiliaires, ont toujours été entraînés sur les trois tâches en même temps et jamais d'autres combinaisons de tâches, aurait

Tableau 6.7 – Résumé des résultats des modèles multitâche entraînés pour les 3 tâches simultanément

Modèle	Malignité	Sous-type	Grade	Δ_{MT}
ResNet (tâche unique)	74.58% (0.00%)	76.44% (0.00%)	69.50% (0.00%)	0.000
Partage forcé	72.17% (▼ 2.41%)	75.56% (▼ 0.88%)	**74.81%** (▲ **5.31%**)	+0.011
Partage incité	77.73% (▲ 3.15%)	74.71% (▼ 1.73%)	64.42% (▼ 5.08%)	−0.020
RAM	**78.15%** (▲ **3.57%**)	**79.08%** (▲ **2.64%**)	68.47% (▼ 1.03%)	**+0.022**
LTB	74.03% (▼ 0.55%)	77.51% (▲ 1.07%)	69.07% (▼ 0.43%)	+0.000
Partage forcé (avec radiomiques)	76.13% (▲ 1.55%)	70.78% (▼ 5.66%)	66.08% (▼ 3.42%)	−0.034
LTB (avec radiomiques)	75.96% (▲ 1.38%)	75.74% (▼ 0.70%)	69.57% (▲ 0.07%)	+0.003

Seul les résultats des entrainement où le réseau de neurones a été entrainé sur les trois tâches de classification en même temps ont été pris en comptes. Afin de simplifier la comparaison, seules la justesse balancée et la mesure multitâche sont utilisées. Moyennes des performances sur cinq entrainements.

aussi pu expliquer cet écart. Or comme le montre le tableau 6.7, les divergences sont encore très présentes. Le RAM est encore le modèle obtenant les meilleurs résultats avec un score de 0.022 pour la mesure multitâche.

Pour ce qui est des autres modèles, les résultats se rapprochent davantage de ceux que nous avons obtenu durant la seconde phase de validation. Par exemple, le modèle de type partage forcé obtient une justesse balancée similaire pour la classification de la malignité et du sous-type sur l'ensemble de retenue, avec des écarts d'environ 1 à 2%. Pour le modèle partage forcé utilisant les caractéristiques radiomiques comme tâches auxiliaires, c'est la même histoire. Si l'on considère le fait que pour ce modèle il y a eu une recherche d'hyperparamètres par tâche principale et une pour toutes les tâches principales en même temps, alors il est possible de simplement comparer les tableaux 6.6 et 5.15 pour s'apercevoir que l'écart moyen entre les résultats de validation et ceux du test final est de 1.83%. Le modèle de type partage incité montre quant à lui un écart moyen non négligeable de 3.21%. Ainsi, nous pouvons supposer que le fait d'avoir testé différentes combinaisons de tâches sur l'ensemble de test final, permet d'expliquer les écarts de résultats pour nos deux modèles de type partage forcé seulement.

6.4.2 Hypothèses et raisonnement

Au final, nous ne voyons que trois autres raisons qui peuvent expliquer l'écart entre les résultats de test et ceux de la seconde phase de validation.

1. L'apprentissage multitâche nécessite une quantité minimale de données pour être efficace.
2. Les modèles à tâches uniques sont plus sujets au sur-apprentissage que les modèles multitâches durant la recherche d'hyperparamètre.
3. Les résultats sont influencés par la séparation des données d'apprentissage/de retenue.

La première explication ne nous semble pas plausible puisque notre ensemble d'apprentissage contenait 880 patients pour la classification de la malignité, 479 pour le sous-type et 477 pour le grade. Ce qui implique que les ensembles d'entraînement durant la seconde phase de validation étaient d'environ 563, 306 et 305 patients pour ces

trois tâches respectives. Alors que durant nos expérimentations finales elles étaient constituées d'environ 704, 383 et 382 patients. Si cette hypothèse était vérifiée, alors cela impliquerait que le nombre de données minimal pour que la classification de malignité profite de l'apprentissage multitâche, serait plus élevé que pour les deux autres tâches, alors que les résultats obtenus avec nos MVS semblent indiquer qu'il s'agit d'une tâche qui est plus simple que les deux autres, puisque la seule connaissance de l'âge, du sexe et de la taille de la tumeur sont nécessaires pour obtenir de bonnes performances.

La seconde hypothèse est quant à elle un peu plus plausible. En effet, il est connu que l'apprentissage multitâche a un pouvoir de régularisation [17, 18, 22, 160]. Dans notre cas, c'est le critère de sauvegarde qui explique le plus simplement une partie de la régularisation. Pour rappel, ce critère nous sert à déterminer à quel cycle d'entraînement le modèle est le plus performant sur l'ensemble de validation. Pour ce faire nous mesurons la moyenne géométrique des rappels sur l'ensemble de validation pour capturer le cycle pour lequel cette valeur est maximale. Lorsqu'il y a plusieurs tâches, nous calculons la moyenne géométrique des moyennes géométriques des rappels de chaque tâche pour n'avoir qu'une seule mesure à comparer. Ainsi, le modèle doit obtenir de bonnes performances pour les trois tâches à la fois. Cet effet de régularisation pourrait bien expliquer pourquoi la plupart de nos modèles multitâches n'ont pas subi trop de perte de performances sur l'ensemble de retenue par rapport à l'ensemble de test échantillonné à partir de l'ensemble d'apprentissage. Cependant, la régularisation ne peut expliquer pourquoi nos modèles ont obtenu de meilleures performances sur l'ensemble de retenue.

Notre dernière hypothèse est que nous avons été en quelque sorte chanceux lorsque nous avons fait notre séparation aléatoire des données d'apprentissage/de retenue. En effet, quoique nous ayons gardé environs 10% des données pour l'ensemble de retenue, soit 106, 61 et 60 patients pour les tâches de prédictions de la malignité, du sous-type et du grade respectivement, il est possible que cet ensemble ne soit pas suffisamment représentatif de notre ensemble de données en entier. Ce impliquerait que, par chance, nos modèles multitâches aient de meilleures performances sur l'ensemble de test final que sur l'ensemble d'apprentissage. Or, ce serait le cas à la fois pour le RAM, le modèle de partage forcé n'utilisant pas de tâche auxiliaire et celui qui est entraîné

pour prédire la malignité et un ensemble de caractéristiques radiomiques. Cela laisse sous-entendre qu'il est possible que certains ensembles de données soient plus sujets à bénéficier de l'apprentissage multitâche que d'autres. Ainsi, l'architecture du modèle et la relation entre différentes tâches ne seraient pas les seuls facteurs qui pourraient influencer les capacités de l'apprentissage multitâche à améliorer les résultats. Pour vérifier cette hypothèse, nous allons changer l'ensemble de retenue et recommencer toutes nos recherches d'hyperparamètres, ainsi que la seconde étape de validation, c'est ce qui sera discuté à la prochaine section.

6.4.3 Validation de résultats

Comme mentionné, nous avons refait toutes nos expériences en commençant par la recherche d'hyperparamètres avec une nouvelle séparation des données. Nous avons aussi refait, certaines expériences préliminaires, telle que la sélection des caractéristiques radiomiques comme tâches auxiliaires. Toutefois, nous n'avons pu tous les refaires par soucis de temps. Ainsi, certains paramètres tels que ceux déterminer par les expériences préliminaires décrites dans les section B.1 à B.3 de l'annexe B ont été préservés. Aussi, dans notre nouvelle séparation des données, nous avons utilisés 20% des données pour former l'ensemble de retenue, afin que cette ensemble soit moins

Tableau 6.8 – Résumé des meilleurs résultats pour chaque type de réseaux neuronaux avec un ensemble de retenue de 20%

Modèle	Malignité	Sous-type	Grade	Δ_{MT}
ResNet (tâche unique)	72.13% (0.00%)	73.98% (0.00%)	**65.21% (0.00%)**	**0.000**
Partage forcé	72.18% (▲ 0.05%)	72.91% (▼ 1.07%)	65.14% (▼ 0.07%)	−0.005
Partage incité	71.81% (▼ 0.32%)	71.80% (▼ 2.18%)	64.81% (▼ 0.40%)	−0.013
RAM	72.71% (▲ 0.58%)	72.31% (▼ 1.67%)	63.62% (▼ 1.59%)	−0.013
LTB	72.85% (▲ 0.72%)	**74.74%** (▲ **0.76%**)	63.70% (▼ 1.51%)	−0.001
Partage forcé (avec radiomiques)	71.70% (▼ 0.43%)	69.83% (▼ 4.15%)	64.92% (▼ 0.29%)	−0.026
LTB (avec radiomiques)	**72.96%** (▲ **0.83%**)	72.98% (▼ 1.00%)	64.24% (▼ 0.97%)	−0.006

Afin de simplifier la comparaison, seules la justesse balancée et la mesure multitâche sont utilisées. Moyennes des performances sur cinq entrainements.

sensible à la séparation des données.

Ces expériences nous ont permis de confirmer notre hypothèse selon laquelle la séparation des données avait un grand impacte sur nos résultats. Comme le montre le tableau 6.8, aucun des modèles multitâche ne réussi à surpasser nos modèles à tâche unique. Le modèle qui s'en sort le mieux est le LTB sans tâche auxiliaire avec une mesure multitâche de -0.001. Ainsi, non seulement on en déduit que nos résultats sont très sensible à la séparation des données, mais aussi que certaines séparation profitent beaucoup mieux de l'apprentissage multitâche que d'autre. En effet, le lecteur se rappellera que dans le tableau 5.14 les six modèles multitâche obtenaient une mesure multitâche supérieur à 0, alors qu'ici c'est le contraire. Cela démontre que les bénéfice de l'apprentissage multitâche ne sont pas seulement influencé par le choix de l'architecture ou encore le groupement de tâche, mais aussi le données utilisées.

Conclusion

Avec une estimation de 79 000 nouveaux cas de cancer du rein en 2022 aux États-Unis, l'intérêt pour les systèmes d'aide à la décision basés sur l'intelligence artificielle est plus fort que jamais. Si un modèle pouvant prédire la malignité d'une tumeur rénale à partir de l'IRM d'un patient permettait de mieux identifier les lésions bénignes ne nécessitant pas d'opération, la connaissance du sous-type et du grade des tumeurs malignes sont, quant à eux, pertinentes pour le pronostic du médecin clinicien. Or, il n'existe pour l'instant aucune étude portant sur la prédiction de ces trois caractéristiques à l'aide d'un seul modèle. Cette approche multitâche qui pourrait permettre d'accélérer le processus de validation et de déploiement ne doit toutefois pas être utilisée en dépit de la qualité des prédictions. Pour garder son intérêt, le modèle multitâche se doit d'être supérieur ou égal aux modèles à tâche unique en termes de performances de classification. Nous résumerons ci-dessous les contributions apportées par nos travaux suivis des connaissances que nous en avons retirées. Nous finirons en discutant de différents travaux qui pourraient suivre cette recherche.

À travers nos travaux et ce mémoire, nous avons apporté des contributions à la science avec cinq points importants.

1. Nous avons confirmé que les réseaux de neurones à convolution utilisant des IRM abdominales et le masque de segmentation de la lésion, étaient bien adaptés pour résoudre des problèmes de classification de tumeurs rénales (section 5.1). De cette manière, nous avons mis de l'avant l'intérêt à utiliser les données imagées au lieu ou en complément des données cliniques pour ce type de problème (sections 6.1.1 et 6.1.2).
2. Nous avons été les premiers à avoir utilisé l'apprentissage multitâche pour la classification de tumeurs rénales. Nos résultats ne nous ont pas permis de

conclure si cette méthode était adaptée à ce problème (sections 5.2 et 5.4).

3. Nous avons aussi implémenté plusieurs types d'architectures multitâches afin de les comparer. Cela forme une validation indépendante supplémentaire de ces modèles sur un nouvel ensemble de tâches concrètes (section 5.4).
4. Nous avons exploré une approche encore inédite qui consiste à utiliser les caractéristiques radiomiques comme tâches auxiliaires plutôt que comme des données tabulaires à donner en entrer à notre modèle (section 5.3).
5. Nous avons aussi expérimenté l'algorithme de groupement de tâches par affinité afin de sélectionner les tâches auxiliaires les plus pertinentes pour chacune de nos tâches principales (section 6.3.1). Pour la même raison, nous avons étudié une façon d'analyser les résultats de la recherche d'architecture de l'algorithme LTB (section 6.3.1). Malheureusement, ces deux techniques n'ont pas donné de résultats concluants.

Si la plupart de nos expérimentations et de nos efforts ont été détaillés dans ce mémoire, il n'en reste pas moins que plusieurs de nos idées n'ont pas été discutées, car elles n'ont été que trop légèrement explorées ou documentées par souci de concision. Parmi celles-ci, nous noterons l'utilisation d'une architecture de type réseau de capsule [111] que nous avons adaptés à notre problème. Une autre de ces idées, fût l'utilisation d'une architecture de type partage forcé renversée, où les couches partagées venaient après les couches spécifiques à chaque tâche [162]. Or tous nos travaux, nos expériences et l'analyse de nos résultats auraient été vains si nous n'avions pas été capables de partager les quelques connaissances retirées.

Pour commencer, nos résultats et leur analyse ont montré que les réseaux de neurones utilisant l'imagerie médicale possèdent un grand potentiel pour de multiples tâches de classification de tumeurs rénales. En particulier pour la prédiction du grade et du sous-type où la simple utilisation de données cliniques telles que l'âge, le sexe et la taille de la tumeur ne sont pas suffisantes. Toutefois, dès le départ, nous avons observé une certaine instabilité dans nos résultats. Si cette instabilité peut en partie être expliquée par la séparation des données (section 6.4), ça ne peut pas être la seule cause. Si l'on compare les résultats des modèles à tâche unique sur l'ensemble de retenue qui est fixe (tableau 5.4) et ceux sur les ensembles de test aléatoires

(voir le tableau 6.3), nous remarquons que les erreurs type moyennes de la justesse balancée sont similaires avec 4.03% pour l'ensemble de retenue fixe et 3.69% pour les ensembles de test aléatoires. Dans le cas de l'ensemble de retenu, les hypothèses qui puissent le mieux expliquer une erreur type aussi élevée est la séparation aléatoire entrainement/validation et l'initialisation des poids de nos modèles. Ne voulant pas fixer notre chiffre d'amorce pour la génération aléatoire des poids, nous avons plutôt décider d'augmenter le nombre d'entraînements nécessaire pour tester nos différentes configurations.

Pour ce qui est de l'application de l'apprentissage multitâche à la classification de tumeurs rénales, nous ne pouvons dire si cette méthode est bénéfique ou non. Alors que nos premiers résultats nous montraient qu'il était possible d'obtenir des gains importants par rapport aux modèles à tâches uniques (tableau 5.15), en changeant la séparation des données nous avons obtenions des résultats qui était totalement différents (tableau 6.8). Ainsi, nous en concluons que l'hétérogénéité des données n'empêche dans le cas présent de tirer des conclusions sur ce point.

De plus, parmi toutes les architectures multitâches que nous avons testées, celle qui nous semble la plus prometteuse est le Learning-To-Branch (LTB). Alors que cet algorithme de recherche d'architecture arrive en seconde place pour les expériences menées avec la première séparation de données (tableau 5.15), elle arrive première avec la deuxième séparation (tableau 6.8). Les plus grands défauts de cet algorithme sont le temps d'entrainement très long et sa difficulté d'implantation. Or, en rendant notre code public nous avons réduit cette seconde difficulté en espérant que la communauté scientifique en profite. En dehors de ça, le LTB s'adapte automatiquement à toute sorte de problèmes en apprenant soi-même l'architecture optimale, ce qui peut réduire le temps passé à développer une architecture multitâche. Nous croyons donc que le LTB mériterait davantage d'intérêt de la part de la communauté scientifique.

