ESTIMATION D'INCERTITUDE DE SEGMENTATION CARDIAQUE PAR APPRENTISSAGE PROFOND

par

Thierry Judge

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, 17 février 2023

Le 17 février 2023

Le jury a accepté le mémoire de Thierry Judge dans sa version finale

Membres du jury

Professeur Pierre-Marc Jodoin
Directeur
Département d'informatique

Professeur Amine Trabelsi
Membre interne
Département d'informatique

Professeur Maxime Descoteaux
Président-rapporteur
Département d'informatique

Sommaire

L'imagerie cardiaque est l'une des meilleures méthodes pour analyser la fonction cardiaque et diagnostiquer de nombreuses pathologies. Afin de visualiser le coeur, l'imagerie par résonance magnétique, la tomodensitométrie et l'échocardiographie sont, entres autres utilisées. L'analyse de la fonction cardiaque repose en grande partie sur l'extraction de métriques cliniques telles que le volume et le débit cardiaque. Cependant, l'extraction de ces métriques est une tâche longue et fastidieuse qui requiert la segmentation (ou délimitation) des parois cardiaques.

En raison de la lourdeur associée à la tâche de segmentation manuelle et des avancées technologiques récentes, un intérêt accru envers les techniques de segmentation automatique est s'est imposé dans les dernières années. En effet, les progrès de l'intelligence artificielle (IA), propulsés par la disponibilité de grandes bases de données et des nouvelles technologies de calculs, offrent une réelle possibilité de développer des algorithmes de segmentation à la fois automatiques, rapides et fiables. Des algorithmes d'IA proposés dans les dernières années, les réseaux de neurones sont les plus prometteurs. Il a notamment été démontré que les réseaux de neurones profonds atteignent des performances similaires à celle des experts humains sur des bases de données d'IRM et d'ultrason cardiaque.

Cependant, malgré d'excellentes performances en moyenne, les réseaux de neurones sont propices aux erreurs lorsqu'ils sont utilisés sur des données qui diffèrent de celles utilisées en entrainement. Les erreurs peuvent parfois être aberrantes ce qui mine la confiance des cliniciens utilisant ces outils.

Pour cette raison, des techniques d'estimation d'incertitudes sont primordiales pour la mise en service clinique de ces techniques. Plusieurs méthodes sont basées sur l'interprétation probabiliste des réseaux de neurones. Cependant, ces méthodes

n'offrent aucune garantie quant à la détection des régions incertaines des segmentations.

Afin de régler ces problèmes, ce document présentera une méthode basée sur l'apprentissage de formes cardiaques dans un espace latent. Cette méthode ne tient pas compte de la formulation probabiliste des réseaux de neurones, mais se base plutôt sur sur la distribution implicite de formes cardiaques présentent dans les bases de données d'entrainement. Ceci permet d'offrir une certaine garantie de détection d'erreurs si la prédiction s'éloigne trop des données d'entrainement. Cette méthode a démontré des performances supérieures aux méthodes issues de l'état de l'art sur des données ultrasonores cardiaque et de rayons X des poumons.

Mots-clés: estimation d'incertitude, segmentation, apprentissage profond, imagerie cardiaque.

Remerciements

Dans un premier temps, j'aimerais remiercier mon superviseur, le professeur Pierre-Marc Jodoin. Son support et son implication depuis mon arrivée à son laboratoire lors de mon baccalauréat, m'ont permis de réaliser des projets beaucoup plus ambitieux que ce dont j'aurais pu imaginer.

Ensuite, j'aimerais remercier tous ceux qui ont contribué à mon projet. Mon co-superviseur, le professeur Oliver Bernard de l'INSA Lyon, a su me partager son expertise en imagerie cardiaque ce qui a rendu mes projets d'autant plus pertinents. Mes collègues chez Ultromics ont été d'une grande aide lors de mon projet et ont été particulièrement accueillants lors de mon séjour à Oxford.

J'aimerais également remercier les membres de mon laboratoire, le VITALAB. Malgré moins de contacts en raison de la pandémie, le temps passé au laboratoire en leur compagnie ont rendu mes années au laboratoire d'autant plus agréables.

Finalement, j'aimerais remercier ma famille et mes amis pour leur appui avec une mention particulière à ma conjointe Audrey-Anne. Sans son support indéfectible, je n'aurais pu entreprendre un tel projet.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Abréviations

AV Atrio-ventriculaires

FE Fraction d'éjection

i.i.d. indépendant et identiquement distribué

IRM Imagerie par résonance magnétique

KL Kullback–Leibler

VTD Volume télédiastolique

VTS Volume télésystolique

Introduction

Les maladies cardiaques sont la plus grande cause de décès au monde [1]. Souvent, le diagnostic rapide de ces pathologies et une intervention accélérée sont primordiaux afin de réduire les chances de décès. Un des meilleurs outils pour prévenir ces maladies est l'imagerie cardiaque. Plusieurs modalités d'imagerie se retrouvent à la disposition des cliniciens. Chacune de ces modalités comporte des avantages et des limitations. Une des modalités progressant en popularité est l'ultrason. En effet, le nombre d'examens par ultrason est en constante augmentation, car il contient plusieurs avantages par rapport à d'autres modalités [51]. Il s'agit notamment d'un examen peu couteux ne comportant aucun risque pour le patient tout en permettant une acquisition rapide.

Que ce soit avec l'ultrason ou d'autres modalités, les images cardiaques permettent d'extraire plusieurs métriques cliniques telles que le volume et le débit sanguin. Afin d'extraire ces métriques, il est nécessaire de segmenter (processus de délimitation ou classification de régions) le coeur. Cette tâche est complexe, longue et nécessite l'expertise d'un médecin. Cependant, des méthodes d'intelligence artificielle, plus précisément les méthodes d'apprentissage automatique, offrent des solutions automatiques.

Dans les dernières années, il a été démontré que les réseaux de neurones peuvent atteindre des performances comparables à celles d'experts humains pour la segmentation d'imagerie à résonance magnétique et d'ultrason cardiaque [4, 30]. Il faut cependant noter qu'il s'agit de résultats moyens et que les réseaux de n'offrent aucune garantie quant à la qualité de leurs prédictions. Des travaux récents ont développé des méthodes pour identifier, quantifier et corriger des résultats impossibles anatomiquement produits par les réseaux de neurones [40, 37]. Bien que ce type d'erreurs soit identifiable

1. OMS, «Les 10 principales causes de mortalité"». Disponible à `https://www.who.int/fr/news-room/fact-sheets/detail/the-top-10-causes-of-death`

automatiquement, d'autres erreurs peuvent être davantage difficiles à détecter.

En effet, bien que des prédictions peuvent être anatomiquement plausibles, il est très difficile d'identifier si ces résultats correspondent réellement à l'anatomie sous-jacente présente dans l'image. L'estimation d'incertitude est donc un domaine de recherche particulièrement pertinent, non seulement pour la segmentation cardiaque, mais aussi pour le domaine de l'apprentissage profond. Plusieurs méthodes d'estimation d'incertitude ont été proposées dans les dernières années pour diverses tâches réalisées par les réseaux de neurones. Ces méthodes sont cependant, souvent mal adaptées à la segmentation cardiaque.

Dans cette optique, ce mémoire abordera le sujet de l'estimation d'incertitude pour la segmentation cardiaque par réseaux de neurones profonds. Afin d'obtenir une vision d'ensemble du sujet, ce mémoire est divisé comme suit.

Dans un premier temps, l'anatomie et la physiologie cardiaque seront introduites. Le fonctionnement du coeur ainsi que certaines pathologies seront expliqués. Le fonctionnement de l'imagerie par ultrason, avec une emphase sur l'échocardiographie, sera ensuite présenté.

Dans un deuxième temps, les techniques d'apprentissage automatique ainsi que des méthodes de segmentation seront introduites. Les enjeux et les méthodes liés à l'estimation d'incertitude des techniques d'apprentissage automatique seront aussi présentés.

Les deux derniers chapitres de ce mémoire présenteront deux contributions scientifiques. La première a pour but de proposer une meilleure approche pour effectuer l'estimation d'incertitude de la segmentation cardiaque. Cette approche est basée sur l'apprentissage de formes dans un espace latent et permet de mieux identifier les régions erronées dans les prédictions.

La deuxième contribution est une méthode pour améliorer la segmentation d'images échocardiographiques en utilisant une métrique d'évaluation de plausibilité anatomique. Cette méthode permet de réduire le nombre d'erreurs anatomiques présentes dans les segmentations prédites en utilisant des contraintes anatomiques dans l'entrainement.