Pour finir, nos expériences impliquant l'utilisation de tâches auxiliaires synthétiques ne furent pas moins intéressantes que les autres. Nous avons réussi à démontrer qu'il était possible d'utiliser les caractéristiques radiomiques pour créer de nouvelles tâches dont l'apprentissage pourrait permettre au modèle de découvrir plus facilement des caractéristiques de plus bas niveau que nous ne pourrions observer nous-mêmes. La plus grande difficulté fut toutefois de déterminer les tâches auxiliaires les mieux

adaptées pour chaque tâche principale. Nous avons, dans un premier temps, mis énormément d'espoir dans l'algorithme de groupement de tâche par affinité (GTA), mais ce dernier n'a pas été capable d'identifier de tâches auxiliaires qui étaient systématiquement bénéfiques pour une tâche principale (voir section 6.3.1). Les causes de l'échec du GTA sont probablement les mêmes que celles qui causent l'instabilité de nos résultats. Par exemple, s'il est vrai que pour une même tâche et un même modèle certains ensembles de tests profitent davantage de l'apprentissage multitâche que d'autre, il est alors naturel de croire que certains ensembles profiteraient aussi mieux de certaines tâches auxiliaires que d'autres.

La réalisation de nos travaux n'a pas seulement permis de répondre à des questions par rapport à la prédiction de multiples caractéristiques de tumeurs rénales à la fois à l'aide d'un réseau de neurones, elle a aussi ouvert la voie à plusieurs travaux à venir qui, nous l'espérons, permettront de faire avancer l'apprentissage machine appliquée à l'imagerie médicale. Parmi les potentielles études qui suivront celle-ci, nous notons ci-dessous quatre sujets d'importances qui pourraient bonifier nos efforts de recherches.

1. L'utilisation des caractéristiques radiomiques comme tâches auxiliaires dans le cadre de l'apprentissage multitâche.
2. L'utilisation de l'algorithme LTB appliqué à l'imagerie médicale.
3. L'étude de la prédisposition de certaines données à bénéficier de l'apprentissage multitâche pour une même tâche et une même architecture.
4. La classification des tumeurs rénales par l'imagerie médicale et les réseaux de neurones entrainés sur un plus grand nombre de données.

Tel que mentionné, à notre connaissance, nous sommes les premiers à avoir utilisé les caractéristiques radiomiques comme tâches auxiliaires dans un contexte d'apprentissage multitâche. Cette approche novatrice pourrait nous permettre de mieux identifier les caractéristiques qui ont une influence sur la prédiction du modèle en plus d'améliorer les performances de ce dernier. Les prochaines études n'auront pas à se limiter à la classification de tumeurs rénales ou même à de multiple tâche de prédiction, tout problème d'imagerie médicale où il est possible d'extraire des caractéristiques radiomiques de l'objet à étudier, seront pertinent pour ce sujet.

Pour ce qui est de l'apprentissage multitâche en général, bien qu'il serait toujours

bon d'offrir davantage de cas de comparaison pour les architectures que nous avons étudiées, nous croyons que le LTB mériterait davantage d'attention, puisqu'à notre connaissance nous sommes les premiers à avoir appliqué cet algorithme de recherche d'architecture à un problème d'imagerie médicale et qu'il commence à peine à être considéré pour ce type d'application [153]. Nous espérons que la publication de notre code encouragera la communauté à explorer davantage cette solution.

D'un autre côté, nous croyons qu'il serait aussi nécessaire d'étudier davantage la dépendance des résultats de l'apprentissage multitâche aux données. À la fin de la section 6.4 nous avions émis l'hypothèse selon laquelle, les bénéfices que peut apporter l'apprentissage multitâche ne dépendent pas seulement des tâches, de l'architecture choisie ou encore de la fonction de coût, mais aussi des données qui sont utilisées.

Pour conclure, les résultats que nous avons obtenus pour la classification de la malignité, du sous-type et du grade des tumeurs rénales sont bons, mais pas suffisants. Dans le meilleur des cas, nous obtenions avec une même architecture une justesse balancée de 78.86% pour la classification de la malignité, et pour la même mesure les prédictions du sous-type et du grade arrivent à 79.08% et 71.05% respectivement. Ce n'est probablement pas suffisant pour remplacer les méthodes aujourd'hui utilisées pour la prédiction de ces caractéristiques. Un ensemble de données plus grand ou moins hétérogène pourrait nous permettre de pousser nos recherches plus loin et d'obtenir des résultats bien plus satisfaisants.

Bibliographie

[1] N. Atia, A. Benzaoui, S. Jacques, M. Hamiane, K. Kourd, et others., « Particle Swarm Optimization and Two-Way Fixed-Effects Analysis of Variance for Efficient Brain Tumor Segmentation, » *Cancers*, vol. 14, p. 4399, 09 2022.

[2] M. B. Amin, C. L. Corless, A. A. Renshaw, S. K. Tickoo, J. Kubus, et D. S. Schultz, « Papillary (Chromophil) Renal Cell Carcinoma : Histomorphologic Characteristics and Evaluation of Conventional Pathologic Prognostic Parameters in 62 Cases, » *The American Journal of Surgical Pathology*, vol. 21, no. 6, 1997.

[3] V. Andrearczyk, P. Fontaine, V. Oreiller, J. Castelli, M. Jreige *et al.*, « Multi-task Deep Segmentation and Radiomics for Automatic Prognosis in Head and Neck Cancer, » dans *Predictive Intelligence in Medicine*, série Lecture Notes in Computer Science. Cham : Springer International Publishing, 2021, pp. 147–156.

[4] Y. S. Abu-Mostafa, « Learning from hints in neural networks, » *Journal of Complexity*, vol. 6, no. 2, pp. 192–198, 1990.

[5] J. Bergstra et Y. Bengio, « Random Search for Hyper-Parameter Optimization, » *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.

[6] J. S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl *et al.*, « Algorithms for Hyper-Parameter Optimization, » dans *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 2546–2554.

[7] E. Brochu, V. M. Cora, et N. de Freitas, « A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, » *arXiv*, Décembre 2010, non publié.

[8] D. G. Bostwick et J. N. Eble, « Diagnosis And Classification Of Renal Cell Carcinoma, » *Urologic Clinics of North America*, vol. 26, no. 3, pp. 627–635, 1999.

[9] L. Beljaards, M. S. Elmahdy, F. Verbeek, et M. Staring, « A Cross-Stitch Architecture for Joint Registration and Segmentation in Adaptive Radiotherapy, » dans *Proceedings of the Third Conference on Medical Imaging with Deep Learning*. PMLR, Septembre 2020, pp. 62–74.

[10] B. E. Boser, I. M. Guyon, et V. N. Vapnik, « A training algorithm for optimal margin classifiers, » dans *Proceedings of the fifth annual workshop on Computational learning theory*, série COLT '92. New York, NY, USA : Association for Computing Machinery, 1992, pp. 144–152.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[12] J. L. Ba, J. R. Kiros, et G. E. Hinton, « Layer Normalization, » *ArXiv*, 2016, non publié.

[13] A. Barragán-Montero, U. Javaid, G. Valdés, D. Nguyen, P. Desbordes *et al.*, « Artificial intelligence and machine learning for medical imaging : A technology review, » *Physica Medica*, vol. 83, pp. 242–256, Mars 2021.

[14] I. Buvat, « Les limites du SUV, » *Médecine Nucléaire*, vol. 31, no. 4, pp. 165–172, Avril 2007, 9è Conférence Internationale de l'ACOMEN.

[15] R. Caruana, « Multitask Connectionist Learning, » dans *In Proceedings of the 1993 Connectionist Models Summer School*, 1993, pp. 372–379.

[16] R. Caruana, « Multitask Learning, » *Machine Learning*, vol. 28, Juillet 1997.

[17] R. Caruana, « Multitask Learning, » *School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213*, p. 255, 1997.

[18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, et A. Rabinovich, « Gradnorm : Gradient normalization for adaptive loss balancing in deep multitask networks, » dans *International conference on machine learning*. PMLR, 2018, pp. 794–803.

[19] C. Chen, W. Bai, D. Rueckert *et al.*, « Multi-task Learning for Left Atrial Segmentation on GE-MRI, » dans *Statistical Atlases and Computational Models of the Heart. Atrial Segmentation and LV Quantification Challenges*. Springer International Publishing, 2019, pp. 292–301.

[20] E. M. Caoili et M. S. Davenport, « Role of Percutaneous Needle Biopsy for Renal Masses, » *Semin intervent Radiol*, vol. 31, no. 01, pp. 020–026, Février 2014.

[21] R. Caruana et V. R. De Sa, « Promoting Poor Features to Supervisors : Some Inputs Work Better as Outputs, » dans *Proceedings of the 9th International Conference on Neural Information Processing Systems*, série NIPS'96. Cambridge, MA, USA : MIT Press, Décembre 1996, p. 389–395.

[22] R. Cipolla, Y. Gal, et A. Kendall, « Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics, » dans *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA : IEEE, Juin 2018, pp. 7482–7491.

[23] H. Coy, K. Hsieh, W. Wu, M. B. Nagarajan, J. R. Young *et al.*, « Deep learning and radiomics : the utility of Google TensorFlow™ Inception in classifying clear cell renal cell carcinoma and oncocytoma on multiphasic CT, » *Abdominal Radiology (Online)*, vol. 44, no. 6, pp. 2009–2020, Juin 2019.

[24] D. Cox et S. John, « A statistical method for global optimization, » dans *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, 1992, pp. 1241–1246.

[25] R. Caruana, S. Lawrence, et L. Giles, « Overfitting in Neural Nets : Backpropagation, Conjugate Gradient, and Early Stopping, » dans *Proceedings of the 13th International Conference on Neural Information Processing Systems*, série NIPS'00. Cambridge, MA, USA : MIT Press, 2000, p. 381–387.

[26] Y. Cheng, Z. Liu, et Y. Morimoto, « Attention-Based SeriesNet : An Attention-Based Hybrid Neural Network Model for Conditional Time Series Forecasting, » *Information*, vol. 11, p. 305, Juin 2020.

[27] J. C. Cheville, C. M. Lohse, H. Zincke, A. L. Weaver, et M. L. Blute, « Comparisons of Outcome and Prognostic Features Among Histologic Subtypes of Renal Cell Carcinoma, » *The American Journal of Surgical Pathology*, vol. 27, no. 5, 2003.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, et B. Schiele, « The Cityscapes Dataset for Semantic Urban Scene Understanding, » dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA : IEEE Computer Society, jun 2016, pp. 3213–3223.

[29] Q. Chen, Y. Peng, T. Keenan, S. Dharssi, E. Agro´n *et al.*, « A multi-task deep learning model for the classification of Age-related Macular Degeneration, » *AMIA Summits on Translational Science Proceedings*, vol. 2019, pp. 505–514, Mai 2019.

[30] A. Chartsias, G. Papanastasiou, S. Semple, M. Williams, D. Newby *et al.*, « Disentangled representation learning in cardiac image analysis, » *Medical Image Analysis*, vol. 58, p. 101535, Juillet 2019.

[31] G. Collewet, M. Strzelecki, et F. Mariette, « Influence of MRI acquisition protocols and image intensity normalization methods on texture classification, » *Magnetic Resonance Imaging*, vol. 22, no. 1, pp. 81–91, 2004.

[32] F. Cornelis, E. Tricaud, A. S. Lasserre, F. Petitpierre, J. C. Bernhard *et al.*, « Multiparametric magnetic resonance imaging for the differen-

tiation of low and high grade clear cell renal carcinoma, » *European Radiology*, vol. 25, no. 1, pp. 24–31, Janvier 2015.

[33] C. Cortes et V. Vapnik, « Support-vector networks, » *Machine Learning*, vol. 20, no. 3, pp. 273–297, Septembre 1995.

[34] B. Chen, L. Yang, R. Zhang, W. Luo, et W. Li, « Radiomics : an overview in lung cancer management—a narrative review, » *Annals of Translational Medicine*, vol. 8, no. 18, p. 1191, Septembre 2020.

[35] L. Duong, T. Cohn, S. Bird, et P. Cook, « Low Resource Dependency Parsing : Cross-lingual Parameter Sharing in a Neural Network Parser, » dans *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2 : Short Papers)*. Beijing, China : Association for Computational Linguistics, Juillet 2015, pp. 845–850.

[36] B. Delahunt, J. C. Cheville, G. Martignoni, P. A. Humphrey, C. Magi-Galluzzi *et al.*, « The International Society of Urological Pathology (ISUP) Grading System for Renal Cell Carcinoma and Other Prognostic Parameters, » *The American Journal of Surgical Pathology*, vol. 37, no. 10, pp. 1490–1504, Octobre 2013.

[37] J. Dagher, B. Delahunt, N. Rioux-Leclercq, L. Egevad, J. R. Srigley *et al.*, « Clear cell renal cell carcinoma : validation of World Health Organization/International Society of Urological Pathology grading, » *Histopathology*, vol. 71, no. 6, pp. 918–925, 2017.

[38] J. Duchi, E. Hazan, et Y. Singer, « Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, » *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.

[39] V. Dumoulin et F. Visin, « A guide to convolution arithmetic for deep learning, » *arXiv*, Mars 2016, non publié.

[40] M. S. Elmahdy, L. Beljaards, S. Yousefi, H. Sokooti, F. Verbeek *et al.*, « Joint Registration and Segmentation via Multi-Task Learning for

Adaptive Radiotherapy of Prostate Cancer, » *IEEE Access*, vol. 9, pp. 95 551–95 568, 2021, conference Name : IEEE Access.

[41] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, C. Finn *et al.*, « Efficiently Identifying Task Groupings for Multi-Task Learning, » dans *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 27 503–27 516.

[42] J. Friedman, T. Hastie, et R. Tibshirani, « Regularization Paths for Generalized Linear Models via Coordinate Descent, » *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[43] D. Francoeur, « Machines à vecteurs de support - Une introduction, » *CaMUS*, p. 19, 2010.

[44] M. Fenstermaker, S. A. Tomlins, K. Singh, J. Wiens, et T. M. Morgan, « Development and Validation of a Deep-learning Model to Assist With Renal Cell Carcinoma Histopathologic Interpretation, » *Urology*, vol. 144, pp. 152–157, Octobre 2020.

[45] X. Glorot et Y. Bengio, « Understanding the difficulty of training deep feedforward neural networks, » dans *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, Mars 2010, pp. 249–256.

[46] I. Goodfellow, Y. Bengio, et A. Courville, *Deep Learning*. The MIT Press, 2016.

[47] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin *et al.*, « Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks, » *arXiv*, Février 2020, non publié.

[48] V. Gupta, T. Koren, et Y. Singer, « Shampoo : Preconditioned Stochastic Tensor Optimization, » dans *35th International Conference on Machine Learning*, 2018, pp. 2956–2964.

[49] P. Guo, C.-Y. Lee, et D. Ulbricht, « Learning to Branch for Multi-Task Learning, » dans *Proceedings of the 37th International Conference on*

Machine Learning, série ICML'20. JMLR.org, Juillet 2020, pp. 3854–3863.

[50] M. Goncharov, M. Pisov, A. Shevtsov, B. Shirokikh, A. Kurmukov *et al.*, « CT-Based COVID-19 triage : Deep multitask learning improves joint identification and severity quantification, » *Medical Image Analysis*, vol. 71, p. 102054, 2021.

[51] E. Huynh, T. P. Coroller, V. Narayan, V. Agrawal, Y. Hou, J. Romano *et al.*, « CT-based radiomic analysis of stereotactic body radiation therapy patients with lung cancer, » *Radiotherapy and Oncology*, vol. 120, no. 2, pp. 258–266, Juin 2016.

[52] H. Huhdanpaa, D. Hwang, S. Cen, B. Quinn, M. Nayyar *et al.*, « CT prediction of the Fuhrman grade of clear cell renal cell carcinoma (RCC) : towards the development of computer-assisted diagnostic method, » *Abdominal Imaging*, vol. 40, no. 8, pp. 3168–3174, Octobre 2015.

[53] S. Han, S. I. Hwang, et H. J. Lee, « The Classification of Renal Cancer in 3-Phase CT Images Using a Deep Learning Method, » *Journal of Digital Imaging*, vol. 32, no. 4, pp. 638–643, Août 2019.

[54] T. He, J. Hu, Y. Song, J. Guo, et Z. Yi, « Multi-task learning for the segmentation of organs at risk with label dependence, » *Medical Image Analysis*, vol. 61, p. 101666, 2020.

[55] D. Han, J. Kim, et J. Kim, « Deep pyramidal residual networks, » dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[56] F. Hutter, L. Kotthoff, et J. Vanschoren, « Automated Machine Learning, » *Springer*, 2018.

[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, et R. R. Salakhutdinov, « Improving neural networks by preventing co-adaptation of feature detectors, » *ArXiv*, 2012, non publié.

[58] J. Hu, L. Shen, et G. Sun, « Squeeze-and-Excitation Networks, » dans *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Juin 2018, pp. 7132–7141.

[59] Haibo He, Yang Bai, E. A. Garcia, et Shutao Li, « ADASYN : Adaptive synthetic sampling approach for imbalanced learning, » dans *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Juin 2008, pp. 1322–1328.

[60] K. He, X. Zhang, S. Ren, J. Sun *et al.*, « Identity Mappings in Deep Residual Networks, » dans *Computer Vision – ECCV 2016*. Springer International Publishing, Septembre 2016, pp. 630–645.

[61] K. He, X. Zhang, S. Ren, et J. Sun, « Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification, » dans *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile : IEEE, Décembre 2015, pp. 1026–1034.

[62] K. He, X. Zhang, S. Ren, et J. Sun, « Deep Residual Learning for Image Recognition, » dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Décembre 2016, pp. 770–778.

[63] S. Ioffe, C. Szegedy *et al.*, « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift, » dans *Proceedings of the 32nd International Conference on Machine Learning*, série Proceedings of Machine Learning Research, vol. 37. Lille, France : PMLR, Juillet 2015, pp. 448–456.

[64] M. Islam, V. S. Vibashan, V. J. M. Jose, N. Wijethilake, U. Utkarsh, H. Ren *et al.*, « Brain Tumor Segmentation and Survival Prediction Using 3D Attention UNet, » dans *Brainlesion : Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Cham : Springer International Publishing, Mai 2020, pp. 262–272.

[65] J. Jeong, A. Ali, T. Liu, H. Mao, W. J. Curran, et X. Yang, « Radiomics in Cancer Radiotherapy : a Review, » *arXiv*, Novembre 2019, non publié.

[66] E. Jang, S. Gu, et B. Poole, « Categorical Reparameterization with Gumbel-Softmax, » dans *International Conference on Learning Representations*, 2017.

[67] Y. Jo, J. Kim, C. Park, J. Lee, J. Hur *et al.*, « Guideline for Cardiovascular Magnetic Resonance Imaging from the Korean Society of Cardiovascular Imaging—Part 1 : Standardized Protocol, » *Korean Journal of Radiology*, vol. 20, p. 1313, Septembre 2019.

[68] A. Javaloy et I. Valera, « RotoGrad : Gradient Homogenization in Multitask Learning, » dans *International Conference on Learning Representations*, Janvier 2022.

[69] G. Kovacs, M. Akhtar, B. J. Beckwith, P. Bugert, C. S. Cooper *et al.*, « The Heidelberg classification of renal cell tumours, » *The Journal of Pathology*, vol. 183, no. 2, pp. 131–133, Octobre 1997.

[70] D. Kingma et J. Ba, « Adam : A Method for Stochastic Optimization, » *International Conference on Learning Representations*, Décembre 2014.

[71] A. Krizhevsky, I. Sutskever, G. E. Hinton *et al.*, « ImageNet Classification with Deep Convolutional Neural Networks, » dans *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., Janvier 2012.

[72] G. Kunapuli, B. A. Varghese, P. Ganapathy, B. Desai, S. Cen *et al.*, « A Decision-Support Tool for Renal Mass Classification, » *Journal of Digital Imaging*, vol. 31, no. 6, pp. 929–939, Décembre 2018.

[73] S.-P. Lin et J. J. Brown, « MR contrast agents : Physical and pharmacologic basics, » *Journal of Magnetic Resonance Imaging*, vol. 25, no. 5, pp. 884–899, Avril 2007.

[74] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard *et al.*, « Backpropagation Applied to Handwritten Zip Code Recognition, » *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[75] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard *et al.*, « Backpropagation Applied to Handwritten Zip Code Recognition, » *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[76] B. R. Lane, D. Babineau, M. W. Kattan, A. C. Novick, I. S. Gill *et al.*, « A Preoperative Prognostic Nomogram for Solid Enhancing Renal Tumors 7 cm or Less Amenable to Partial Nephrectomy, » *Journal of Urology*, vol. 178, no. 2, pp. 429–434, Août 2007.

[77] Q. Liao, Y. Ding, Z. L. Jiang, X. Wang, C. Zhang, et Q. Zhang, « Multi-task deep convolutional neural network for cancer diagnosis, » *Neurocomputing*, vol. 348, pp. 66–73, Juillet 2019.

[78] X. Liu, L. Faes, A. U. Kale, S. K. Wagner, D. J. Fu *et al.*, « A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging : a systematic review and meta-analysis, » *The Lancet Digital Health*, vol. 1, no. 6, pp. 271–297, Septembre 2019.

[79] H. Lee, H. Hong, J. Kim, et D. C. Jung, « Deep feature classification of angiomyolipoma without visible fat and renal cell carcinoma in abdominal contrast-enhanced CT images with texture image patches and hand-crafted feature concatenation, » *Medical Physics*, vol. 45, no. 4, pp. 1550–1561, Avril 2018.

[80] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim *et al.*, « Deep Learning in Medical Imaging : General Overview, » *Korean Journal of Radiology*, vol. 18, no. 4, pp. 570–584, Mai 2017.

[81] S. Liu, E. Johns, et A. J. Davison, « End-To-End Multi-Task Learning With Attention, » dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Juin 2019, pp. 1871–1880.

[82] L. Liu, H. Jiang, P. He, W. Chen, X. Liu *et al.*, « On the Variance of the Adaptive Learning Rate and Beyond, » dans *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, Avril 2020.

[83] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, et R. Feris, « Fully-Adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification, » dans *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Juillet 2017, pp. 1131–1140.

[84] A. S. Lundervold et A. Lundervold, « An overview of deep learning in medical imaging focusing on MRI, » *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, Mai 2019.

[85] Z. Liu, P. Luo, X. Wang, et X. Tang, « Deep Learning Face Attributes in the Wild, » dans *Proceedings of International Conference on Computer Vision (ICCV)*, Décembre 2015, pp. 3730–3738.

[86] F. Lin, C. Ma, J. Xu, Y. Lei, Q. Li *et al.*, « A CT-based deep learning model for predicting the nuclear grade of clear cell renal cell carcinoma, » *European Journal of Radiology*, vol. 129, p. 109079, 2020.

[87] P. Lambin, E. Rios-Velazquez, R. Leijenaar, S. Carvalho, R. G. P. M. v. Stiphout *et al.*, « Radiomics : Extracting more information from medical images using advanced feature analysis, » *European Journal of Cancer*, vol. 48, no. 4, pp. 441–446, 2012.

[88] C. J. Maddison, A. Mnih, et Y. W. Teh, « The Concrete Distribution : A Continuous Relaxation of Discrete Random Variables, » dans *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[89] Y. Megumi et K. Nishimura, « Chromophobe Cell Renal Carcinoma, » *Urologia Internationalis*, vol. 61, no. 3, pp. 172–174, 1998.

[90] V. F. Muglia et A. Prando, « Renal cell carcinoma : histological classification and correlation with imaging findings, » *Radiologia Brasileira*, vol. 48, pp. 166–174, Juin 2015.

[91] I. Misra, A. Shrivastava, A. Gupta, et M. Hebert, « Cross-Stitch Networks for Multi-task Learning, » dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3994–4003.

[92] V. Nair et G. Hinton, « Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair, » *Proceedings of ICML*, vol. 27, pp. 807–814, Juin 2010.

[93] D. G. Nishimura, *Principles of Magnetic Resonance Imaging*. Stanford Calif : Stanford Univ., 1996.

[94] P. G. Newman et G. S. Rozycki, « The history of ultrasound, » *Surgical Clinics of North America*, vol. 78, no. 2, pp. 179–195, 1998.

[95] A. M. Osowska-Kurczab, T. Markiewicz, M. Dziekiewicz, et M. Lorent, « Textural and deep learning methods in recognition of renal cancer types based on CT images, » dans *2020 International Joint Conference on Neural Networks (IJCNN)*, Juillet 2020, pp. 1–8.

[96] L. Prechelt *et al.*, « Early Stopping — But When ? » dans *Neural Networks : Tricks of the Trade : Second Edition*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, pp. 53–67.

[97] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury *et al.*, « PyTorch : An Imperative Style, High-Performance Deep Learning Library, » dans *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[98] H. Peng, F. Long, et C. Ding, « Feature selection based on mutual information criteria of max-dependency,max-relevance, and min-redundancy, » *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[99] T. Pan, G. Yang, C. Wang, Z. Lu, Z. Zhou *et al.*, « A Multi-Task Convolutional Neural Network for Renal Tumor Segmentation and Classification Using Multi-Phasic CT Images, » dans *2019 IEEE International Conference on Image Processing (ICIP)*, Septembre 2019, pp. 809–813.

[100] S. Ruder, J. Bingel, I. Augenstein, et A. Søgaard, « Latent Multi-Task Architecture Learning, » *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4822–4829, Juillet 2019.

[101] S.-A. Rebuffi, H. Bilen, et A. Vedaldi, « Efficient Parametrization of Multi-domain Deep Neural Networks, » *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8119–8127, 2018.

[102] S. P. Raman, Y. Chen, J. L. Schroeder, P. Huang, et E. K. Fishman, « CT Texture Analysis of Renal Masses : Pilot Study Using Random Forest Classification for Prediction of Pathology, » *Academic Radiology*, vol. 21, no. 12, pp. 1587–1596, 2014.

[103] F. Rosenblatt, « The perceptron : a probabilistic model for information storage and organization in the brain. » *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[104] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão, D. M. Oliveira, P. R. Gomes *et al.*, « Automatic diagnosis of the 12-lead ECG using a deep neural network, » *Nature Communications*, vol. 11, no. 1, p. 1760, Avril 2020.

[105] E. Rezende, G. Ruppert, A. Theophilo, et T. Carvalho, « Exposing Computer Generated Images by Using Deep Convolutional Neural Networks, » *Signal Processing : Image Communication*, vol. 66, pp. 113–126, Août 2018.

[106] D. A. Roberts, S. Yaida, et B. Hanin, « The Principles of Deep Learning Theory, » *Cambridge University Press*, p. 5, Mai 2021.

[107] D. Sannasy, « Development of Lanthanide Metal Complexes as Contrast Agents for Magnetic Resonance Imaging (MRI). » Thèse de doctorat, University of the Witwatersrand, Août 2012.

[108] B. A. Shannon, R. J. Cohen, H. de Bruto, et R. J. Davies, « The Value of Preoperative Needle Core Biopsy for Diagnosing Benign Lesions Among Small, Incidentally Detected Renal Masses, » *Journal of Urology*, vol. 180, no. 4, pp. 1257–1261, 2008.

[109] C. H. Sudre, M. J. Cardoso, et S. Ourselin, « Longitudinal segmentation of age-related white matter hyperintensities, » *Medical Image Analysis*, vol. 38, pp. 50–64, Mai 2017.

[110] M. Scrivener, E. de Jong, J. Van Timmeren, T. Pieters, B. Ghaye, et X. Geets, « Radiomics applied to lung cancer : A review, » *Translational Cancer Research*, vol. 5, pp. 398–409, 2016.

[111] S. Sabour, N. Frosst, et G. E. Hinton, « Dynamic Routing between Capsules, » dans *Proceedings of the 31st International Conference on Neural Information Processing Systems*, série NIPS'17. Red Hook, NY, USA : Curran Associates Inc., 2017, p. 3859–3869.

[112] H. Suresh, J. J. Gong, et J. V. Guttag, « Learning Tasks for Multitask Learning : Heterogenous Patient Populations in the ICU, » dans *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, série KDD '18. New York, NY, USA : Association for Computing Machinery, 2018, pp. 802–810.

[113] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout : A Simple Way to Prevent Neural Networks from Overfitting, » *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[114] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout : A Simple Way to Prevent Neural Networks from Overfitting, » *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[115] S. G. Silverman, G. M. Israel, B. R. Herts, et J. P. Richie, « Management of the Incidental Renal Mass, » *Radiology*, vol. 249, no. 1, pp. 16–31, 2008.

[116] C. Shorten et T. M. Khoshgoftaar, « A survey on Image Data Augmentation for Deep Learning, » *Journal of Big Data*, vol. 6, no. 1, p. 60, Juillet 2019.

[117] A. B. Shinagare, K. M. Krajewski, M. Braschi-Amirfarzan, et N. H. Ramaiya, « Advanced Renal Cell Carcinoma : Role of the Radiologist in the Era of Precision Medicine, » *Radiology*, vol. 284, no. 2, pp. 333–351, 2017.

[118] K. Sansare, V. Khanna, et F. Karjodkar, « Early victims of X-rays : a tribute and current perception, » *Dento maxillo facial radiology*, vol. 40, no. 2, pp. 123–125, Février 2011.

[119] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed *et al.*, « Going deeper with convolutions, » dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[120] K. Sasaguri et N. Takahashi, « CT and MR imaging for solid renal mass characterization, » *European Journal of Radiology*, vol. 99, pp. 40–54, 2018.

[121] S. Santurkar, D. Tsipras, A. Ilyas, et A. Mądry, « How Does Batch Normalization Help Optimization ? » dans *NeurIPS*, série NIPS'18. Red Hook, NY, USA : Curran Associates Inc., 2018, p. 2488–2498.

[122] S. Sun, « Multitask learning for EEG-based biometrics, » dans *2008 19th International Conference on Pattern Recognition*, Décembre 2008, pp. 1–4.

[123] W. H. Sweet, « The Uses of Nuclear Disintegration in the Diagnosis and Treatment of Brain Tumor, » *New England Journal of Medicine*, vol. 245, no. 23, 1951.

[124] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, et S. Savarese, « Which Tasks Should Be Learned Together in Multi-task Learning ? » dans *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Novembre 2020, pp. 9120–9132.

[125] H. Shen, W. Zhang, H. Wang, G. Ding, et J. Xie, « NDDR-LCS : A Multi-Task Learning Method for Classification of Carotid Plaques, » dans *2020 IEEE International Conference on Image Processing (ICIP)*, Octobre 2020, pp. 2461–2465.

[126] N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan *et al.*, « N4ITK : Improved N3 Bias Correction, » *IEEE Transactions on Medical Imaging*, vol. 29, no. 6, pp. 1310–1320, Juin 2010, conference Name : IEEE Transactions on Medical Imaging.

[127] J. Tompson, R. Goroshin, A. Jain, Y. Lecun, et C. Bregler, « Efficient object localization using Convolutional Networks, » dans *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Juin 2015, pp. 648–656.

[128] T. Tanaka, Y. Huang, Y. Marukawa, Y. Tsuboi, Y. Masaoka *et al.*, « Differentiation of Small ($\leq$ 4 cm) Renal Masses on Multiphase Contrast-Enhanced CT by Deep Learning, » *American Journal of Roentgenology*, vol. 214, no. 3, pp. 605–612, 2020.

[129] F. Türk, M. Lüy, et N. Barışçı, « Kidney and Renal Tumor Segmentation Using a Hybrid V-Net-Based Model, » *Mathematics*, vol. 8, no. 10, p. 1772, Octobre 2020.

[130] S. Tabibu, P. K. Vinod, et C. V. Jawahar, « Pan-Renal Cell Carcinoma classification and survival prediction from histopathology images using deep learning, » *Scientific Reports*, vol. 9, no. 1, p. 10509, Juillet 2019.

[131] K.-H. Uhm, S.-W. Jung, M. H. Choi, H.-K. Shin, J.-I. Yoo *et al.*, « Deep learning for end-to-end kidney cancer diagnosis on multi-phase abdominal computed tomography, » *npj Precision Oncology*, vol. 5, no. 1, pp. 1–6, Juin 2021.

[132] D. Ulyanov, A. Vedaldi, et V. S. Lempitsky, « Instance Normalization : The Missing Ingredient for Fast Stylization, » *ArXiv*, 2016, non publié.