À la fin de ce mémoire, nous aurons démontré que ces deux contributions permettent à la fois de réduire le nombre d'erreurs produites par les réseaux de neurones, mais aussi à mieux les identifier automatiquement puisque leur élimination complète est, à

ce jour et possiblement indéfiniment, impossible.

Chapitre 1

Anatomie cardiaque

Il est crucial de comprendre le fonctionnement du coeur afin de développer des technologies pour aider au diagnostic des pathologies cardiaques. Celui-ci est un muscle complexe dont la fonction est d'acheminer le sang dépourvu d'oxygène aux poumons et d'ensuite acheminer le sang oxygéné au reste du corps. Le coeur assure cette fonction de manière quasi autonome grâce à une structure nerveuse complexe. En raison de sa fonction vitale, les pathologies liées au coeur sont souvent d'une grande sévérité pour la santé globale humaine.

1.1 Structures cardiaques

Le coeur est un muscle creux, composé de quatre chambres se trouvant dans la cage thoracique. Chaque cavité a une fonction bien précise. Le déplacement du sang est régulé par quatre valves. Les cavités sont entourées de plusieurs couches dont le muscle cardiaque, le myocarde. La pointe inférieure du coeur se nomme l'apex tandis que la partie supérieure, la base. Les différentes structures du coeur sont détaillées ci-dessous et illustrées à la figure 1.1

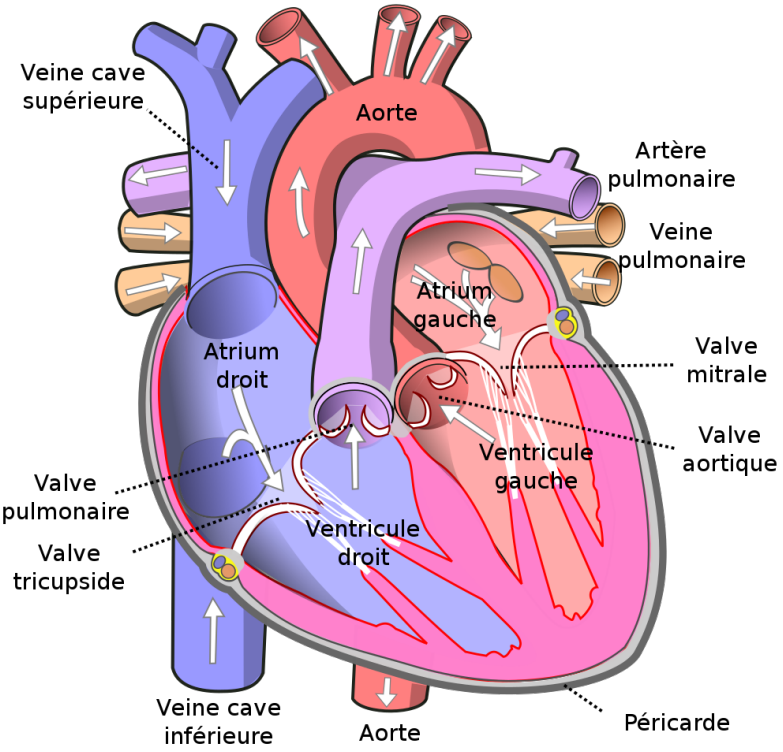


Figure 1.1 – Anatomie cardiaque. Source : Y. Wapcaplet, « Diagram of the human heart » [1]

Cavités : Les quatre chambres sont l'atrium [2] droit, le ventricule droit, l'atrium gauche et le ventricule gauche. L'atrium droit recueille le sang dépourvu d'oxygène à partir des veines caves. Ce sang passe par le ventricule droit qui le pompe vers les poumons. Le sang oxygéné par les poumons revient au coeur dans l'atrium gauche par les veines pulmonaires. Le sang est acheminé vers le ventricule gauche qui pompe le sang vers le reste du corps.

Valves : Afin d'assurer une circulation à sens unique, les quatre chambres sont munies de valves. Les valves tricuspide et mitrale se trouvent entre l'atrium droit et

1. Wikimedia Commons, *GNU Free Documentation License, Version 1.2*. Récupéré de `https://en.wikibooks.org/wiki/File:Diagram_of_the_human_heart.svg`
2. Ancienement appelée oreillette, l'atrium est la nouvelle nomenclature afin d'éviter des confusions avec les dérivés du mot auricule (ex. auriculaire maintenant dit atrial)

le ventricule droit et entre l'atrium gauche et le ventricule gauche respectivement. Ces valves se nomment les valves atrio-ventriculaires (AV). Celles-ci sont reliées aux muscles papillaires par les cordes tendinées. Les muscles papillaires se trouvent sur la paroi interne des ventricules. Ils contrôlent l'ouverture et la fermeture des valves atrio-ventriculaires et empêchent l'ouverture des valves lors de la contraction des ventricules. La valve pulmonaire se trouve entre le ventricule droit et l'artère pulmonaire. La valve aortique se trouve entre le ventricule gauche et l'aorte.

Paroi cardiaque : Les cavités du coeur sont entourées du muscle cardiaque, le myocarde. La partie la plus interne du myocarde se nomme l'endocarde. Le myocarde est un muscle composé de fibres organisées d'une manière complexe permettant la contraction du myocarde ce qui expulse le sang du coeur. Le myocarde est entouré d'une double membrane nommée le péricarde. Cette double membrane entoure la cavité péricardique qui contient le fluide péricardique. Ce fluide assure la lubrification du coeur par rapport aux structures l'entourant. Ces structures sont illustrées à la figure 1.2.

1.2 Système circulatoire

Le coeur est au centre du système circulatoire. Ce système assure le transport du sang qui achemine les nutriments et l'oxygène aux organes du corps. Le sang circule dans cinq types de vaisseaux sanguins. Ces cinq types de vaisseaux sont les artères, les artérioles, les capillaires, les veinules et les veines. Les artères et les artérioles transportent le sang du coeur aux organes du corps tandis que les veinules et les veines expédient le sang vers le coeur. Les capillaires permettent le transfert de gaz (O2 et CO2) et de nutriments entre le sang, les tissus et les organes. Le système circulatoire peut être distingué en deux parties : la circulation pulmonaire et la circulation systémique.

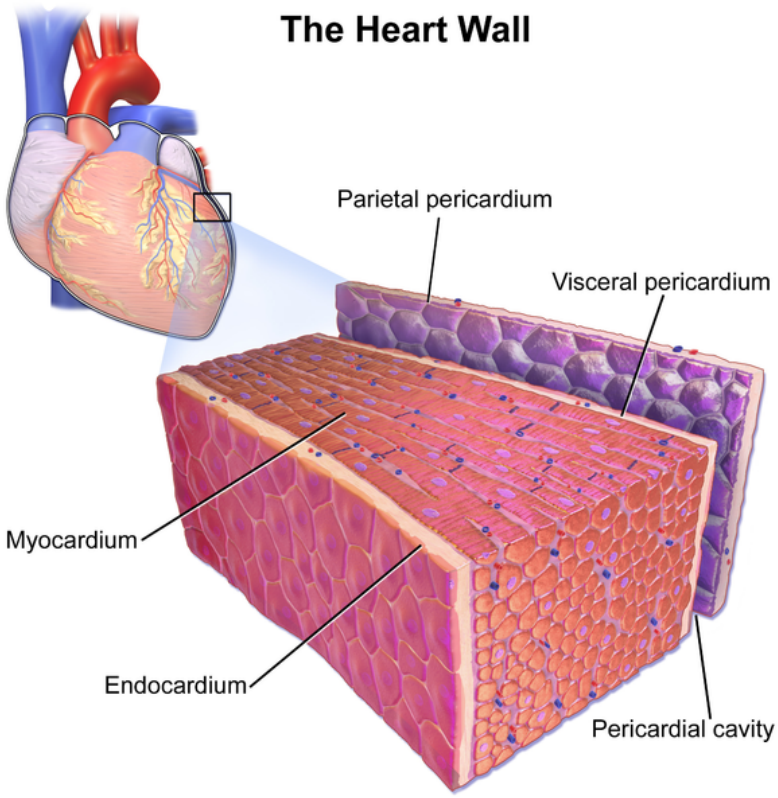


Figure 1.2 – Couches de la paroi cardiaque. Source : B. Blaus, « HeartWall - Medical gallery of Blausen Medical 2014» [3]

Circulation pulmonaire

La circulation pulmonaire a pour but d'acheminer le sang pauvre en oxygène vers les poumons. Ceci se fait à partir du ventricule droit qui expulse le sang vers les poumons par l'entremise de l'artère pulmonaire qui se divise en deux pour atteindre les poumons. Chaque poumon remplace le dioxyde de carbone dans le sang par de l'oxygène. Le sang retourne au coeur par les quatre veines pulmonaires qui se regroupent, et se déverse dans l'atrium gauche.