[133] S. Vandenhende, S. Georgoulis, B. De Brabandere, et L. Van Gool, « Branched Multi-Task Networks : Deciding What Layers To Share, » dans *The British Machine Vision Conference 2020 (BMVC 2020)*, Août 2020.

[134] F. Valdora, N. Houssami, F. Rossi, M. Calabrese, et A. S. Tagliafico, « Rapid review : radiomics and breast cancer, » *Breast cancer research and treatment*, vol. 169, no. 2, pp. 217–229, Juin 2018.

[135] X. Wang, H. Chen, A.-R. Ran, L. Luo, P. P. Chan *et al.*, « Towards multi-center glaucoma OCT image screening with semi-supervised joint structure and function multi-task learning, » *Medical Image Analysis*, vol. 63, p. 101695, 2020.

[136] Y. Wu et K. He, « Group Normalization, » *arXiv*, 2018, non publié.

[137] X. Wang, L. Jiang, L. Li, M. Xu, X. Deng *et al.*, « Joint Learning of 3D Lesion Segmentation and Classification for Explainable COVID-19 Diagnosis, » *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2463–2476, Septembre 2021, conference Name : IEEE Transactions on Medical Imaging.

[138] X. Wang, L. Ju, X. Zhao, Z. Ge *et al.*, « Retinal Abnormalities Recognition Using Regional Multitask Learning, » dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, série Lecture Notes in Computer Science. Cham : Springer International Publishing, 2019, pp. 30–38.

[139] X. Wang, Q. Li, J. Cai, W. Wang, P. Xu *et al.*, « Predicting the invasiveness of lung adenocarcinomas appearing as ground-glass nodule on CT scan using multi-task learning and deep radiomics, » *Translational lung cancer research*, vol. 9, no. 4, pp. 1397–1406, Août 2020.

[140] P. Wang, V. M. Patel, I. Hacihaliloglu *et al.*, « Simultaneous Segmentation and Classification of Bone Surfaces from Ultrasound Using a Multi-feature Guided CNN, » dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*. Springer International Publishing, 2018, pp. 134–142.

[141] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville *et al.*, « Show, Attend and Tell : Neural Image Caption Generation with Visual Attention, » dans *Proceedings of the 32nd International Conference on Machine Learning*, série Proceedings of Machine Learning Research, vol. 37. Lille, France : PMLR, Juillet 2015, pp. 2048–2057.

[142] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, et K. He, « Aggregated Residual Transformations for Deep Neural Networks, » *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2016.

[143] W. Xia, B. Luo, et X.-p. Liao, « An enhanced optimization approach based on Gaussian process surrogate model for process control in injec-

tion molding, » *The International Journal of Advanced Manufacturing Technology*, vol. 56, no. 9-12, pp. 929–942, Octobre 2011.

[144] X. Xu, C. Lian, S. Wang, T. Zhu, R. C. Chen *et al.*, « Asymmetric multi-task attention network for prostate bed segmentation in computed tomography images, » *Medical Image Analysis*, vol. 72, p. 102116, Août 2021.

[145] Q. Xu, Q. Zhu, H. Liu, L. Chang, S. Duan *et al.*, « Differentiating Benign from Malignant Renal Tumors Using T2- and Diffusion-Weighted Images : A Comparison of Deep Learning and Radiomics Models Versus Assessment from Radiologists, » *Journal of Magnetic Resonance Imaging*, 2021.

[146] I. Xi, Y. Zhao, R. Wang, M. Chang, S. Purkayastha *et al.*, « Deep Learning to Distinguish Benign from Malignant Renal Lesions Based on Routine MR Imaging, » *Clinical Cancer Research*, vol. 26, p. clincanres.0374.2019, Janvier 2020.

[147] M. Yani, S. Irawan, et C. Setianingsih, « Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail, » *Journal of Physics : Conference Series*, vol. 1201, p. 012052, Mai 2019.

[148] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, et C. Finn, « Gradient surgery for multi-task learning, » *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[149] H. Yu, J. Scalera, M. Khalid, A.-S. Touret, N. Bloch *et al.*, « Texture analysis as a radiomic marker for differentiating renal tumors, » *Abdominal Radiology*, vol. 42, no. 10, pp. 2470–2478, Octobre 2017.

[150] Y. Zhou, H. Chen, Y. Li, Q. Liu, X. Xu *et al.*, « Multi-task learning for segmentation and classification of tumors in 3D automated breast ultrasound images, » *Medical Image Analysis*, vol. 70, p. 101918, 2021.

[151] Y. Zhao, M. Chang, R. Wang, I. Xi, K. Chang *et al.*, « Deep Learning Based on MRI for Differentiation of Low- and High-Grade in

Low-Stage Renal Cell Carcinoma, » *Journal of Magnetic Resonance Imaging*, vol. 52, Mars 2020.

[152] M. D. Zeiler, « ADADELTA : An Adaptive Learning Rate Method, » *ArXiv*, 2012, non publié.

[153] V. A. Zimmer, A. Gomez, E. Skelton, R. Wright, G. Wheeler *et al.*, « Placenta segmentation in ultrasound imaging : Addressing sources of uncertainty and limited field-of-view, » p. 102639, 2023.

[154] J. Zhang, T. He, S. Sra, et A. Jadbabaie, « Why Gradient Clipping Accelerates Training : A Theoretical Justification for Adaptivity, » dans *International Conference on Learning Representations*, 2020.

[155] S. Zagoruyko et N. Komodakis, « Wide Residual Networks, » *arXiv*, 2016, non publié.

[156] M. R. Zhang, J. Lucas, G. Hinton, et J. Ba, *Lookahead Optimizer : K Steps Forward, 1 Step Back*. Red Hook, NY, USA : Curran Associates Inc., 2019.

[157] A. Znaor, J. Lortet-Tieulent, M. Laversanne, A. Jemal, et F. Bray, « International Variations and Trends in Renal Cell Carcinoma Incidence and Mortality, » *European Urology*, vol. 67, no. 3, pp. 519–530, 2015.

[158] A. Zwanenburg, S. Leger, M. Vallières, et S. Löck, « Image biomarker standardisation initiative - feature definitions, » *arXiv*, 2016, non publié.

[159] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, et S. Savarese, « Taskonomy : Disentangling Task Transfer Learning, » dans *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, Juillet 2019, pp. 6241–6245.

[160] Y. Zhang et Q. Yang, « A Survey on Multi-Task Learning, » *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.

[161] X. Zhu, J. Yao, et J. Huang, « Deep convolutional neural network for survival analysis with pathological images, » dans *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2016, pp. 544–547.

[162] L. Zhang, Q. Yang, X. Liu, et H. Guan, « Rethinking Hard-Parameter Sharing in Multi-Task Learning, » *arXiv*, 2021, non publié.

[163] L. Zhou, Z. Zhang, Y.-C. Chen, Z.-Y. Zhao, X.-D. Yin, et H.-B. Jiang, « A Deep Learning-Based Radiomics Model for Differentiating Benign and Malignant Renal Tumors, » *Translational Oncology*, vol. 12, no. 2, pp. 292–300, 2019.

Annexe A

Théorie complémentaire sur l'apprentissage machine

Cette annexe rassemble plusieurs sections que nous avons dû retirer du chapitre 2 afin d'alléger ce dernier. Nous avons fait le choix de les garder en annexe puisqu'elles sont complémentaires au chapitre 2 en plus d'être pertinentes pour mieux comprendre certains détails dans l'implantation de nos modèles.

A.1 Régularisation et augmentation de données

Le sur-apprentissage est l'un des plus gros problèmes de l'apprentissage profond. Lorsque le jeu de données n'est pas suffisamment grand et/ou représentatif et qu'il est difficile, voire impossible, de cueillir davantage de données, comme c'est souvent le cas en imagerie médicale, il faut trouver d'autres solutions pour renforcer la généralisation. Pour un réseau de neurones donné, outre la possibilité de réduire son nombre de poids ou sa profondeur, il existe d'autres techniques bien plus astucieuses pour limiter sa capacité. Ces méthodes sont dites de régularisation. À l'inverse, plutôt que de limiter la capacité du modèle, il est aussi possible d'agrandir virtuellement la taille de notre jeu de données, afin de rendre plus difficile pour notre algorithme d'apprentissage de faire du sur-apprentissage. C'est ce que l'on appelle l'augmentation de données.

A.1.1 Régularisation

Comme mentionné plus tôt, les méthodes de régularisation ont pour objectif de limiter la capacité du modèle ou d'introduire un biais afin de renforcer la généralisation. À la différence de l'augmentation de données, la régularisation ne modifie pas les données en entrées, mais seulement le modèle. Cette section ne présentera que quelques-unes des méthodes développées ces dernières années.

Régularisation sur les poids : Pour commencer, la façon la plus simple de réduire la capacité d'un modèle sans lui enlever des poids est de limiter la valeur que peuvent prendre ces derniers. Comment ? En appliquant une pénalité à la fonction de coût basée sur la norme des matrices de poids afin de forcer l'algorithme à faire tendre la valeur de la somme des poids vers 0. En fonction de la norme choisie pour calculer la pénalité, cette méthode peut avoir des effets variés [42].

Par exemple, si la pénalité est calculée à l'aide de la norme $L1$ (Lasso), alors plupart des poids seront très près de zéro alors que d'autres auront une valeur très élevée. Ce qui aura un effet du type sélection de caractéristiques. En d'autres termes, le modèle basera ses décisions sur seulement quelques caractéristiques/structures identifiées dans les données en entrées. À l'inverse si une pénalité de type $L2$ (Ridge) est choisie, la distribution des poids suivra davantage celle d'une loi normale (gaussienne). Cela aura pour effet de forcer le modèle à identifier davantage de structure pour faire sa prédiction. De plus, un poids similaire sera accordé aux caractéristiques qui sont corrélées. Les normes L1 et L2 sont données par les équations

$$\|^i\theta\|_{L1} = \sum_{j,k} {}^i|\theta_{j,k}| \qquad\qquad \|^i\theta\|_{L2} = \left(\sum_{j,k} {}^i\theta_{j,k}\right)^{1/2}.$$

Afin de contrôler la force de cette régularisation, un hyperparamètre λ viendra multiplier la somme des normes des vecteurs et des matrices de poids. L'équation

$$\mathbb{E}_{D_{\text{train}}}\left[\mathcal{L}\left(f(\vec{x},\theta),\vec{y}\right)\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_{ij}\log(f(\vec{x}_i,\theta)) + \lambda \sum_{{}^i\theta\in\theta} \|^i\theta\|_{L2}$$

représente une fonction de coût de type entropie croisée avec une pénalité $L2$ appliquée aux poids.

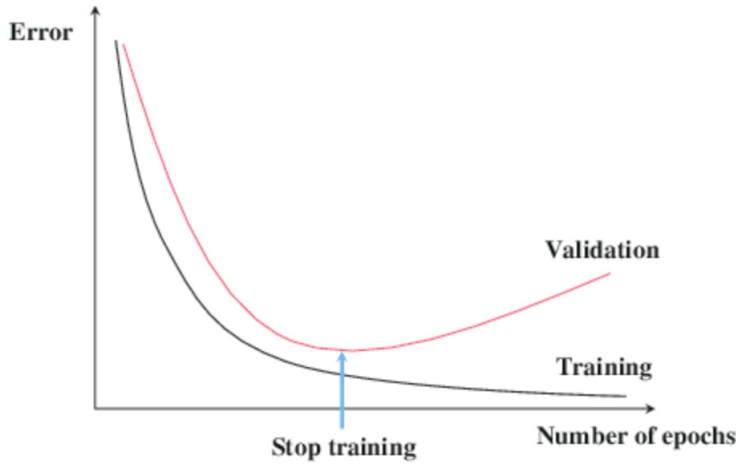


Figure A.1 – Illustration du problème de sur-apprentissage et du nombre de cycles d'entrainement optimale. Image tirée de [1].

arrêt prématuré : Le sur-apprentissage se caractérise par une diminution soudaine des performances sur l'ensemble de validation et/ou de test après plusieurs cycles d'entrainement, alors que l'erreur d'entrainement continue de décroître (voir figure A.1). La première solution que l'on pourrait être amené à tester pourrait être de simplement réduire la durée de l'entrainement. Or, déterminer le nombre optimal de cycles d'entrainement est évidemment laborieux. Il est beaucoup plus simple de se fier à l'erreur de validation mesurée durant l'entrainement afin de stopper l'entrainement ou sauvegarder les poids du modèle à l'itération qui est la plus susceptible de maximiser les performances sur l'ensemble de test. C'est ce que l'on appelle l'arrêt prématuré [25].

Toutefois, l'arrêt prématuré n'est pas sans risque. Stopper l'entrainement en se basant sur les performances sur l'ensemble de validation peut induire un effet de sur-apprentissage sur ce dernier. Sinon, outre le fait que cette technique peut améliorer la généralisation du modèle, elle peut aussi accélérer l'entrainement en l'arrêtant lorsque le modèle commence à converger trop lentement. Ainsi, il n'existe pas de critère d'arrêt unique pour appliquer la technique de l'arrêt prématuré [96]. Certains sont meilleurs pour la généralisation du modèle et d'autres pour accélérer l'entrai-

1. Chen, B., Juillet 2020, Early Stopping in Practice : an example with Keras and TensorFlow 2.0 dans Towards Data Sciences, Récupéré de https ://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd, consulté le 17 mars 2023

nement. De plus, pour les problèmes de classification déséquilibrés (lorsque certaines classes sont sous-représentées dans le modèle), il peut être préférable de mesurer les performances sur l'ensemble de validation à l'aide de la justesse balancée plutôt que l'entropie croisée. Le choix de la métrique à utiliser pour le critère d'arrêt et/ou de sauvegarde est un hyperparamètre supplémentaire qui doit être déterminé avant de lancer l'entraînement.

Extinction des neurones : L'extinction des neurones (*dropout*) [57, 114] est sûrement l'une des techniques de régularisation les plus populaires du siècle présent. Il consiste à retirer temporairement et aléatoirement des neurones du réseau avec probabilité p. À l'itération suivante, les neurones précédemment retirés sont réintégrés et une nouvelle vague de neurones est sélectionnée pour être retirée.Afin que les prédictions restent fixent durant la phase de test/inférence, aucun neurone ne sera retiré, mais leur sortie sera plutôt multiplié par la valeur $1-p$. L'intuition qui motive l'extinction des neurones est qu'en retirant aléatoirement des neurones du réseau le modèle est obligé de baser sa prédiction sur un sous-ensemble aléatoire de caractéristique. Ce faisant, tout comme avec la régularisation $L2$, il sera forcé de considérer davantage de caractéristiques/structures identifiées et d'accorder un poids moindre, mais mieux répartie à chaque neurone.

Malgré le succès de cette technique dès ses débuts, les recherches semblent indiquer que l'extinction des neurones n'est pas tout à fait adapté aux couches de convolution [127]. En effet, les réseaux de neurones à convolution étant conçus pour être entraînés sur des données pour lesquelles il existe une relation spatiale entre les différents éléments qui les composent, comme ces le cas pour les images, les cartes de caractéristiques présentent eux aussi le même type de corrélation spatiale. Ainsi, retirer un seul neurone d'une carte de caractéristiques peut ne pas être suffisant puisque les neurones qui sont à proximité vont compenser sa perte. Pour corriger ce problème, l'extinction spatiale des neurones (*spatial dropout*) [127], une variante qui consiste à faire tomber des cartes de caractéristiques en entier plutôt que de simples neurones, est proposé pour palier ce problème.

A.1.2 Augmentation de données

L'augmentation de données est une technique qui consiste à modifier les données déjà existantes afin d'en créer des nouvelles pour concevoir un jeu de données plus volumineux et/ou plus représentatif. L'hypothèse est qu'en modifiant intelligemment les données, il est possible de créer des cas de figure qui ne sont pas présents dans l'ensemble de données et qui seraient bénéfiques à l'apprentissage de la tâche à effectuer. Il est important de mentionner que dans tous les cas, l'augmentation de données ne devrait être faite que sur les données d'entrainement et non ceux de test afin que les résultats soient représentatifs. Afin de limiter cette section le plus possible, seules les méthodes utilisées dans notre recherche seront abordées. On commencera donc en résumant diverses techniques utilisées sur des images avant de discuter brièvement du Adasyn une méthode spécifique aux données tabulaires que nous avons utilisée avec l'un de nos modèle MVS.

Augmentation de données pour les images : Dans le cas où l'on entraîne un modèle pour effectuer une tâche de vision par ordinateur et que les données sont des images, il existe une panoplie de techniques d'augmentation de données différentes qui peuvent être appliquées, allant d'une simple rotation à la combinaison linéaire de deux images, en passant par le changement de saturation et de contraste [116]. Il existe certaines transformations qui sont spécifiques à l'imagerie médicale, telles que l'ajout d'artefact de Gibbs ou l'ajout d'un champ de polarisation (*bias field*) [109]. Les méthodes d'augmentation de données pour les images sont appliquées durant l'entrainement. En d'autres mots, à chaque itération, chaque image du mini lot va subir un ensemble de transformations qui le plus souvent sont aléatoire, comme le retournement qui va retourner l'image avec une probabilité p donnée. Certaines méthodes possèdent aussi un ou des paramètres qui servent à contrôler l'intensité des modifications apportées, par exemple la rotation aléatoire aura pour paramètre l'angle maximal de rotation. Les différentes méthodes et leurs paramètres d'intensités doivent être soigneusement choisis, car des transformations excessives peuvent rendre l'image irréaliste et nuire à l'entrainement en produisant du sous-apprentissage [46].

Adasyn : Le Adasyn [59] pour *Adaptive Synthetic Sampling Approach for Imbalanced Learning* est un algorithme d'augmentation de données pour les jeux de données tabulaires déséquilibrés. Adapté au problème de classification binaire seulement, le Adasyn a pour objectif de créer des données synthétiques pour la classe minoritaire afin de réduire le débalancement de classe. Pour chaque donnée de la classe minoritaire x_i, on applique l'algorithme des K plus proches voisins pour identifier les exemples minoritaires x_{ij} qui sont les plus près de x_i. Des données synthétiques s_i sont générées en faisant la combinaison linéaire entre chaque x_i et un de leurs plus proches voisins x_{ij} choisi aléatoirement. Cet algorithme comporte deux hyperparamètres, le premier β sert à ajuster le niveau de balancement désiré alors que le second K est utilisé pour l'algorithme des K plus proche voisin.

A.2 Recherche d'hyperparamètres

Malgré l'expertise humaine et la compréhension des réseaux neuronaux acquise au fil des années, l'une des plus grandes difficultés dans le développement des modèles mathématiques reste encore leur entrainement. En effet, la conception du modèle prend parfois jusqu'à plusieurs mois lorsque la tâche à effectuer est assez lourde. Car bien que les paramètres du modèle soient déterminés durant l'entrainement à l'aide d'un algorithme d'optimisation tel que la descente du gradient, le fonctionnement de cet entrainement et la convergence de l'algorithme vers une solution optimale dépendent des paramètres de l'entrainement tel que le taux d'apprentissage, de l'architecture ou encore des paramètres de l'augmentation de données. Ces paramètres qui ne peuvent pas être déterminés par un algorithme de descente standard durant l'entrainement sont dits des hyperparamètres et ils doivent être déterminés avant l'entrainement et par un être humain la plupart du temps.

De ce fait, puisque la sélection des hyperparamètres nécessite des connaissances avancées non seulement en apprentissage machine, mais aussi dans le domaine d'application, cette étape devient rapidement très longue et fastidieuse. Pour corriger cette difficulté, de nombreux algorithmes d'optimisation d'hyperparamètres ont été développés [56, 46] afin d'automatiser cette tâche. Dans cette section, nous commencerons par mettre en place la terminologie et la théorie nécessaire. Ensuite, nous aborderons

les algorithmes de recherche d'hyperparamètres simple dits naïfs avant de s'attaquer à des algorithmes plus complexes basés sur l'optimisation bayésienne.

A.2.1 Fonction boîte noire

En apprentissage machine une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite boîte noire si sa forme analytique est inconnue. Une telle fonction peut cependant être évaluée pour obtenir une valeur, estimer le gradient pour un certain point dans l'espace d'origine ou encore pour être optimisé. Ainsi, très souvent un modèle d'apprentissage machine, tel qu'un réseau de neurones, est considéré comme une boite noire car bien qu'il peut approximer une fonction quelconque s'il est correctement entrainé, il ne nous permet d'acquérir aucune connaissance ou presque sur la fonction à estimer. Par exemple, il est possible d'entrainer un réseau de neurones profond pour détecter et classifier les anomalies chez un patient à partir d'un électrocardiogramme [104]. Toutefois, il est encore très difficile de comprendre les décisions du modèle à partir d'une analyse de ce dernier.

A.2.2 Recherche d'hyperparamètres

Soit A un algorithme d'apprentissage (modèle) possédant N hyperparamètres et soit $\Lambda = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_N$ l'espace des hyperparamètres où Λ_n est le domaine du n-ième hyperparamètre. Posons $\lambda \in \Lambda$ un vecteur d'hyperparamètres. Alors, nous dirons que A_λ représente l'algorithme d'apprentissage A auquel on a attribué les hyperparamètres λ. Alors, l'objectif de la recherche d'hyperparamètres est de calculer le vecteur [56]

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}\left(A_\lambda, D_{\text{train}}, D_{\text{valid}}\right), \tag{A.1}$$

où $\mathcal{L}$ représente la fonction de coût à minimiser, c'est-à-dire la fonction qui mesure le taux d'erreur de notre modèle A_λ sur l'ensemble de données D_{valid} lorsqu'il est entrainé sur D_{train}. La fonction $\mathcal{L}$ est une fonction de type boîte noire et il est donc impossible de résoudre analytiquement le problème (A.1).

A.2.3 Méthode d'optimisation sans modélisation

Afin d'obtenir un vecteur λ qui s'approche de la configuration λ^* qui minimise la fonction de coût ci-dessus, des méthodes naïves et simples, qui ne nécessitent pas de modéliser la fonction boîte noire $\mathcal{L}$, ont été proposées. La recherche par grille est l'une des plus anciennes et des plus simples d'entre elles. Elle consiste à réduire le domaine de chaque hyperparamètre à un espace discret limité (quelques valeurs par hyperparamètres), de faire le produit cartésien de tous ces domaines afin de produire un sous-espace d'hyperparamètres à explorer [56]. Exemple, si nos hyperparamètres sont le taux d'apprentissage ε et le taux de régularisation α ayant chacun pour domaine $\mathbb{R}^+$, alors nous pourrions réduire leur domaine respectif à [0.1, 0.01, 0.001] et [0.05, 0.005]. Ainsi le sous-espace d'hyperparamètres deviendrait $[(0.1, 0.05), (0.1, 0.005), (0.01, 0.05), (0.01, 0.005), (0.001, 0.05), (0.001, 0.005)]$.

Une méthode encore plus simple et efficace est la recherche aléatoire [5]. Il suffit de donner une distribution de probabilité pour chaque domaine d'hyperparamètre et d'échantillonner ces distributions afin de créer un vecteur d'hyperparamètres λ à tester et de choisir celui qui minimise le plus la fonction $\mathcal{L}$. Comparativement à la recherche par grille, cette méthode ne souffre pas du problème de fléau de la dimension, car le nombre d'itérations est toujours indépendant du nombre d'hyperpamètres [56]. De plus, pour des raisons évidentes cet algorithme est facilement parallélisable.

A.2.4 Optimisation bayésienne

Les méthodes d'optimisation bayésienne sont des algorithmes séquentiels qui sont souvent utilisés lorsque l'évaluation de la fonction à minimiser est couteuse. En effet, ces méthodes permettent d'estimer la fonction de coût $\mathcal{L}$ à l'aide d'un modèle de substitution qui est beaucoup plus rapide à évaluer, ensuite ce modèle est utilisé pour déterminer le point le plus pertinent à évaluer avec la fonction $\mathcal{L}$ pour ensuite mettre à jour le modèle.

De manière plus simple, les algorithmes d'optimisation bayésienne sont munis de deux éléments importants. Un modèle de substitution probabiliste $\mathcal{M}$ pour estimer la fonction $\mathcal{L}\left(A_\lambda, D_{\text{train}}, D_{\text{valid}}\right)$ à partir d'un échantillon $\mathcal{S}$ et une fonction d'acquisition que nous dénoterons $q(\lambda, \mathcal{M})$ pour déterminer la prochaine configuration à évaluer

Algorithme 3 : Algorithme d'optimisation bayésienne.

Entrées : $\mathcal{L}$: fonction de coût;
D_{train} : Ensemble d'entrainement;
D_{valid} : Ensemble de validation;
A : Algorithme d'apprentissage machine;
$\mathcal{M}_0$: Modèle de substitution;
q : Fonction d'acquisition;
k : nombre d'itération;

Initialiser $y_{min} \leftarrow \infty$;
Initialiser $\mathcal{S} \leftarrow \emptyset$;
pour $i \leftarrow 0$ *to* $k-1$ **faire**
  $\lambda^* \leftarrow \arg\max_{\lambda \in \Lambda} q(\lambda, \mathcal{M}_i)$;
  $y \leftarrow \mathcal{L}\left(A_{\lambda^*}, D_{train}, D_{valid}\right)$;
  si $y < y_{min}$ *or* $i = 0$ **alors**
    $y_{min} \leftarrow y$;
    $\lambda_{min} \leftarrow \lambda^*$;
  fin
  $\mathcal{S} \leftarrow \mathcal{S} \bigcup (\lambda^*, y)$;
  Estimer $\mathcal{M}_{i+1}$ à partir de $\mathcal{S}$;
fin
Retournerλ_{min};

avec la fonction $\mathcal{L}\left(A_\lambda, D_{\text{train}}, D_{\text{valid}}\right)$. L'algorithme 3 donne la forme générale de la recherche d'hyperparamètres avec les méthodes d'optimisation bayésiens.

Processus gaussien : Les processus gaussiens forment une famille de méthodes d'optimisation bayésienne bien particulière. Un processus gaussien $G\left(m(\lambda), k(\lambda, \lambda')\right)$ peut être pleinement défini par une fonction moyenne $m(\lambda)$ et une fonction de covariance $k(\lambda, \lambda')$. Si nous supposons que les résultats de l'évaluation de la fonction $\mathcal{L}\left(A_\lambda, D_{\text{train}}, D_{\text{valid}}\right)$ ne sont pas bruités, alors nous pouvons définir le processus gaussien en estimant sa moyenne et sa variance à l'aide des équations

$$\mu(\lambda) = k_*^T K^{-1} y,$$

$$\sigma^2(\lambda) = k(\lambda, \lambda) - k_*^T K^{-1} k_*,$$

où k_* est le vecteur de covariance entre λ et toutes les observations précédentes. La fonction de covariance $k(\lambda, \lambda)$ est estimée à l'aide d'un noyau et le choix de ce dernier peut généralement avoir un grand impact sur le modèle de substitution qui en résulte. Le choix qui est le plus populaire est le noyau Matern 5/2 [11].

Pour ce qui est des fonction d'acquisition, le choix est plus souvent laissé à l'utilisateur. Les trois principales fonctions sont l'amélioration espérée (AE) (*expected improvement* :EI) [56, 7], la probabilité maximale d'amélioration (PMA) (*maximum probability of improvement* :MPI) [7, 143] et la borne de confiance inférieure (BCI) (*lower confidence bound* :LCB) [7, 24]. La figure (A.2) montre le comportement des différentes fonctions d'acquisitions.

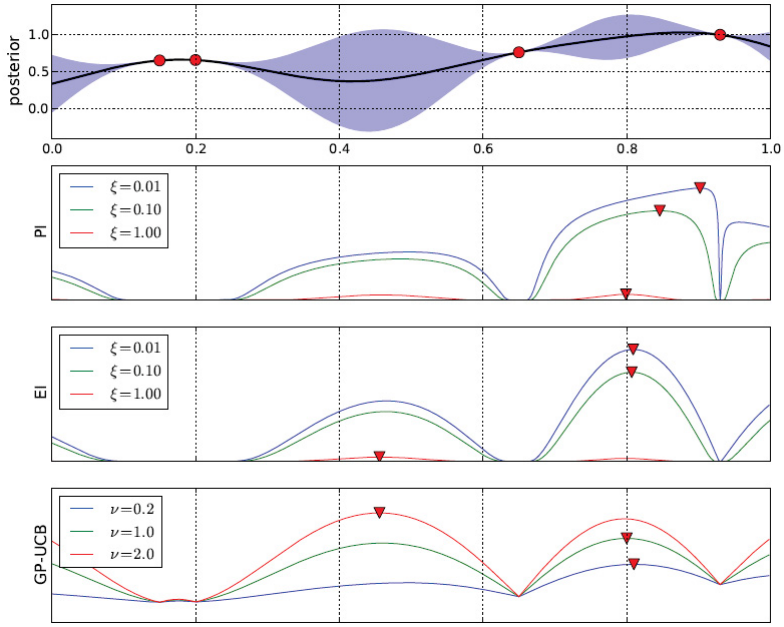


Figure A.2 – Visualisation du modèle de substitution en haut et des trois fonctions d'acquisitions avec différentes configurations ξ Ici, la fonction BCI est remplacée par la borne de confiance supérieure (BCU) car dans le cas ci-dessus nous tentons de maximiser au lieu de minimiser. Image tirée de [7]

L'amélioration espérée, que l'on calcul à l'aide de la formule

$$\mathbb{E}\left[\mathbb{I}(\lambda)\right] = \mathbb{E}_{y\sim\mathcal{M}(\lambda)}\left[\max\left\{y_{min} - y, \right\}\right], \tag{A.2}$$

a pour but de proposer le vecteur λ qui apportera la plus grande amélioration, en maximisant l'amélioration espérée. Cette fonction donnera souvent à l'algorithme un caractère plus exploratoire, c'est-à-dire qu'il aura tendance à explorer une région plus diversifiée de l'espace d'hyperparamètres Λ, de cette manière un plus grand nombre de mauvaises configurations λ seront évalué, mais il aura aussi moins tendance à rester bloquer dans des minimums locaux.

La fonction d'acquisition PMA, donnée par l'équation

$$\mathbb{P}(\mathbb{I}(\lambda)) = \mathbb{E}_{y\sim\mathcal{M}(\lambda)}\left[\mathbb{I}_{[0,y_{min}]}\left(y\right)\right], \tag{A.3}$$

a quant à elle pour but de proposer la configuration qui a la plus grande probabilité de mener à une amélioration, aussi minime soit-elle. Il en résulte que les algorithmes d'optimisation qui utilise cette fonction auront davantage un comportement d'exploitation, c'est-à-dire qu'ils proposeront plus souvent de bonnes configurations, mais auront aussi plus tendance à rester pris dans des minimums locaux.

Pour sa part, le BCI (A.4), dont la fonction est

$$\mathrm{LCB}(\lambda) = \mu(\lambda) - \kappa * \sigma(\lambda),\ \kappa > 0, \tag{A.4}$$

a pour objectif de proposer la configuration λ pour laquelle la borne inférieure de son intervalle de confiance est la plus basse. Le paramètre κ permet justement de contrôler le taux de confiance de cet intervalle. C'est-à-dire que plus la valeur de κ est élevée, alors plus grande sera la probabilité que la valeur $\mathcal{L}_{A,D_{train},D_{valid}}(\lambda)$ tombe dans l'intervalle de confiance. De cette manière, l'algorithme aura souvent plus tendance à explorer des régions qui sont très incertaines. Donc, cette fonction d'acquisition penche davantage vers l'exploration que l'exploitation.

Dans la plupart des cas, le noyau et la fonction d'acquisition utilisés ne permettent pas de trouver une solution analytique et donc l'optimisation de la fonction d'acquisition doit être faite en utilisant une recherche par grille ou un carré latin. Cette

opération possède une complexité algorithmique de l'ordre de $\mathcal{O}(n^3)$, puisque son temps de calcul croît de manière cubique par rapport au nombre d'échantillons et linéaire par rapport au nombre de variables [6]. Malgré tout, cela reste généralement négligeable par rapport à l'évaluation de la fonction de coût $\mathcal{L}$.