3. WikiJournal of Medicine, vol. 1, no. 2, 2014. Récupéré de `https://doi.org/10.15347/wjm/2014.010`

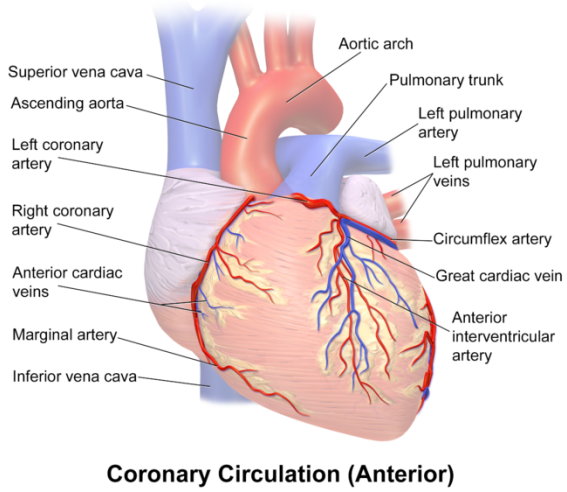


Figure 1.3 – Illustraion des principaux vaisseaux sanguins coronaires. Source : I. Blausen Medical Communications, « Coronary Circulation (Anterior View)» [4]

Circulation systémique

Le sang oxygéné par les poumons arrive dans l'atrium gauche. Il passe ensuite dans le ventricule gauche qui pompe le sang vers le reste du corps par l'intermédiaire de l'aorte. L'aorte se sépare pour fournir le sang au haut et au bas du corps. Les artères se divisent en artérioles. Le sang circule des artérioles aux capillaires pour ensuite passer aux veinules. Finalement, les veinules se regroupent en veines. Celles-ci se regroupent finalement dans les veines caves inférieure et supérieure avant d'atteindre l'atrium droit.

Circulation coronaire : Une portion importante de la circulation systémique à considérer est la circulation coronaire. Celle-ci est particulièrement importante puisque les pathologies liées à la circulation coronaire peuvent souvent être graves. La circulation coronaire commence par deux artères coronaires, les artères coronaires droite et gauche. Celles-ci débutent dans l'aorte et se divisent rapidement en différentes artères. Plusieurs veines ramènent le sang vers l'atrium droit. La majorité du sang

4. Wikimedia Commons, *Creative Commons Attribution 3.0 Unported license.* Récupéré de `https://commons.wikimedia.org/wiki/File:Blausen_0260_CoronaryVessels_Anterior.png`

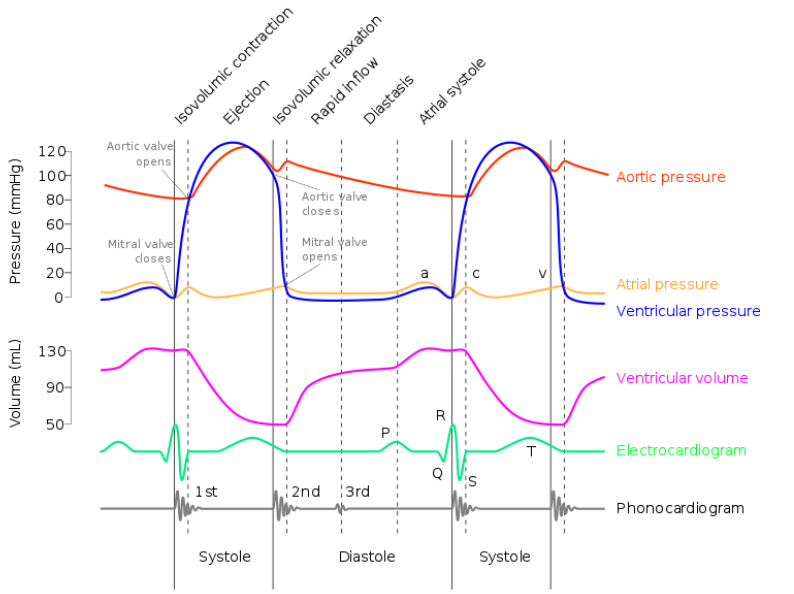


Figure 1.4 – Diagrame de Wigger. Source : Daniel Chang, «Wiggers Diagram 2» [5]

passe par le sinus coronarien tandis que d'autres veines se déversent directement dans l'atrium droit. Les artères et veines coronaires sont illustrées à la figure 1.3.

1.3 Cycle cardiaque

Le cycle cardiaque peut être séparé en deux phases, la systole et la diastole. Comme le coeur est une pompe, il doit se remplir et ensuite se vider. Le coeur se remplit lors de la phase diastolique alors que le muscle cardiaque est relâché. Le coeur pompe le sang vers les poumons et le corps lors de la phase systolique alors que le myocarde se contracte.

Les deux phases du cycle peuvent être divisées davantage. Comme le sang doit se déplacer entre les atriums et ventricules durant le cycle leurs contractions doivent être décalées. De plus, les valves atrio-ventriculaires ainsi que les valves aortique et pulmonaire doivent s'ouvrir et se fermer en conséquence.

La relation entre la pression, le volume ventriculaire ainsi que les évènements du

5. Wikimedia Commons, *Creative Commons Attribution-Share Alike 4.0 International*. Récupéré de `https://commons.wikimedia.org/wiki/File:Wiggers_Diagram_2.svg`

cycle (ouverture/fermeture de valves et contraction) peuvent entre être visualisés avec un diagramme de Wiggers [53]. Un exemple de ce diagramme est illustré à la figure 1.4.

1. **Contraction atriale :** La phase systolique du coeur commence par la contraction des atriums. Lors de cette phase, les valves atrio-ventriculaires sont ouvertes. Avant cette contraction, les ventricules contiennent environ 80% du sang de leur capacité complète. La contraction atriale permet de remplir le dernier 20% [14]. À la fin de la contraction atriale, le volume ventriculaire est à son maximum. On identifie ce volume par le volume télédiastolique (VTD).
2. **Contraction isovolumétrique :** La contraction isovolumétrique se produit quand le myocarde commence sa contracte au niveau des ventricules. Quand la pression des ventricules excède la pression des atriums, les valves atrio-ventriculaires se ferment. La pression ventricule augmente alors rapidement sans changement de volume puisque les valves pulmonaire et aortique sont encore fermées.
3. **Éjection systolique :** L'éjection systolique débute quand la pression ventriculaire excède la pression dans l'aorte et l'artère pulmonaire. La différence de pression force l'ouverture des deux valves et permet l'éjection du sang.
4. **Relaxation isovolumétrique :** Vers la fin de la contraction du myocarde, la pression ventriculaire chute ce qui force la fermeture des valves pulmonaire et aortique. Le volume ventriculaire est à son plus faible lors de cette phase, on appelle ce volume le volume télésystolique (VSD).
5. **Remplissage :** Alors que le myocarde n'est plus en contraction, la pression ventriculaire finit par descendre sous le niveau de pression des atriums. Les valves atrio-ventriculaires s'ouvrent et le remplissage passif des ventricules débute.

À partir des volumes mentionnés plus haut, il est possible de définir des métriques très importants pour l'évaluation de la fonction cardiaque. Premièrement, le volume d'éjection systolique (VES) est le volume de sang éjecté par le ventricule à chaque battement. Il se calcule comme suit

$$\text{VES} = \text{VTD} - \text{VTS}. \tag{1.1}$$

Il est possible de calculer le débit cardiaque Q_c, soit la quantité de sang pompé par le coeur à chaque minute

$$Q_c = \text{VES} \cdot F_c, \tag{1.2}$$

où F_c est la fréquence cardiaque.

Finalement, on peut calculer la fraction d'éjection (FE) qui indique le pourcentage de sang éjecté par battement

$$\text{FE} = \frac{\text{VTD} - \text{VTS}}{\text{VTD}} = \frac{\text{VES}}{\text{VTD}}. \tag{1.3}$$

Ces trois métriques peuvent être calculées pour le ventricule gauche et le ventricule droit. Par contre, il est plus commun de rapporter ces valeurs pour le ventricule gauche comme elles indiquent plus souvent la performance du coeur. À moins d'indications contraires, les valeurs de VTD, VTS, VES, Q_c et FE feront référence au ventricule gauche pour le reste de ce document.

1.3.1 Tissu cardionecteur

Le coeur effectue le cycle cardiaque de manière autonome, grâce à un ensemble de cellules nommées le tissu cardionecteur. Ce tissu est dispersé à travers le myocarde et compose environ 1% du muscle cardiaque. Le tissu cardionecteur, aussi appelé tissu nodal, déclenche la contraction du myocarde dans un ordre très précis et à un certain rythme. Ce tissu est composé de plusieurs noeuds ayant des fonctions bien précises qui sont illustrés à la figure 1.5.

Le rythme du cycle cardiaque est principalement contrôlé par le noeud sinusal (SA). Il se trouve sur la paroi de l'atrium droit et donne une impulsion électrique au reste du coeur à un rythme d'environ 70 à 80 battements par minute, soit le rythme cardiaque moyen chez un humain en santé. Ce rythme est généré de manière autonome, mais peut être changé par le système nerveux qui peut accélérer le rythme cardiaque lors de l'effort physique par exemple.

L'impulsion électrique générée par le noeud SA est transmis par trois faisceaux de tissus nodaux. Lors de chaque impulsion, le signal cause la contraction des atriums. Le

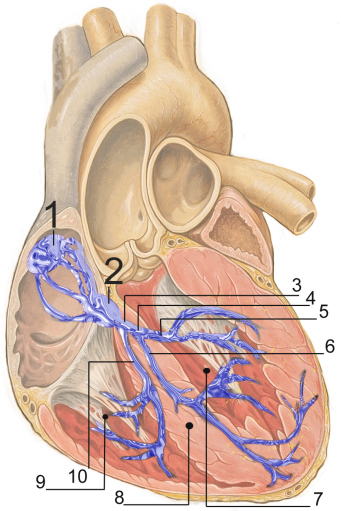


Figure 1.5 – Illustration du tissu cardionecteur. 1. nœud sinusal 2. nœud atrio-ventriculaire 3. faisceau de His 4. branche gauche 5. faisceau postérieur gauche 6. faisceau antérieur gauche 7. ventricule gauche 8. septum interventriculaire 9. ventricule droit 10. branche droite. Source : J. Heuser, « Répartition du tissus nodal dans le coeur » [6]

signal se rend vers le noeud atrioventriculaire. Ce noeud est primordial dans le cycle cardiaque puisqu'il crée un délai dans la transmission de l'impulsion. Ceci permet aux atriums de se contracter avant les ventricules lors de la systole.

Les impulsions poursuivent leur trajet dans le myocarde par le faisceau de His. Le signal se subdivise de plus en plus. Le signal se rend aux fibres de Purkinje qui sont dispersées dans le myocarde. Le myocarde se contracte de l'apex vers la base. Ceci a pour effet de pousser le sang vers le haut, soit vers l'aorte et l'artère pulmonaire.

6. Wikimedia Commons, *Creative Commons Attribution 2.5 License 2007*. Récupéré de `https://commons.wikimedia.org/wiki/File:RLS_12blauLeg.png`

1.4 Pathologies cardiaques

Ayant un rôle si important dans le fonctionnement du corps humain, les pathologies qui atteignent le coeur peuvent souvent avoir des répercussions importantes sur la santé globale. Différentes pathologies sont présentées ci-dessous.

1.4.1 Cardiomyopathie

La cardiomyopathie est une maladie qui affecte la structure du myocarde. Il existe trois types principaux de cardiomyopathie : la cardiomyopathie dilatée, la cardiomyopathie hypertrophique et la cardiomyopathie restrictive. La cardiomyopathie dilatée se produit quand les parois du ventricule s'amincissent. Ce type de cardiomyopathie peut être causé par une infection virale ou une maladie génétique et peut provoquer de l'insuffisance cardiaque. La cardiomyopathie hypertrophique se produit quand le myocarde s'épaissit et se raidit. Ceci est souvent causé par une anomalie génétique. Due à l'épaississement du myocarde, le myocarde ne se décontracte pas correctement ce qui diminue la capacité du coeur à se remplir adéquatement. Finalement, la cardiomyopathie restrictive est similaire à la cardiomyopathie hypertrophique, mais se manifeste seulement par le raidissement des parois ventriculaires ce qui peut aussi provoquer de l'insuffisance cardiaque.

1.4.2 Maladies coronariennes

Les maladies coronariennes se produisent quand il y a une obstruction partielle ou complète d'une ou plusieurs artères coronaires. Celles-ci peuvent être causées par l'accumulation de dépôts graisseux le long des parois des artères, pathologie qui est nommée athérosclérose. Une de ces maladies est l'angor qui se produit quand une partie du coeur ne reçoit pas assez de sang. Ceci provoque souvent des douleurs à la poitrine et peu être causé par une charge de travail trop grande pour les capacités du coeur. Une autre maladie coronarienne, plus sévère, est l'infarctus du myocarde. Ceci se produit quand une artère coronaire est obstruée soudainement. La nécrose (la mort d'un tissu) d'une portion du myocarde peut en découler si l'obstruction est d'une durée prolongée. L'angor et l'infarctus du myocarde .

1.4.3 Maladies des valves cardiaques

Tel que mentionné plus haut, le coeur est muni de quatre valves. Deux d'entre-elles, les valves AV régulent l'entrée de sang dans les ventricules et les deux autres, les valves pulmonaire et aortique, régulent. la sortie du sang. Chacune de ces valves peut être atteinte de pathologies. On distingue deux types de pathologies en ce qui concerne les valves. La régurgitation indique que le sang traverse la valve dans la mauvaise direction, car la valve ne se ferme pas correctement. La sténose indique un rétrécissement de l'ouverture de la valve ce qui empêche le sang d'entrer ou sortir des ventricules.

Chapitre 2

Imagerie ultrasonore

L'imagerie médicale est l'un des piliers de la médecine moderne. Elle permet de voir l'intérieur du corps humain. Il existe plusieurs modalités d'imagerie, chacune ayant ses avantages et ses inconvénients.

Le choix de modalité doit être fait en tenant compte des inconvénients de celle-ci tels que le coût, l'invasivité et l'ionisation par rapport à la qualité de l'image requise en fonction de la sévérité de la pathologie à diagnostiquer. Différentes modalités communes sont présentées au tableau 2.1.

L'imagerie par ultrason comporte plusieurs avantages. Ce type d'imagerie est non-invasif, peu coûteux et facile d'opération. Ceci vient cependant au prix d'une image de faible qualité. L'ultrason est un particulièrement utilisé pour l'imagerie de tissus mous et de vaisseux sanguins.

Afin de développer des outils logiciels adéquats pour l'analyse d'images échographiques, une bonne compréhension du fonctionnement et des limitations sont nécessaires.

2.1 Propagation d'ondes

L'imagerie par ultrason repose sur la physique de la propagation d'ondes. L'ultrason utilise des ondes mécaniques envoyées dans le corps pour identifier les interfaces entre

Tableau 2.1 – Résumé des modalités communément utilisée en clinique

Modalité	Fonctionnement	Utilisation	Avantages et Inconvénients
Rayon-X	Mesure l'absorption de rayons X qui traversent les différentes structures du corps.	◦ Os ◦ Poumons	**Avantages** ◦ Simple ◦ Rapide **Inconvénients** ◦ Radiation ◦ Limité au 2D
CT-scan	Reconstruction 3D de plusieurs acquisitions de rayon X dans différentes directions.	◦ Os ◦ Organes internes ◦ Cancer	**Avantages** ◦ Facile ◦ 3D **Inconvénients** ◦ Radiation ◦ Coûteux ◦ Encombrant ◦ Nécessite un opérateur technicien.
IRM	Mesure du temps de réalignement des molécules d'hydrogène perturbées par une radiofréquence.	◦ Cerveau ◦ Coeur ◦ Foie	**Avantages** ◦ Bonne résolution ◦ 3D **Inconvénients** ◦ Coûteux ◦ Encombrant ◦ Nécessite un opérateur technicien.
Ultrason	Mesure la réflexion causée par les frontières entre structures d'une onde envoyée dans le corps	◦ Coeur ◦ Foetus ◦ Foie	**Avantages** ◦ Peu coûteux ◦ Temps réel **Inconvénients** ◦ Faible résolution

les tissus anatomiques à l'aide des échos. Une onde est une perturbation dans un milieu qui se propage d'un point à un autre. Il existe deux sortes d'ondes mécaniques.