Pour finir, le principal avantage des processus gaussiens est qu'ils permettent de découvrir des liens entre les différents hyperparamètres et ainsi converge plus vite lorsque certains hyperparamètres ne sont pas indépendants. Toutefois, ils souffrent aussi de faiblesses importantes, comme le fait qu'ils sont difficilement parallélisable (mais pas impossible) et que leur performance est grandement affectée lorsqu'il y a un grand nombre d'hyperparamètres.

Arbre d'estimateurs de Parzen : L'arbre d'estimateur de Parzen (AEP) est un algorithme d'optimisation bayésien très différent des processus gaussiens. En effet, plutôt que de tenter d'estimer la fonction de densité $\mathbb{P}\left(\mathcal{L}(A_\lambda, D_{train}, D_{valid})|\lambda\right)$, sont modèle essaie plus d'estimer les fonctions de densité suivante : $\mathbb{P}\left(\lambda|\mathcal{L}(A_\lambda, D_{\text{train}}, D_{\text{valid}})\right)$ et $\mathbb{P}\left(\mathcal{L}(A_\lambda, D_{\text{train}}, D_{\text{valid}})\right)$. Pour ce faire, l'algorithme sépare l'échantillon $\mathcal{S}$ en deux partie. Une partie pour les bonnes observations $\mathcal{S}_g$ et une partie pour les mauvaises observations $\mathcal{S}_b$. Ainsi, si l'on pose $\mathcal{L}(A_\lambda, D_{\text{train}}, D_{\text{valid}}) = y$, la fonction de densité $\mathbb{P}\left(\lambda|y\right)$ devient

$$\mathbb{P}\left(\lambda|y\right) = \begin{cases} l(\lambda) & \text{si } y < y^* \\ g(\lambda) & \text{sinon} \end{cases},$$

où y^* représente le seuil, défini par un percentile α, qui sépare les bonnes observations des mauvaises observations (généralement on utilise $\alpha = 0.15$ [56], c'est-à-dire que $\mathbb{P}(y < y^*) = \alpha = 0.15$). Ensuite $l(\lambda)$ représente la fonction de densité estimée à partir des bonnes observations et $g(\lambda)$ est la fonction de densité estimée former par les autres observations. De cette manière, chaque itération du AEP aura une complexité de $\mathcal{O}(n)$, ce qui le rend plus rapide que le processus gaussien. Pour cet algorithme, la fonction d'acquisition est

$$q_{\text{AEP}}(\lambda, \mathcal{M}) = \left(\alpha + \frac{g(\lambda)}{l(\lambda)}(1-\alpha)\right)^{-1}. \tag{A.5}$$

Pour maximiser cette équation, il faut sélectionner la configuration λ qui a une grande probabilité de fournir une bonne observation $l(\lambda)$ et une petite probabilité d'être une mauvaise observation $g(\lambda)$. Il a déjà été démontré que maximiser l'équation (A.5) est équivalent à maximiser la fonction d'acquisition AE [6].

En conclusion, le AEP peut aisément fournir différents points à évaluer en changeant simplement la valeur du percentile α utilisé, ce qui en fait un algorithme facilement parallélisable. La plus grande faiblesse de cet algorithme vient du fait qu'il est incapable de découvrir des relations entre les différents hyperparamètres. De cette manière, si l'algorithme est utilisé sur un ensemble d'hyperparamètres qui ne sont pas indépendants, alors il pourrait mettre plus de temps avant de converger vers une solution optimale[2]. Pour pallier ce problème, les liens de dépendance entre les différents hyperparamètres doivent être définis par l'utilisateur, ce qui est généralement très difficile à faire.

A.3 Perceptron

Développé à la fin des années 50 par Rosenblatt [103], le perceptron pourrait bien être considéré comme le premier algorithme d'apprentissage machine. Cet algorithme a pour but de trouver l'équation d'un hyperplan qui sépare 2 nuages de points dans un espace à d dimensions. Reprenons la notation définit plus haut et posons $D_{\text{train}} = \{(x_i, y_i)\}_{i=i}^{n}$ et supposons que x_i est un vecteur qui représente un point dans un espace à d dimension et y_i une étiquette de classe. Sans perte de généralité, nous poserons $x_i = \vec{x}_i = [1, x_{i,1}, \ldots, x_{i,d}]$. La fonction de décision est présenté par l'équation

$$f(\vec{x}, \theta) = H\left(\theta^\intercal \vec{x}\right) \qquad\qquad H = \begin{cases} 1 & \text{si } a > 0 \\ 0 & \text{sinon} \end{cases},$$

où H représente la fonction Heaviside. Alors que l'équation

$$\mathbb{E}_{D_{\text{train}}}\left[\mathcal{L}\left(f(\vec{x}, \theta), y\right)\right] = \frac{1}{N}\sum_{i=1}^{N} \max\left(0, -y_i \cdot \theta^\intercal \vec{x_i}\right),$$

2. Tiré de Hyper-parameter optimization algorithms : a short review par Aloïs Bissuel publié le 16 août 2019, dans Criteo R&D Blog, récupéré de https://medium.com/criteo-engineering/hyper-parameter-optimization-algorithms-2fe447525903, consulté le 17 mars 2023

représente, quant à elle, la fonction de coût du perceptron. Pour trouver le vecteur de poids $\vec{\theta}$ du modèle, il suffit d'utiliser un algorithme de descente.

La régression logistique est semblable au perceptron à la différence qu'au lieu de simplement classifier x_i, elle donne la probabilité suivante $\mathbb{P}(y_i = 1|x_i)$. Pour ce faire, la fonction de décision est un peu modifier en remplaçant la fonction Heaviside par une fonction sigmoide $\sigma(a) = \frac{1}{1-e^{-a}}$. La fonction de décision et la fonction de coût de la régression logistique sont donnée par les équations ci-dessous respectivement :

$$f(\vec{x}, \theta) = \sigma(\theta^\intercal \vec{x}),$$

$$\mathbb{E}_{D_{\text{train}}}\left[\mathcal{L}\left(f(\vec{x}, \theta), y\right)\right] = \frac{1}{N}\sum_{i=1}^{N} \vec{x_i}\left(\sigma(\theta^\intercal \vec{x_i}) - y_i\right).$$

A.4 Couches de mise en commun et de normalisation par lot

La plupart du temps, les réseaux de neurones à convolutions n'intègre pas seulement des couches de convolution, d'activation et des couches pleinement connecté. Les couches dite de mise en commun et les couches de normalisation font aussi parti intégrante de ces architecture. Cette sous-section présenteras les mécanismes de différentes couches de mise en commun et de la couche de normalisation par lot.

A.4.1 Couches de mise en commun :

Les couches de mise en commun ont le même objectif que les foulées dans les couches de convolutions. Elles doivent rendre le modèle invariant aux légères translations et rotations en réduisant la dimensionnalitée des cartes de caractéristiques. À la différence, des foulées qui sont utilisées durant l'opération de convolution pour passer certaines régions, la couche de mise en commun sera plutôt appliquée après la couche de convolution et elle a pour objectif de résumer le contenue des cartes de caractéristiques sans faire intervenir de poids qui devront être appris par le réseau. La plupart du temps, ce sont des couches de mise en commun maximale et de mise

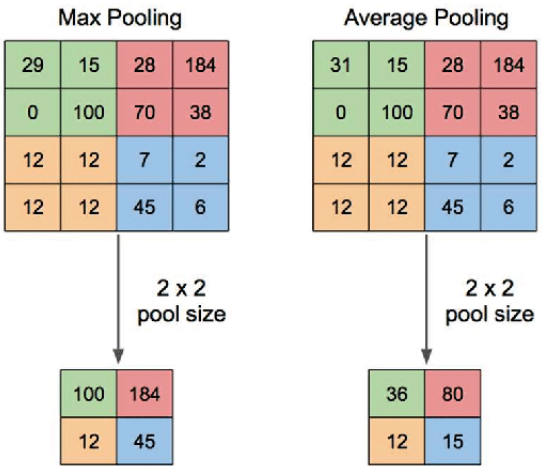


Figure A.3 – Exemple d'opération de mise en commun maximale et de mise en commun moyenne appliquées à une carte de caractéristiques de taille 4×4. Les deux opérations utilise un filtre de taille 2×2 pour séparer la carte en 4 zones distinctes. La mise en commun maximale extrait la plus grande valeur de chaque région, alors que la mise en commun moyenne calcule la moyenne des valeurs pour chacune des régions. Image tirée de [147].

en commun moyenne qui sont utilisées.

Comme le montre la figure A.3, les couches de mise en commun vont séparer la carte d'entrée en plusieurs régions de même taille et vont calculer un élément de chacune de ces régions. Dans le cas de la mise en commun maximale, ce sera la plus grande valeur de chaque valeur qui sera extraite, alors que pour la mise en commun moyenne c'est plutôt la moyenne des valeurs dans chacune des régions qui est calculée. Contrairement au noyau de convolution, le filtre utilisé par les couches de mise en commun sera toujours de même dimension que la carte en entrée.

A.4.2 Normalisation par lot :

Bien qu'apparut relativement récemment dans la littérature (2015 [63]), les couches de normalisation sont maintenant incluses dans la plupart des architectures conçues pour des tâches de vision par ordinateur et le traitement de texte [121]. Bien qu'il existe plusieurs types de couche de normalisation différente [12, 132, 136], c'est principalement la normalisation par lot qui est utilisée en vision par ordinateur. Ci-dessous,

nous commencerons par discuter du fonctionnement de la normalisation par lot avant de résumer les justifications qui expliquent sont succès.

Pour commencer, lors de l'entrainement, la couche de normalisation calculera la moyenne μ_i et l'écart-type σ_i pour chaque canaux i par rapport à tous les éléments du canaux de toutes les cartes d'entrées du lot. Ensuite, chaque canaux sera centré et réduit par rapport aux valeur calculées précédemment. Ensuite, le modèle ce permet d'annuler cet effet à l'aide des paramètres λ_i et β_i qui seront appris durant l'entrainement. De plus, la moyenne et l'écart-type serviront à mettre-à-jour une moyenne mobile $\mu_{\text{mov},i}$ et un écart-type mobile $\sigma_{\text{mov},i}$ qui seront utilisé pendant la phase d'inférence pour remplacé μ_i et σ_i respectivement. Les formules utilisées durant la phase d'entrainement et celle d'inférence sont respectivement donnée par les deux équations suivante :

$$I'_i = \lambda_i \cdot \frac{I_i - \mu_i}{\sigma_i} + \beta_i,$$

$$I'_i = \lambda_i \cdot \frac{I_i - \mu_{\text{mov},i}}{\sigma_{\text{mov},i}} + \beta_i.$$

À la base, les couches de normalisation par lot ont été motivées par la difficulté à choisir le taux d'apprentissage pour les réseaux de neurones très profond. L'hypothèse de départ était que puisque les réseaux de neurones représente en réalité une composition de plusieurs fonctions, changer les poids d'une couche pouvait avoir un impact très important sur la distribution des cartes d'entrée des couches suivantes et si cette distribution est trop modifiée, alors les couches concernées ne seront plus adaptées [46]. Les effets de ce phénomène, surnommé Décalage des covariables internes, pouvaient être minimisés en réduisant le taux d'apprentissage ce qui a aussi pour effet de ralentir l'entrainement du modèle. Les couches de normalisation par lot avaient pour objectif de corriger ce problème, en normalisant la distribution des cartes de caractéristiques afin de contrôler les changement d'une couche à l'autre [63].

Toutefois, en 2018 les effets des couches de normalisation par lot ont été remis en cause [121]. Au final, la fluctuation de la distribution des cartes d'entrées ne serait pas la cause des problèmes d'entrainement mentionnés plus haut puisque le problème proviendrait plutôt de la forme de la surface d'optimisation de la fonction de perte.

L'utilisation de ces couches aurait pour résultat de rendre cette surface plus lisse ce qui a pour effet de rendre « le comportement des gradients plus prévisible et stable, ce qui permet un entrainement plus rapide » [3]. Une surface d'optimisation plus lisse réduit aussi les risques d'explosion ou de disparition du gradient qui seront décrit plus bas. Cette dernière explication semble concorder avec les récents travaux qui montre que les réseaux de neurones plus profond induise des surfaces d'optimisation qui sont hautement non-convexe par rapport à leur variante plus petite.

Pour résumé, les couches de mise en commun et de normalisation par lot sont devenues des incontournable pour les architectures de réseaux de neurones à convolutions. Alors que les premières permettent de réduire la taille des cartes de caractéristiques tout en préservant l'information essentielle et de rendre le modèle un peu invariant aux légères translations et rotations, les couches de normalisation par lot permettent d'accélérer l'entrainement des modèles plus profond en plus d'améliorer leur performances. Avec les couches de convolution et les fonctions d'activation, elles constituent l'essentiel des architectures récemment développer pour les tâches de vision par ordinateur.

3. Traduit et tiré de [121]

Annexe B

Expérimentations préliminaires

Dans cette annexe, nous décrirons les expérimentations préliminaires les plus importantes que nous avons effectuées. Ces expérimentations ont été placées en ordre chronologique, c'est-à-dire celles qui portent sur nos modèles d'apprentissage à tâche unique en premier, suivi du modèle de partage forcé, de celui de partage incité et pour finir de l'algorithme Learning-To-Branch (LTB).

B.1 Architecture à tâche unique

Durant les premiers mois de travail sur ce projet, nous avons rencontré des difficultés avec l'entraînement de nos modèles sur une tâche en particulière, la classification du sous-type. En effet, les résultats obtenus avec nos modèles 3D étaient très sensibles à l'initialisation des poids du réseau de neurones et à la séparation des données entraînement/validation/test. La valeur de la SSC (surface sous la courbe) sur l'ensemble de test pouvait varier entre 0.50 et 0.80. En regardant les courbes d'entraînement, nous avons remarqué que nos modèles avaient souvent de la difficulté à converger durant la première moitié de l'entraînement B.1. Comme le montre le graphique B.2, la classe minoritaire semble être prédominante. Dans un premier temps, nous avons cru que le problème venait de la manière que nous balancions les poids de la fonction de perte. En effet, pour pallier le débalancement de classe, nous effectuons une moyenne pondérée de la fonction de perte, afin qu'au final les classes minoritaires aient autant de poids dans l'optimisation du modèle que les classes majoritaires. Afin de vérifier si

le problème venait de là, nous avons retiré ce balancement, mais comme le montrent les graphiques B.3 et B.4, cela n'a pas réglé le problème.

Au vu de ce problème, nous avons décidé d'effectuer une étape préliminaire à nos expérimentations afin de déterminer quelle architecture était la mieux adaptée pour chacune des tâches.

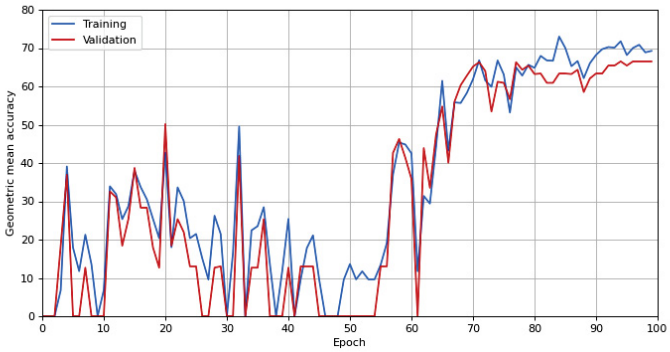


Figure B.1 – Moyenne géométrique du rappel de chaque classe après chaque cycle d'entraînement lors de l'utilisation d'une fonction de coût balancée. Un ResNet de type pré-activation a été utilisé.

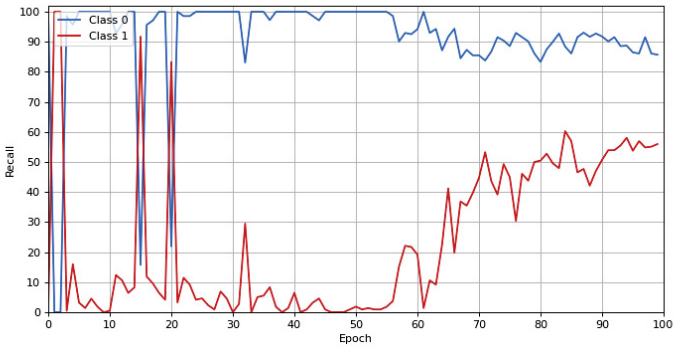


Figure B.2 – Rappel de chaque classe sur l'ensemble d'entraînement après chaque cycle d'entraînement lors de l'utilisation d'une fonction de coût balancée. La classe 0 est la classe minoritaire (74 :227). Un ResNet de type pré-activation a été utilisé.