Le type d'onde utilisée par l'ultrason est une onde mécanique nommée l'onde longitudinale. Contrairement à l'onde transversale qui déforme son milieu de transmission perpendiculairement à sa direction de propagation (une vague dans l'eau), l'onde longitudinale déforme le milieu de propagation dans la direction de propagation.

Une onde longitudinale est le résultat d'une variation de pression appliqué sur un milieu de transmission. Cette variation de pression crée des zones de compressions, où les particules sont en plus forte concentration et des zones de raréfaction où les particules sont en faible concentration.

2.1.1 Caractérisques de l'onde

Une onde peut être caractérisée par plusieurs paramètres. D'abord, l'amplitude, A, d'une onde est la distance maximale parcourue par une particule dans le milieu par rapport à son point d'origine ou de repos. Il est important de noter que les particules du milieu traversé par une onde longitudinale ne se déplacent pas dans le milieu. Elles oscillent plutôt d'un côté et de l'autre de leurs points d'origine. D'ailleurs, le nombre d'oscillations par seconde est nommé la fréquence f. Cette dernière est directement liée à la période, T, par la relation suivante : $f = 1/T$. La période d'une onde indique le temps requis pour que les particules effectuent une oscillation complète. La fréquence est exprimée en Hertz (Hz) tandis que la période est exprimée en secondes (s).

L'onde se propage dans le milieu à une certaine vitesse c. Cette vitesse dépend du milieu de transmission. La relation entre le milieu de transmission et la vitesse sera expliquée sous peu. La vitesse et la fréquence peuvent être reliées par l'équation suivante $c = \lambda f$ où λ est la longueur d'onde qui correspond à la distance parcourue par l'onde pour effectuer une oscillation.

2.1.2 Caractérisques du milieu

Le milieu de propagation a une grande influence sur la propagation de l'onde. La caractéristique première d'un milieu à considérer est le module d'élasticité isostatique (aussi appelé module de rigidité à la compression). Le module d'élasticité isostatique k est une constante qui mesure la résistance d'un matériau à une compression. La valeur est obtenue en calculant le changement de volume, dV pour une augmentation

Tableau 2.2 – Impédance acoustique et vitesse de propagation d'onde dans différents milieux du corps humain. Tiré de [16].

Milieu	Impédence acoustique - Z ($\mathrm{kgm^{-2}s^{-1}}$)	Vitesse - c ($\mathrm{m\,s^{-1}}$)
Eau	1.48×10^6	1480
Air	430	333
Gras	1.33×10^6	1430
Foie	1.66×10^6	1578
Rein	1.64×10^6	1560
Os	6.47×10^6	3190 - 3406

de pression dP ; $k = -V\frac{dV}{dP}$ où V est le volume initial. La seconde caractéristique est la densité $\rho = \frac{m}{V}$, soit la quantité de masse m par unité de volume V. À partir de ces valeurs, il est possible de déterminer la vitesse de propagation d'une onde mécanique $c = \sqrt{k/\rho}$. On peut ensuite calculer la valeur de l'impédance acoustique $Z = \rho c$, soit la résistance d'un milieu à la propagation d'ondes. Les valeurs d'impédance acoustique et de vitesse de propagation de certains milieux du corps humain sont présentées au tableau 2.2.

2.1.3 Interactions entre onde et milieu

La clef de l'imagerie par ultrason repose sur la propagation d'ondes d'un milieu vers un autre. Le changement de caractéristiques crée des échos qui sont retournés vers le point d'origine de l'onde. Cependant, plusieurs interactions sont possibles

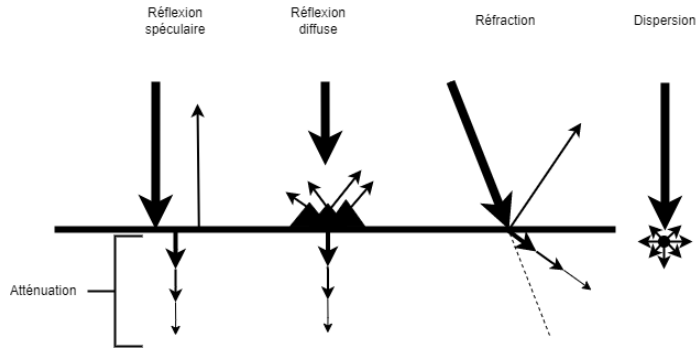


Figure 2.1 – Illustrations des différentes interactions entre une onde et son milieu

entre l'onde et son milieu. Certaines de ces interactions sont l'atténuation, la réflexion diffuse et spéculaire, la réfraction et la dispersion [16]. Ces interactions sont expliquées ci-dessous et illustrées à la figure 2.1

Atténuation : L'atténuation est le processus par lequel une onde perd de l'énergie lorsqu'elle se propage dans un milieu. La principale source de perte d'énergie est l'absorption, soit le transfert de l'énergie mécanique en énergie thermique. L'atténuation en fonction de la profondeur à la forme d'une fonction exponentielle décroissante. C'est-à-dire que l'atténuation est plus rapide au début de la propagation et ralentit en fonction de la profondeur. On exprime l'atténuation en décibel (dB). Comme le décibel à une forme logarithmique, l'atténuation en décibel est une valeur constante pour un milieu donné. Toutefois, il faut noter que cette valeur assume une fréquence d'onde constante. En effet, l'atténuation a aussi une dépendance sur la fréquence de l'onde puisque l'atténuation sera plus grande pour des fréquences plus élevées. On peut mesurer l'atténuation d'un milieu en fonction de la fréquence en $\mathrm{dB\,cm^{-1}\,Hz}$. Par exemple, l'atténuation de sang est de $0.15\,\mathrm{dB\,cm^{-1}\,Hz}$ [16].

Réflexion : La réflexion se produit quand une onde traverse l'interface entre deux milieux avec des impédances acoustiques différentes. Une partie de l'onde traverse l'interface tandis que l'autre est réfléchie et retourne dans la direction inverse. La proportion de l'onde qui traverse, T par rapport à la portion qui est réfléchie, R dépend des impédances du premier et deuxième milieu, Z_1 et Z_2 respectivement. Ces deux valeurs sont données par

$$R = \left(\frac{Z_1 - Z_2}{Z_1 + Z_2}\right), \quad T = \frac{2Z_1 Z_2}{Z1 + Z_2}. \tag{2.1}$$

La proportion de l'onde transmise et réfléchie se reflète par un changement d'amplitude en fonction de R et de T. Il est important de noter que $R + T = 1$. Ceci indique qu'aucune portion de l'onde n'est perdue.

Ce type de réflexion se nomme la réflexion spéculaire. Cependant, ceci se produit seulement si l'onde incidente arrive à un angle droit ($90°$) par rapport à l'interface et que l'interface est lisse par rapport à la longueur d'onde λ. Si la surface n'est pas lisse, la réflexion se fait dans plusieurs directions et se nomme réflexion diffuse.

Réfraction : La réfraction se produit lorsque qu'une onde atteint une interface à un angle autre que 90°. Dans ce cas, l'onde transmise change de direction. Cela se produit dû au changement de vitesse de l'onde entre les deux milieux. Comme l'onde arrive à un angle, une portion de l'onde traverse et change de vitesse avant le reste. Ceci cause une déviation dans la trajectoire.

Afin de déterminer l'angle de réfraction de l'onde θ_2, on définit l'angle d'incidence θ_1 de l'onde par rapport à la normale de la surface (un vecteur perpendiculaire). L'équation suivante décrit la relation entre θ_1 et θ_2 par rapport à la vitesse de propagation dans le premier milieu c_1 et dans le deuxième milieu c_2

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}. \tag{2.2}$$

Il faut aussi noter que la portion réfléchie de l'onde le sera avec un angle de $-\theta_1$ par rapport à la normale.

Dispersion : La dispersion se produit quand l'onde atteint des particules qui sont plus petites que la longueur d'onde. L'onde est alors dispersée dans toutes les directions. Ceci crée une onde circulaire. Quand plusieurs phénomènes de dispersion se produisent en même temps, les ondes circulaires se croisent et s'additionnent. Cette addition d'ondes se nomme l'interférence. Celle-ci produit des effets dans l'image ultrasonore qu'on nomme *speckle*.

2.1.4 Émission et réception d'ondes

La génération et la réception d'ondes ultrasonores reposent sur l'effet piézoélectrique. L'effet piézoélectrique est un effet qui se produit dans certains matériaux qui se déforment sous la présence de polarisation électrique et qui, inversement, crée un potentiel électrique lorsqu'ils sont déformés. Le quartz est un exemple de matériau ayant des caractéristiques piézoélectriques. Dans la fabrication de sonde ultrasonore utilisée en médecine, une céramique synthétique, telle que le titano-zirconate de plomb (PZT), est plutôt utilisée.

Afin de générer et réceptionner des ondes, une couche de matériau piézoélectrique est placée entre deux bornes conductrices (figure 2.2). Il est alors possible de créer des

contractions et des expansions du piézoélectrique à l'aide d'un courant alternatif d'une certaine fréquence. Les oscillations du piézoélectrique en contact avec le tissu créent des ondes ultrasonores. Au moment de la réception d'ondes, aucun potentiel n'est appliqué. Les ondes atteignant le piézoélectrique génèrent un potentiel électrique mesurable qui est proportionnel aux variations de pression appliquées au piézoélectrique.

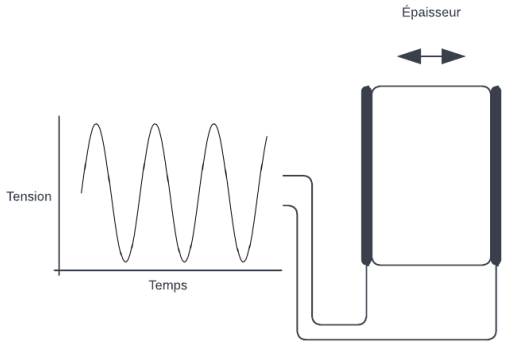


Figure 2.2 – Illustration d'un élément piézoélectrique

2.1.5 Génération d'images

Il existe plusieurs modes d'ultrason. Le mode B est le mode le plus commun permettant d'obtenir une image qui représente les structures anatomiques. Cependant, avant d'introduire le mode B, il est plus simple de comprendre le fonctionnement du mode A. Dans les deux cas, le spectre de fréquences utilisé dans le domaine médical est de 1 MHz à 15 MHz. C'est d'ailleurs d'où provient le nom *ultrason* puisque les fréquences utilisées sont plus grandes que celles perceptibles par l'oreille humaine (maximum 20 000 Hz).

Mode A : Le mode A utilise un seul élément piézoélectrique qui émet une onde dans le médium. Une fois, l'onde émise, l'élément piézoélectrique est mis en mode réception. L'élément piézoélectrique transforme alors l'onde mécanique reçue en onde électrique. Dans cette onde, l'amplitude représente la proportion de l'onde réfléchie par les différentes interfaces du médium tandis que le temps représente la distance

parcourue par l'onde. La distance, d, entre une interface et la sonde peut être calculée par

$$d = \frac{t \cdot c}{2}, \tag{2.3}$$

où t est le temps entre l'émission de l'onde et la mesure de la réflexion. Il est important de noter qu'on assume une vitesse constante de $c = 1540\,\mathrm{m\,s^{-1}}$ pour tous les tissus parcourus par l'onde.

Mode B : L'imagerie ultrasonore en mode B utilise le même principe de fonctionnement, mais utilise plusieurs éléments piézoélectriques afin de générer une image en deux dimensions au lieu d'un signal en une dimension comme le mode A. Un balayage est effectué à travers les éléments effectués en envoyant une séquence de faisceaux d'ultrasons dans le tissu afin de parcourir le médium dans la direction latérale. Dans ce cas, chaque point d'amplitude et de distance représente un pixel dont la valeur latérale est déterminée par position latérale du balayage. Une illustration simplifiée du balayage est montrée à la figure 2.3. En réalité, plusieurs éléments sont utilisés en parallèle pour l'émission et la réception de l'onde. La synchronisation des éléments dépend du type de sonde utilisée.

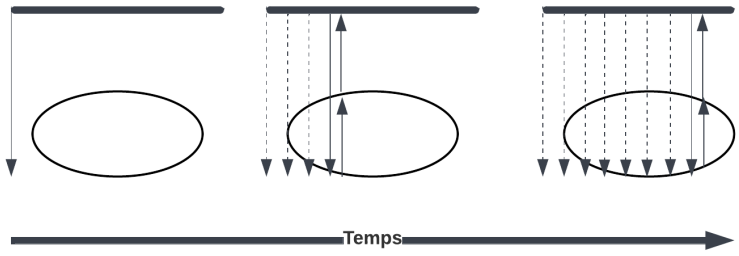


Figure 2.3 – Illustration simplifiée du processus de balayage latéral

Sondes séquentielles

Ce type de sonde utilise plusieurs éléments piézoélectriques alignés. L'alignement peut être linéaire (figure 2.3 a) ou courbé (convexe) (figure 2.3 b).

Dans les deux cas, le fonctionnement est similaire. Chaque élément piézoélectrique émet une onde circulaire. Si plusieurs éléments émettent de manière synchronisée, les ondes s'additionnent pour créer une onde plane. Le balayage latéral se fait en activant un sous-groupe d'éléments et en déplaçant ce sous-groupe latéralement dans l'alignement des éléments. Il est possible de définir un point d'accent par chaque faisceau au centre du sous-groupe.

Sondes à réseau phasé

Les sondes à réseau phasé génèrent des images sectorielles comme les sondes convexes (figure 2.3 c). Ces sondes utilisent des éléments piézoélectriques beaucoup plus rapprochés. Lors du balayage, chaque élément piézoélectrique est utilisé et le balayage se fait en décalant l'activation des éléments. Ceci permet d'orienter la direction de l'onde plane générée. Ce type de sonde effectue aussi un accent sur un point qui est ajustable en modifiant les paramètres de synchronisation des ondes.

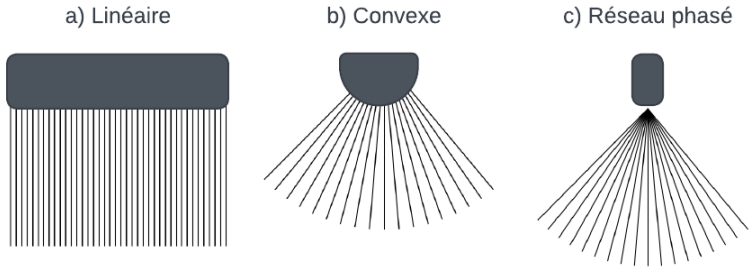


Figure 2.4 – Schéma de différents types de sondes et la direction de leurs faisceaux

Rémunération de gain de temps

Tel que mentionné précédemment, l'atténuation affecte la propagation d'onde. Les structures plus profondes apparaitront donc avec une intensité plus faible. Cependant, comme le but de l'imagerie en mode B est d'utiliser l'intensité du pixel pour représenter l'amplitude de la réflexion peu importe la profondeur, il est nécessaire de compenser ce facteur. Cette compensation se fait en amplifiant le signal d'échos provenant de

structures en fonction de leur profondeur. La profondeur est déduite par rapport au temps de retour des échos, d'où le nom rémunération de gain de temps.

2.1.6 Autres modes d'ultrason

Autre que le mode A et B, il existe également le mode M et le mode Doppler. Le mode M (M pour motion) permet d'analyser le mouvement de structures dans le corps. Lors de l'utilisation de ce mode, une ligne dans l'image est sélectionnée et l'acquisition dans le temps le long de cette ligne est visualisée sous forme d'image 2D.

Le mode Doppler utilise l'effet Doppler pour visualiser le mouvement de tissus et de fluides dans le corps. L'effet Doppler est le principe qu'une onde changera de fréquence si elle est réfléchie par un objet en mouvement. Il est alors possible de voir si une structure ou un fluide se déplace vers la sonde ou s'éloigne.

2.2 Échocardiographie

L'échocardiographie est le nom donné à l'imagerie ultrasonore du coeur. Il s'agit d'un type d'imagerie complexe puisque le coeur est entouré d'air (poumons) et d'os (cage thoracique), deux milieux qui rendent l'acquisition difficile.

Il est donc nécessaire d'acquérir les images du coeur dans des fenêtres bien précises. L'acquisition se fait normalement à partir du torse (trans thoracique) ou parfois par l'oesophage (transe œsophagienne). À noter que ce mémoire porte exclusivement sur l'échographie trans thoracique.