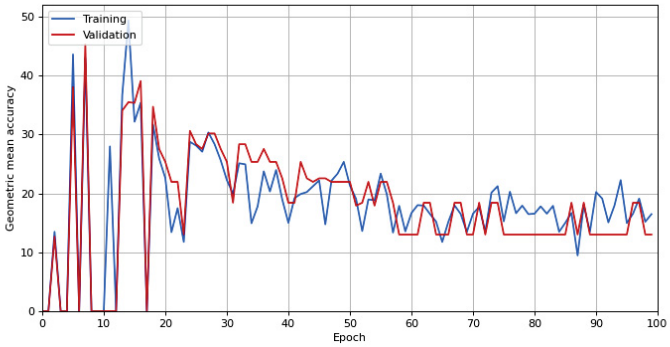


Figure B.3 – Moyenne géométrique du rappel de chaque classe après chaque cycle d'entraînement lors de l'utilisation d'une fonction de coût **non balancée**. Un ResNet de type pré-activation a été utilisé.

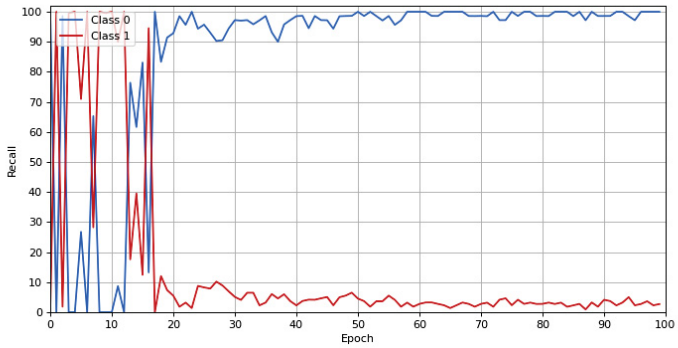


Figure B.4 – Rappel de chaque classe sur l'ensemble d'entraînement après chaque cycle d'entraînement lors de l'utilisation d'une fonction de coût **non balancée**. La classe 0 est la classe minoritaire (74 :227). Un ResNet de type pré-activation a été utilisé.

B.1.1 Bloc résiduel

En tentant de résoudre le problème mentionné plus, il nous est apparu, après plusieurs expérimentations, que le problème pouvait être résolu en changeant l'ordre des différentes couches dans nos blocs résiduels. Après cela, nous avons cru bon de devoir expérimenter deux variantes populaires du ResNet. Soit celle de type pré-activation [60] que nous avons adaptée et une version légèrement modifiée du ResNet standard [62] qui peut être dite de type post-activation. Les détails de ces architectures

se trouvent à la section 4.3.

Pour chaque type de bloc résiduel, pour chaque tâche et pour deux profondeurs de modèle différentes (18 couches et 34 couches) nous avons mesuré la moyenne des résultats de 10 entraînements où l'initialisation des poids et la séparation des données étaient aléatoires à chaque entraînement. Tel que présenté au tableau B.1, le ResNet post-activation offre effectivement, pour la classification du sous-type, de meilleurs résultats que celui de type pré-activation en plus de nous donner des résultats qui sont beaucoup plus stables. Cependant, le modèle pré-activation ResNet donne de bien meilleurs résultats lorsqu'il s'agit des tâches de prédiction de la malignité et du grade.

Suite à ces résultats, nous avons expérimenté différentes architectures hybrides, qui sont composées à la fois de bloc résiduel de type pré-activation (voir 4.7) et de type post-activation (voir 4.6), afin de voir si un compromis était bénéfique. La première de ces deux architectures hybrides utilise des blocs de post-activation pour les deux premiers niveaux et des blocs de type pré-activation pour les deux niveaux suivants. Le second modèle utilise des blocs du type pré-activation pour la première moitié suivie de blocs de type post-activation pour la suite.

Comme le montre les résultats du tableau B.2, les modèles hybrides sont moins avantageux pour la prédiction de la malignité qui semble toujours préférer le ResNet

Tableau B.1 – Comparaison des ResNet de type pré-activation et post-activation

		Pré-act		Post-act	
Tâche	Profondeur	SSC	Justesse balancée	SSC	Justesse balancée
Malignité	18	0.812 ± 0.038	$\mathbf{75.94\% \pm 3.42\%}$	0.764 ± 0.069	$70.83\% \pm 6.91\%$
	34	$\mathbf{0.812 \pm 0.032}$	$75.12\% \pm 3.35\%$	0.769 ± 0.046	$71.44\% \pm 4.30\%$
Sous-type	18	0.740 ± 0.105	$68.64\% \pm 7.91\%$	0.789 ± 0.035	$69.39\% \pm 2.74\%$
	34	0.605 ± 0.122	$58.81\% \pm 10.16\%$	$\mathbf{0.810 \pm 0.048}$	$\mathbf{72.39\% \pm 5.28\%}$
Grade	18	0.708 ± 0.049	$\mathbf{66.69\% \pm 3.33\%}$	0.663 ± 0.051	$62.21\% \pm 5.14\%$
	34	$\mathbf{0.717 \pm 0.050}$	$66.10\% \pm 4.62\%$	0.666 ± 0.050	$62.81\% \pm 5.87\%$

Pour chaque tâche et pour chaque architecture, les résultats présentent la moyenne de 10 entraînements.

Tableau B.2 – Comparaison des modèles hybride à tâche unique

		Post-act x2 - Pré-act x2		Pré-act x2 - Post-act x2	
Tâche	Profondeur	SSC	Justesse balancée	SSC	Justesse balancée
Malignité	18	0.798 ± 0.034	$72.99\% \pm 3.12\%$	0.780 ± 0.032	$72.44\% \pm 3.93\%$
	34	$\mathbf{0.790 \pm 0.049}$	$\mathbf{73,12\% \pm 3.53\%}$	0.770 ± 0.054	$72.20\% \pm 4.27\%$
Sous-type	18	0.779 ± 0.070	$73.89\% \pm 7.50\%$	$\mathbf{0.815 \pm 0.044}$	$\mathbf{74.25\% \pm 4.09\%}$
	34	0.750 ± 0.098	$69.22\% \pm 8.72\%$	0.768 ± 0.080	$71.01\% \pm 6.00\%$
Grade	18	0.702 ± 0.037	$65.57\% \pm 2.95\%$	$\mathbf{0.718 \pm 0.061}$	$\mathbf{66.56\% \pm 4.92\%}$
	34	0.702 ± 0.063	$66.19\% \pm 8.01\%$	0.682 ± 0.053	$62.46\% \pm 4.13\%$

Pour chaque tâche et pour chaque architecture, les résultats présentent la moyenne de 10 entraînements. SSC : Surface sous la courbe.

avec les blocs de type pré-activation. Pour le sous-type, seule notre architecture à 18 couches de profondeur semble profiter du second modèle hybride, alors que le ResNet utilisant des blocs de type post-activation à 34 couches semble plus stable que ses variants hybrides. En vue du développement d'un modèle de type partage forcé dans une section future, nous avons choisi de retenir le modèle hybride (pré-act - post-act) pour cette tâche. Pour la classification du grade, les résultats semblent globalement meilleurs avec le ResNet utilisant des blocs de type pré-activation et c'est donc le modèle que nous retiendrons pour la suite de nos expériences avec cette tâche.

B.1.2 Fusion des masques

Lors de l'analyse du jeu de données, nous avons remarqué pour plusieurs une différence importante entre le masque du scan de pondération T1C et celui de pondération T2. Nous croyons que cette différence peut-être due au fait que la lésion était parfois difficilement visible pour l'une des deux modalités. Nous avons donc supposé dans un premier temps que la fusion des deux masques pourrait aider à identifier la région qui délimite la tumeur. Nous avons donc fusionné la région coupée autour des deux masques. Cette région étant centré sur le centre de masse de chaque lésion, il est possible que certains masques fusionnés n'étaient pas été alignés. Toutefois, comme nous l'avons mentionné plus haut, il n'était pas possible de recaler plusieurs images

Tableau B.3 – Comparaison entre les modèles qui utilisent ou non des masques fusionnés

		Fusionnés		**Non fusionnés**	
Tâche	Profondeur	SSC	Justesse balancée	SSC	Justesse balancée
Malignité	18	0.795 ± 0.031	$72.58\% \pm 3.17\%$	0.812 ± 0.038	$\mathbf{75.94\% \pm 3.42\%}$
	34	0.765 ± 0.057	$70.26\% \pm 3.96\%$	$\mathbf{0.812 \pm 0.032}$	$75.12\% \pm 3.35\%$
Sous-Type	18	0.775 ± 0.067	$71.10\% \pm 6.96\%$	$\mathbf{0.815 \pm 0.044}$	$\mathbf{74.25\% \pm 4.09\%}$
	34	0.783 ± 0.027	$72.04\% \pm 3.08\%$	0.810 ± 0.048	$72.39\% \pm 5.28\%$
Grade	18	$\mathbf{0.717 \pm 0.044}$	$\mathbf{67.71\% \pm 4.60\%}$	0.708 ± 0.049	$66.69\% \pm 3.33\%$
	34	0.703 ± 0.046	$66.89\% \pm 3.38\%$	0.717 ± 0.050	$66.10\% \pm 4.62\%$

Les résultats présentent la moyenne des performances sur 10 entraînements pour chaque tâche et chaque profondeur de réseau. SSC : Surface sous la courbe.

à l'aide de la librairie *Antspy*[1] et il était donc souvent plus efficace de simplement supposer que le centre de masse des deux lésions était centré au même endroit.

Pour cette expérimentation, nous avons utilisé un ResNet de type pré-activation pour les tâches de classification de la malignité et du grade et le modèle hybride pour celle du sous-type. Aucune de ces architectures n'utilise de convolution groupée. Chaque modèle a été entraîné 10 fois pour chaque tâche, pour chaque profondeur (18 et 34 couches) et pour chaque configuration (masques fusionnés ou non). Comme le montre le tableau B.3, il est préférable, lorsqu'il s'agit de classifier la malignité ou le sous-type, de ne pas fusionner les masques. Pour la classification du grade, les modèles utilisant des masques fusionnés semblent plus efficaces, mais pas suffisamment pour compenser la perte de performance avec les autres tâches. En vue d'utiliser l'apprentissage multitâches et afin de garder une solution qui sera uniforme pour toutes les tâches, nous avons choisi de ne pas fusionner les masques pour les prochaines expériences.

B.1.3 Critère de sauvegarde

Afin de contrer les problèmes de surapprentissage, nous avons décidé de sauvegarder notre modèle à l'itération où ses performances sur l'ensemble de validation étaient

1. https ://github.com/ANTsX/ANTsPy

Algorithme 4 : Algorithme d'entrainement avec critère de sauvegarde.

Entrées : M : Modèle;
D_{train} : Ensemble d'entrainement;
X_{Valid} : Données de validation;
Y_{Valid} : Étiquette de validation;
T : Facteur de tolérance;
$\mathcal{L}$: Fonction de coût;
N : Nombre de cycle d'entrainement;

$b_{loss} \leftarrow \infty$; `// Meilleur perte`
$b_{GMR} \leftarrow 0$; `// Meilleur moyenne géométrique des rappels`
$i \leftarrow 0$;
pour $i < N$ **faire**
  $M \leftarrow \text{EntrainerModèle}(M, D_{Train})$;
  $y \leftarrow M(D_{valid})$;
  $l \leftarrow \mathcal{L}(y, t_{valid})$;
  $r \leftarrow \text{CalculerGMR}(y, t_{valid})$;
  si $(l < b_{loss} \ \& \ r >= b_{recall}) || (l < b_{loss} * (1 + T) \ \& \ r > b_{recall})$ **alors**
    $b_{loss} \leftarrow l$;
    $b_{recall} \leftarrow r$;
    SauvegarderPoids(M);
  $i++$;
$M \leftarrow \text{ChargerPoids}()$;

les meilleures et d'utiliser cette sauvegarde pour évaluer les performances du modèle sur l'ensemble de test. Pour définir quand le modèle doit être sauvegardé, nous utilisons le critère de sauvegarde basé sur la fonction de coût et la moyenne géométrique du rappel de chaque classe évaluée sur l'ensemble de validation. Toutefois, nous avons cru nécessaire de vérifier si une mesure aurait été plus appropriée pour définir quand le modèle était le plus susceptible d'atteindre une performance maximale sur l'ensemble de test. Ainsi, nous avons testé 2 critères de sauvegarde supplémentaire. Le premier utilise la surface sous la courbe (SSC) pour remplacer la moyenne géométrique des rappels, tandis que le second utilise le coefficient de corrélation de Matthews (CCM). Dans tous les cas, la fonction de coût est utilisée comme dans l'algorithme 4.

Comme le montrent les résultats des tableaux B.4, l'utilisation de la moyenne géométrique des rappels est de loin la meilleure approche pour définir quand sauvegarder

Tableau B.4 – Comparaisons des critères de sauvegarde utilisés pour l'arrêt prématuré

Mesure	Profondeur	SSC	JB(%)
MGR	18	0.812 ± 0.038	$\mathbf{75.94 \pm 3.42}$
	34	$\mathbf{0.812 \pm 0.032}$	75.12 ± 3.35
SSC	18	0.782 ± 0.038	71.64 ± 3.80
	34	0.792 ± 0.027	72.83 ± 2.64
CCM	18	0.804 ± 0.026	73.19 ± 2.10
	34	0.809 ± 0.023	74.14 ± 2.39

(a) Classification de la malignité.

Mesure	Profondeur	SSC	JB(%)
MGR	18	0.815 ± 0.044	74.25 ± 4.09
	34	0.810 ± 0.048	72.39 ± 5.28
SSC	18	$\mathbf{0.823 \pm 0.041}$	$\mathbf{75.58 \pm 3.30}$
	34	0.769 ± 0.040	69.57 ± 4.24
CCM	18	0.791 ± 0.071	72.14 ± 5.43
	34	0.788 ± 0.043	71.99 ± 5.60

(b) Classification du sous-type.

Mesure	Profondeur	SSC	JB(%)
MGR	18	0.708 ± 0.049	66.69 ± 3.33
	34	0.717 ± 0.050	66.10 ± 4.62
SSC	18	$\mathbf{0.719 \pm 0.047}$	65.30 ± 3.30
	34	0.715 ± 0.040	66.69 ± 5.03
CCM	18	0.710 ± 0.037	$\mathbf{67.75 \pm 3.38}$
	34	0.713 ± 0.075	66.43 ± 6.00

(c) Classification du grade.

Les performances sont calculées à partir de la moyenne des résultats sur 10 entrainements pour chaque architecture et chaque critère de sauvegarde. MGR : Moyenne géométrique des rappels. SSC : Surface sous la courbe CFR. CCM : Coefficient de corrélation de Matthews

le modèle, lorsqu'il s'agit de classifier la malignité. Dans le cas de la classification du sous-type, le critère utilisant le SSC obtient de meilleurs résultats, mais seulement pour le modèle hybride à 18 couches. Pour celui à 34 couches, les résultats sont meilleurs avec le critère de sauvegarde utilisant la moyenne géométrique des rappels. Pour finir, la prédiction du grade montre des résultats qui sont mitigés entre le critère utilisant le SSC et celui utilisant le CCM. Le critère de sauvegarde utilisant le coefficient de corrélation de Matthews ne s'est démarqué que dans un seul test, et ce partiellement, mais il a tout de même obtenu des résultats satisfaisants pour la classification de la malignité. Puisque les résultats sont très serrés, nous avons décidé de poursuivre avec le critère d'arrêt utilisant la moyenne géométrique des rappels pour les prochaines expériences.