Une sonde de type réseau phasé linéaire est souvent utilisée puisque la forme image sectorielle permet aux faisceaux de passer entre les côtes et d'avoir un champ de vision large au niveau du coeur. Dû à la profondeur du coeur par rapport au point d'acquisition, une fréquence de 2.5 MHz est utilisée pour éviter trop d'atténuation.

Les trois fenêtres principales sont la fenêtre parasternale, la fenêtre apicale et la fenêtre sous-costale. La fenêtre parasternale se trouve légèrement à la gauche du sternum, normalement dans l'espace intercostal 3 ou 4 (espace entre les côtes). La fenêtre apicale se trouve légèrement sous la poitrine vis-à-vis l'apex du coeur. Finalement, la fenêtre sous-costale se trouve sous le sternum. Le coeur est alors vu a

travers le foie.

Comme l'ultrason en mode B acquiert une image en deux dimensions d'une structure en trois dimensions, l'orientation de la sonde change l'image. On appelle ces orientations des vues. Certaines vues principales sont expliquées en détail ci-dessous et sont illustrées à la figure 2.5.

Grand axe parasternal : Afin d'obtenir cette vue, l'orientation de la sonde devrait couvrir l'axe entre la base et l'apex. On peut observer le ventricule gauche, l'atrium gauche, le ventricule droit et parfois l'atrium droit. On peut aussi voir la valve mitrale et l'aorte.

Petit axe parasternal : Cette vue est obtenue avec une orientation de sonde perpendiculaire à la vue grand axe. On observe alors une coupe transversale du ventricule gauche, qui aura une forme de cercle, et du ventricule droit.

Quatre chambres apicales : Cette vue permet de voir les quatre chambres ainsi que les valves mitrale et tricuspide.

Deux chambres apicales : Cette vue permet de voir l'atrium et le ventricule gauche ainsi que la valve mitrale. Cette acquisition se trouve à être perpendiculaire à la vue apicale quatre chambres ce qui permet une compréhension du ventricule gauche en trois dimensions.

2.2.1 Diagnostic en échocardiographie

L'échocardiographie permet de diagnostiquer plusieurs pathologies énumérées à la section 1.4. Les images acquises en mode B permettent d'analyser la grandeur des différentes cavités ainsi que l'épaisseur du myocarde. Le volume ventricule gauche peut notamment être calculer à l'aide de la segmentation du ventricule gauche en vue apicale deux chambres et quatre chambres en utilisant la méthode de Simpson [23].

Le mouvement des valves peut aussi être visualisé en mode B, mais pour une analyse plus détaillée, le mode M peut être utilisé. Le flux sanguin peut être analysé

avec le mode Doppler qui permet notamment de voir s'il y a présence de régurgitation dans les valves.

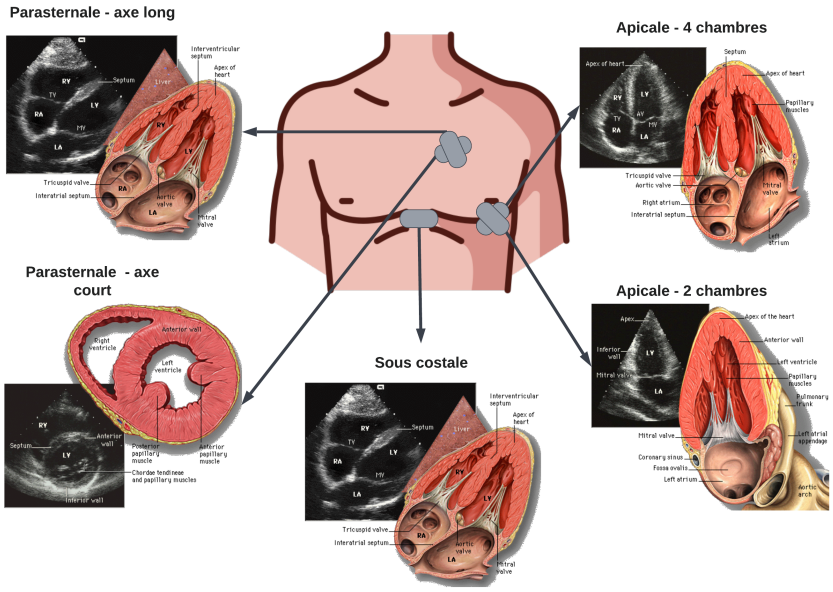


Figure 2.5 – Différentes vues et positionnement de sonde d'échocardiographie transthoracique. Source : P. J. Lynch et C. C. Jaffe. « Apical two chamber view of heart», « Left parasternal long axis view of heart», « Short axis view of left ventricle of heart», « Subcostal view of heart» [1]

1. Wikimedia Commons, *creative Commons Attribution 2.5 License 2006.* Récupéré de https://commons.wikimedia.org/wiki/File:Apical_4_chamber_view.png, https://commons.wikimedia.org/wiki/File:Apical2Chamber.png, https://commons.wikimedia.org/wiki/File:LeftParasternalLongAxis.gif, https://commons.wikimedia.org/wiki/File:LeftVentricleShortAxis.gif et https://commons.wikimedia.org/wiki/File:Subcostal_view_of_heart.gif.

Chapitre 3

Apprentissage automatique

Depuis quelques années, l'intelligence artificielle est à l'avant-garde de nombreuses nouvelles technologies, notamment dans le domaine de l'imagerie médicale. Le principal moteur pour ces percées est l'apprentissage machine, en particulier l'apprentissage supervisé par réseaux de neurones. Ce chapitre introduira les bases de l'apprentissage supervisé.

3.1 Apprentissage supervisé

L'apprentissage supervisé est un ensemble de techniques qui visent à apprendre la correspondance entre un ensemble de données d'entrées et de sorties. Cette correspondance est apprise à partir d'une base de données comprenant N vecteurs d'entrées, $\vec{x}$, et vecteurs cibles correspondants $\vec{y}$:

$$\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\} = \{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), ..., (\vec{x}_n, \vec{y}_n)\}. \tag{3.1}$$

Cette base de données est utilisée pour entraîner une fonction paramétrique $f_\theta(\vec{x})$. Cette fonction est dite paramétrée, car elle dépend de l'entrée $\vec{x}$, mais aussi d'un ou de plusieurs paramètres θ. Une fois entraînée, cette fonction peut être utilisée pour

estimer la cible, $\hat{y}^*$, correspondant à une nouvelle entrée $\vec{x}^*$ avec

$$\hat{y}^* = f_\theta(\vec{x}^*). \tag{3.2}$$

L'habileté d'un modèle à prédire correctement des cibles pour de nouvelles entrées se nomme la généralisation. Afin d'évaluer la performance en généralisation, on divise l'ensemble de données $\mathcal{D}$ en un ensemble d'entraînement, $\mathcal{D}_{train}$, et un ensemble pour évaluer la généralisation, l'ensemble de test $\mathcal{D}_{test}$. En pratique, une seconde division est réalisée pour créer un ensemble dit de validation $\mathcal{D}_{val}$. Cet ensemble de données permet d'identifier les meilleurs hyperparamètres, soit les paramètres de configuration de $f_\theta(\vec{x})$. Plus de détails sur la notation d'hyperparamètres seront donnés dans ce chapitre. À des fins de concision, la notation de $\mathcal{D}$ sera utilisée pour désigner l'ensemble d'entrainement à moins d'indications contraires.

Pour entraîner la fonction $f_\theta(\vec{x})$, il est nécessaire de définir un objectif. Celui-ci prend souvent la forme d'une fonction de coût. Cette fonction de coût mesure à quel point la sortie prédite du modèle, $f_\theta(\vec{x}_i)$, correspond à la cible, $\vec{y}_i$, pour une entrée $\vec{x}_i$. Cette fonction $\mathcal{L}$ permet d'identifier les paramètres idéaux du modèle θ^* qui sont donnés par

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\vec{x}_i), \vec{y}_i). \tag{3.3}$$

Le choix de la fonction de coût dépend de plusieurs facteurs dont le type de tâches et les suppositions mises sur les données. Une approche possible est l'approche probabiliste qu'on appelle maximum de vraisemblance.