B.2 Foncion de coût multitâche

Lors de nos premières expériences, nous avons remarqué que nos modèles ne profitaient pas toujours de l'apprentissage multitâche. Parfois, les modèles à tâche unique obtenaient de meilleurs résultats. Plusieurs hypothèses peuvent expliquer ce phénomène : les deux principales sont le transfert négatif et l'équilibre des fonctions de coûts. Dans cette section, nous explorons deux méthodes pouvant corriger ces problèmes. La première méthode utilise l'incertitude homoscédastique de chaque tâche afin de laisser le modèle apprendre comment balancer les fonctions de coût dynamiquement durant l'entraînement [22] (voir équation 3.1 de la section 3.2). Pour des raisons de simplicité, nous nommerons cette méthode, la fonction de coût d'incertitude. La seconde solution utilise l'algorithme PCGrad [148] afin d'ajuster les gradients des différentes tâches qui entres en conflit. Il est important de noter que cette implémentation du PCGrad utilise aussi l'algorithme GradNorm. Nous avons mené nos expérimentations avec un modèle de type partage forcé inspiré de nos résultats à la section B.1. Les couches partagées par les différentes tâches correspondent à des blocs résiduels de type pré-activation, de même pour les couches spécifiques aux tâches de prédiction de la malignité et du grade, alors que les couches spécifique à la classification du sous-type utilisent des blocs résiduels de type post-activation. Les détails de cette architecture sont disponibles à la section 4.3.

B.2. Foncion de coût multitâche

Comme le montre les résultats des tableaux B.5a et B.5b, la meilleure solution dépend des tâches qui sont entraînées et de l'architecture du modèle. Pour l'architecture à 18 couches de profondeur, les meilleurs résultats pour les tâches de prédiction de la malignité et du Grade sont obtenues à l'aide du PCGrad, alors que la classification du sous-type a obtenu ses meilleurs résultats à l'aide la fonction de perte d'incertitude. Il est à noter que lorsque le modèle est entraîné sur les trois tâches en même temps, c'est la fonction de coût d'incertitude qui obtient les meilleurs résultats sur toutes les tâches. Pour ce qui de l'architecture à 34 couches de profondeurs, la plupart des meilleurs résultats sont obtenus en utilisant une fonction de coût uniforme, c'est-à-dire en n'appliquant aucune des deux solutions. En effet, seule la tâche de prédication de la malignité semble profiter du PCGrad et ce seulement dans 2 scénarios sur 3. Dans un autre ordre d'idée, les résultats montrent aussi que la tâche de prédiction du grade semble profiter davantage d'une architecture à 34 couches de profondeur plutôt que 18, et ce, dans la vaste majorité des scénarios et des solutions utilisées. À l'inverse, les tâches de classification de la malignité et du sous-type ont obtenu leurs meilleurs

Tableau B.5 – Comparaisons des différentes fonctions de coûts multitâches

(a) Architecture à 18 couches de profondeur

	Malignité		**Sous-type**		**Grade**	
Fonction de perte	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
Uniforme	0.791 ± 0.027	73.57 ± 3.02	0.793 ± 0.044	71.34 ± 3.04	0.678 ± 0.072	66.15 ± 5.89
	0.790 ± 0.032	73.39 ± 3.19	0.812 ± 0.037	72.47 ± 5.58	–	–
	0.802 ± 0.033	74.41 ± 2.97	–	–	0.682 ± 0.035	64.54 ± 3.34
	–	–	0.784 ± 0.023	69.97 ± 3.83	0.688 ± 0.068	65.24 ± 5.83
Incertitude	0.803 ± 0.028	73.58 ± 2.00	$\mathbf{0.818 \pm 0.037}$	$\mathbf{75.01 \pm 4.99}$	0.690 ± 0.079	65.70 ± 7.03
	0.802 ± 0.028	73.04 ± 3.01	0.816 ± 0.047	71.46 ± 4.79	–	–
	0.794 ± 0.023	73.61 ± 2.58	–	–	0.678 ± 0.042	65.34 ± 3.52
	–	–	0.781 ± 0.040	69.14 ± 4.16	0.685 ± 0.066	64.72 ± 5.43
PCGrad	0.780 ± 0.021	72.54 ± 2.30	0.772 ± 0.071	69.00 ± 5.76	0.679 ± 0.070	64.79 ± 5.34
	$\mathbf{0.814 \pm 0.019}$	$\mathbf{75.68 \pm 2.92}$	0.787 ± 0.063	69.74 ± 6.88	–	–
	0.804 ± 0.044	74.35 ± 3.52	–	–	$\mathbf{0.714 \pm 0.044}$	$\mathbf{67.50 \pm 3.85}$
	–	–	0.767 ± 0.067	70.23 ± 6.88	0.704 ± 0.044	67.47 ± 3.91

Tableau B.5 – Comparaisons des différentes fonctions de coûts multitâches

(b) Architecture à 34 couches de profondeur

	Malignité		**Sous-type**		**Grade**	
Fonction de perte	SSC	JB (%)	SSC	JB (%)	SSC	JB (%)
Uniforme	0.777 ± 0.023	72.48 ± 1.69	0.794 ± 0.045	68.46 ± 5.51	0.725 ± 0.045	$\mathbf{68.74 \pm 3.96}$
	0.789 ± 0.038	72.66 ± 2.64	0.785 ± 0.044	67.76 ± 5.15	–	–
	0.797 ± 0.035	73.19 ± 4.35	–	–	0.706 ± 0.050	67.85 ± 3.51
	–	–	$\mathbf{0.807 \pm 0.052}$	$\mathbf{72.40 \pm 6.72}$	$\mathbf{0.730 \pm 0.031}$	68.26 ± 2.78
Incertitude	0.778 ± 0.023	72.24 ± 3.37	0.793 ± 0.032	70.85 ± 4.31	0.703 ± 0.068	66.83 ± 4.53
	0.779 ± 0.038	73.24 ± 3.63	0.763 ± 0.044	70.11 ± 5.02	–	–
	0.780 ± 0.033	72.25 ± 3.69	–	–	0.712 ± 0.040	68.57 ± 4.55
	–	–	0.790 ± 0.049	70.99 ± 3.95	0.709 ± 0.053	66.90 ± 4.84
PCGrad	0.786 ± 0.034	72.27 ± 3.65	0.781 ± 0.040	69.37 ± 3.40	0.704 ± 0.054	66.82 ± 4.43
	$\mathbf{0.806 \pm 0.031}$	$\mathbf{73.64 \pm 4.13}$	0.789 ± 0.051	71.31 ± 7.40	–	–
	0.762 ± 0.038	71.60 ± 3.19	–	–	0.703 ± 0.051	68.11 ± 4.17
	–	–	0.763 ± 0.048	69.22 ± 4.52	0.684 ± 0.063	65.12 ± 4.52

Performance sur 10 entrainements pour chaque combinaison de tâches et chaque fonction de coût. SSC : Surface sous la courbe CFR. JB : Justesse balancée.

résultats avec une architecture à 18 couches de profondeurs. Au final, puisque c'est la classification du sous-type qui est le plus affecté par le choix de la fonction de coût multitâche, nous avons choisi d'utiliser la fonction de coût basé sur l'incertitude pour nos prochaines expérimentations. Bien que l'algorithme PCGrade n'obtienne pas de mauvais résultats, il ralentit considérablement l'entraînement de nos modèles ce qui pourrait devenir problématique pour nos futures expérimentations.

B.3 Modèle de type partage incité : pré-entraînement des sous-réseaux

Tel que à la section 4.3, notre modèle multitâche de type partage incité est inspiré de l'architecture point de croix [91] qui utilise des modules de partage afin de permettre à un sous-réseau entraîné pour une tâche A d'utiliser l'information apprise par

B.3. Modèle de type partage incité : pré-entraînement des sous-réseaux

Tableau B.6 – Effet du pré-entrainement des sous-réseaux avant l'entrainement d'un modèle de type partage incité

	Malignité		**Sous-type**		**Grade**	
Pré-entraîné	SSC	JB (%)	SSC	JB(%)	SSC	JB (%)
Non	$\mathbf{0.804 \pm 0.021}$	73.16 ± 2.85	0.819 ± 0.041	73.08 ± 3.20	0.701 ± 0.051	65.16 ± 3.89
	0.802 ± 0.038	$\mathbf{74.31 \pm 3.44}$	0.817 ± 0.034	71.76 ± 3.88	–	–
	0.790 ± 0.031	73.79 ± 3.52	–	–	0.717 ± 0.039	66.24 ± 3.07
	–	–	0.779 ± 0.076	69.14 ± 7.42	$\mathbf{0.719 \pm 0.036}$	$\mathbf{67.83 \pm 3.81}$
Oui	0.788 ± 0.022	73.00 ± 2.49	0.801 ± 0.032	73.19 ± 2.85	0.701 ± 0.067	65.74 ± 4.76
	0.785 ± 0.040	72.70 ± 2.51	0.806 ± 0.028	73.73 ± 4.54	–	–
	0.788 ± 0.024	71.32 ± 3.07	–	–	0.694 ± 0.053	64.07 ± 4.88
	–	–	$\mathbf{0.825 \pm 0.022}$	$\mathbf{74.26 \pm 5.96}$	0.692 ± 0.065	65.61 ± 3.11

Performance sur 10 entraînements pour chaque combinaison de tâches avec et sans pré-entraînement. Des architectures à 18 couches de profondeur sont utilisées. SSC : Surface sous la courbe CFR. JB : Justesse balancée.

un sous-réseau entraîné sur une autre tâche. Dans l'article original de l'architecture point de croix (*cross-stitch*), il est suggéré de pré-entraîné les sous-réseaux sur leurs tâches respectives avant de les connecter avec des modules de partage. Toutefois, agir ainsi ralentit considérablement l'entraînement de nos modèles puisque cela revient à entraîner 4 modèles (3 à tâche unique et 1 multitâche). Nous avons donc voulu vérifier si cette étape était réellement bénéfique. Pour cette expérience, nous avons pré-entraîné les 3 sous-réseaux sur leur tâche respective et à la fin, nous avons utilisé les poids de ses sous-réseaux au meilleur cycle d'entraînement à l'aide de la technique de l'arrêt prématuré. Pour l'entraînement de notre modèle utilisant les sous-réseaux pré-entraînés, nous avons utilisé un taux d'apprentissage 10 fois plus élevé pour les modules de partage.

Comme le montrent les résultats du tableau B.6, à l'exception de la classification du sous-type, les autres tâches ne semblent pas profiter d'un pré-entraînement. Bien au contraire, les résultats sont même moins bons pour la classification de la malignité et du grade. Il nous apparaît donc clair qu'il est préférable de ne pas pré-entraîner les sous-réseaux et de plutôt investir plus de temps dans la recherche d'hyperparamètres.

B.4 Hyperparamètre de température du Learning-To-Branch

L'algorithme Learning-To-Branch possède un hyperparamètre de température τ qui contrôle la capacité du modèle à explorer différent branchement. Une valeur τ de très élevé aura pour conséquence de forcer le modèle à utiliser les différents branchements. Le risque est que ce dernier n'arrive pas à converger vers une solution optimale à temps. C'est pourquoi la valeur de τ doit décroître durant l'entraînement. Nous avons fait quelques expérimentations pour comprendre l'impacte qu'à le choix de la valeur initial de τ par rapport à nos résultats. Afin d'accélérer les expérimentations, nous n'avons entraîné notre modèle que pour la classification de la malignité et du sous-type des tumeurs rénales. Nous avons utilisé une architecture de type ResNet à 18 couches et avec des blocs résiduels de type post-activation.

Tel que le présentent les résultats du tableau B.7, la justesse balancée semble décroître pour la classification de la malignité lorsque la valeur de τ augmente. À l'exception de la valeur $\tau = 20$, le même constat peut être fait pour la classification du sous-type. Au vu de ces résultats, nous avons choisi de fixer la valeur initiale de τ à 10.

Tableau B.7 – Évaluation du paramètre de température pour l'algorithme Learning-To-Branch

	Malignité		**Sout-type**	
τ	SSC	JB (%)	SSC	JB (%)
10	$\mathbf{0.785 \pm 0.026}$	$\mathbf{72.12 \pm 2.38}$	0.804 ± 0.055	72.06 ± 5.27
20	0.745 ± 0.069	67.56 ± 6.53	0.806 ± 0.042	$\mathbf{72.73 \pm 4.63}$
30	0.766 ± 0.034	71.99 ± 3.67	$\mathbf{0.817 \pm 0.037}$	72.02 ± 5.49
40	0.766 ± 0.047	70.15 ± 4.28	0.807 ± 0.044	71.58 ± 5.47
50	0.776 ± 0.042	70.75 ± 4.84	0.780 ± 0.036	67.58 ± 2.71

Performances du LTB pour la classification de la malignité et du sous-type en fonction de l'hyperparamètre de température τ. Une architecture à 18 couches de profondeur et des blocs de type post-activation ont été utilisés. Moyenne des performances sur 10 entraînements. SSC : Surface sous la courbe CFR. JB : Justesse balancée.

Annexe C

Hyperparamètre de base

Dans cette annexe, nous listons tous les hyperparamètres de base que nous avons utilisée durant expérimentations préliminaires (celles qui précédaient nos recherches d'hyperparamètres) à moins d'indication contraire. Ces hyperparamètres de bases ne concerne que les modèles impliquant des réseaux de neurones.

Tableau C.1 – Liste des valeurs de base utilisées pour les hyperparamètres communs à tous nos réseaux de neurones durant la première phase de validation

Hyperparamètre	Valeur par défaut
Taille des lots	32
Taux de d'extinction des neurones	0.3
Nombre de canaux	32
Taux de régularisation L2	3e-6
Taux d'apprentissage	1e-4
ϵ	1e-3
η	1e-6

Nombre de canaux : Nombre de canaux à l'entrée de tous les blocs du **premier niveau** ($N1$). ϵ : Hyperparamètre lié à l'algorithme d'optimisation *Adam*. η : Taux d'apprentissage à la fin de l'entrainement.

Tableau C.2 – Liste des valeurs de base utilisées pour les hyperparamètres spécifiques aux modèles à tâche unique durant la première phase de validation

Hyperparamètre	Valeur par défaut
Profondeur du réseaux	18
configuration des niveaux	4

Tableau C.3 – Liste des valeurs de base utilisées pour les hyperparamètres spécifiques aux modèles de type partage forcé durant la première phase de validation

Hyperparamètre	Valeur par défaut
Coefficient des tâches auxiliaires*	0.5
Configuration	A†, B
Profondeur	18
Niveau de séparation	3

Niveau de séparation : Premier niveau à être dupliqué en plusieurs branches. † : La configuration A et la profondeur mixte ne sont disponibles que pour les expériences qui n'incluent pas de tâches auxiliaires. La configuration B est la valeur par défaut pour les expériences qui incluent des tâches auxiliaires. * : Pour les expériences incluant des tâches auxiliaires seulement.

Tableau C.4 – Liste des valeurs de base utilisées pour les hyperparamètres spécifiques aux modèles de type partage incité durant la première phase de validation

Hyperparamètre	Valeur par défaut
c	0.85
Profondeur	Mixte
Régularisation L2 (Module)	3e-6
Configuration des modules	4
Dispersion	0.1

c : Correspond au paramètre de conservation c des modules de partage. Dispersion : Correspond au paramètre de dispersion d des modules de partage.

Tableau C.5 – Liste des valeurs de base utilisées pour les hyperparamètres spécifiques aux modèles de type RAM durant la première phase de validation

Hyperparamètre	Valeur par défaut
Configuration des niveaux	4
Module d'attention	Spatiaux
Profondeur	18

Tableau C.6 – Liste des valeurs de base utilisées pour les hyperparamètres spécifiques à l'algorithme LTB durant la première phase de validation et les expériences sur l'ensemble de test final

Hyperparamètre	Valeur par défaut
Coefficient des tâches auxiliaires*	0.5
Profondeur	18
Régularisation L2 (branchement)	0
Taux d'apprentissage (branchement)	1e-3
Échauffement	5
η (branchement)	1e-5
τ	10

Échauffement : Nombre de cycles d'entraînements avant de commencer à entrainé les modules de branchement. η (branchement) : Taux d'apprentissage des modules de branchement à la fin de l'entrainement. τ : Hyperparamètre de "température" utilisé par les couches *gumbel-softmax* qui composent les modules de branchement. * : Pour les expériences incluant des tâches auxiliaires seulement.