3.1.1 Maximum de vraisemblance

L'approche par maximum de vraisemblance suppose que les données de l'ensemble d'entraînement $\mathcal{D}$ proviennent d'un processus aléatoire dicté par des paramètres inconnus. Le but de la technique du maximum de vraisemblance est de trouver ces paramètres. Pour ce faire, on optimise les paramètres du modèle θ afin de maximiser la probabilité des données cibles en fonction des données d'entrée et du modèle. Ceci

est exprimé mathématiquement comme suit

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, p(\boldsymbol{Y}|\boldsymbol{X}, \theta). \tag{3.4}$$

Si toutes les données indépendantes et identiquement distribuées (i.i.d.) on peut réécrire cette probabilité sous forme de produit

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(\vec{y}_i|\vec{x}_i, \theta). \tag{3.5}$$

En reformulant par rapport au modèle paramétré par θ on obtient

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(\vec{y}_i|f_\theta(\vec{x}_i)). \tag{3.6}$$

Finalement, le produit peut être transformé en somme à l'aide d'une fonction logarithmique

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(\vec{y}_i|f_\theta(\vec{x}_i)). \tag{3.7}$$

Les prochaines étapes dépendent de la tâche et des suppositions posées sur les données qui dicteront la forme de $p(y_i|f_\theta(x_i))$. Le cas de la régression et de la classification seront étudiés.

Régression

Le but de la régression est de prédire une valeur continue à partir des entrées. Dans le cas standard, les entrées sont des vecteurs de D dimensions, $\vec{x} \in \mathbb{R}^D$, et les sorties sont des scalaires $y \in \mathbb{R}$. Il est fréquent de supposer que les données sont issues d'une distribution gaussienne de variance σ^2. Cette variance représente le bruit supposé dans les données. La distribution est donnée par

$$\begin{aligned} p(y_i|f_\theta(x_i)) &= \mathcal{N}(y_i|f_\theta(\vec{x}_i), \sigma^2) \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(f_\theta(\vec{x}_i)-y_i)^2}{2\sigma^2}}. \end{aligned} \tag{3.8}$$

En reprenant l'équation 3.7, on obtient

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(f_\theta(\vec{x}_i)-y_i)^2}{2\sigma^2}} \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \frac{1}{\sigma\sqrt{2\pi}} + \sum_{i=1}^{n} -\frac{(f_\theta(\vec{x}_i)-y_i)^2}{2\sigma^2}.\end{aligned} \tag{3.9}$$

En éliminant tous les termes qui ne dépendent pas de θ, on retrouve

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} -(f_\theta(\vec{x}_i)-y_i)^2. \tag{3.10}$$

Puisque maximiser une valeur correspond à minimiser son opposé, on peut donc exprimer cette maximisation sous forme de fonction de coût compatible avec l'équation 3.3. On retrouve ici la fonction de coût appelée erreur quadratique (en anglais *squared error* ou **SE**)

$$\mathcal{L}_{SE}(f_\theta(\vec{x}_i), y_i) = (f_\theta(\vec{x}_i)-y_i)^2. \tag{3.11}$$

Classification binaire

Le but de la classification est d'assigner une classe à une entrée $\vec{x} \in \mathbb{R}^D$. La classification est dite binaire s'il y a deux classes possibles et catégoriques si plus de deux classes sont possibles. Dans le cas de la classification binaire $y \in \{0, 1\}$. Bien qu'il soit possible d'utiliser la formulation précédente pour la classification, ceci supposerait que les données sont indépendantes et identiquement distribuées (iid.) d'une distribution gaussienne. Comme ceci n'est pas toujours le cas, il est préférable de supposer que les données sont issues d'une distribution de Bernouilli. La fonction à maximiser est donc

$$p(y_i|f_\theta(x_i)) = f_\theta(\vec{x}_i)^{y_i}(1-f_\theta(\vec{x}_i))^{y_i}. \tag{3.12}$$

En reprenant l'équation 3.7, on obtient

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log f_\theta(\vec{x}_i)^{y_i}(1 - f_\theta(\vec{x}_i))^{(1-y_i)} \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} y_i f_\theta(\vec{x}_i) + (1 - y_i)\log(1 - f_\theta(\vec{x}_i)).\end{aligned} \tag{3.13}$$

En inversant encore la maximisation pour une minimisation, on retrouve la fonction de coût appelée entropie croisée binaire (en anglais *binary cross-entropy* ou *BCE*)

$$\mathcal{L}_{BCE}(f_\theta(\vec{x}_i), y_i) = y_i \log(f_\theta(\vec{x}_i)) + (1 - y_i)\log(1 - f_\theta(\vec{x}_i)). \tag{3.14}$$

Contrairement à la régression par modèle gaussien pour laquelle le modèle f_θ prédit une moyenne, la classification binaire par BCE import au modèle de prédire la probabilité qu'$\vec{x}$ appartient à la classe C_1. Il faut donc contraindre la sortie du modèle afin que sa celle-ci soit entre 0 et 1 afin de respecter la définition de distribution de probabilité. En assumant que la sortie de f_θ est un réel, on peut utiliser la fonction *sigmoïde* qui transforme l'ensemble des réels dans l'intervalle $[0, 1]$. La fonction *sigmoïde* est définie par

$$\text{sigmoid}(z) = \sigma(z) = \frac{1}{1 - e^{-z}}. \tag{3.15}$$

Comme la classification binaire permet de prédire l'une des deux classes, c_0 et c_1, la fonction *sigmoïde*, permet de modéliser la probabilité conditionnelle associée à l'une d'entre elles $p(c_1|x)$ (donc $p(c_0|x) = 1 - p(c_1|x)$). En reprenant le théorème de Bayes, on peut voir comment la sigmoïde permet cette modélisation comme suit

$$\begin{aligned}p(c_1|\vec{x}) &= \frac{p(\vec{x}|c_1)p(c_1)}{p(\vec{x})} \\ &= \frac{p(\vec{x}|c_1)p(c_1)}{p(\vec{x}|c_0)p(\vec{x}) + p(\vec{x}|c_1)p(c_1)} \\ &= \frac{1}{1 + \frac{p(\vec{x}|c_0)p(\vec{x})}{p(\vec{x}|c_1)p(c_1)}}.\end{aligned} \tag{3.16}$$

En posant $z = \ln \frac{p(\vec{x}|c_0)p(\vec{x})}{p(\vec{x}|c_1)p(c_1)}$, on retrouve $\frac{1}{1-e^{-z}}$, soit la forme de la *sigmoïde.* On appelle cette formulation de la classification binaire, la régression logistique.

Classification multiclasses

Il est aussi possible d'effectuer la classification multiclasses si plus de deux classes sont possibles. Dans le cas de la classification à K classes, le modèle doit avoir K sorties, c'est à dire $f_\theta(\vec{x})R^d \rightarrow R^K$. Afin d'avoir en sortie une distribution de probabilités valide, il est nécessaire d'utiliser une fonction d'activation. Dans ce cas, la sigmoïde ne permet pas d'obtenir une distribution valide. On utilise plutôt la fonction softmax donnée par

$$\text{softmax}(\vec{z})_i = \frac{e^{\vec{z}_i}}{\sum_{j=1}^{K} e^{\vec{z}_j}}. \tag{3.17}$$

La fonction de coût utilisée est l'entropie croisée. Ici les cibles sont encodées en format *one-hot*, c'est-à-dire que chaque cible $\vec{y}_i$ est un vecteur de 0 à l'exception d'une valeur de 1 à l'indice qui indique la classe. Cette fonction est donnée par

$$\mathcal{L}_{CE}(f_\theta(\vec{x}_i), \vec{y}_i) = \sum_{i=1}^{K} \vec{y}_{ik} log(f_\theta(\vec{x}_i)_k). \tag{3.18}$$

3.1.2 Maximum *a posteriori*

Au lieu de maximiser la vraisemblance des données en fonction des paramètres, $p(\boldsymbol{Y}|\boldsymbol{X}, \theta)$, tel que présenté dans la section précédente, il est aussi possible de maximiser la probabilité des paramètres en fonction des données $p(\theta|\boldsymbol{X}, \boldsymbol{Y})$. On nomme cette formulation la probabilité *a posteriori*. On peut définir la relation entre la vraisemblance et la probabilité *a posteriori* avec le théorème de Bayes

$$p(\theta|\boldsymbol{X}, \boldsymbol{Y}) = \frac{p(\boldsymbol{Y}|\boldsymbol{X}, \theta)p(\theta)}{P(\boldsymbol{Y}|\boldsymbol{X})}. \tag{3.19}$$

On remarque que la maximisation de la vraisemblance est équivalente à la maximisation de la probabilité *a posteriori*

$$p(\theta|\boldsymbol{X}, \boldsymbol{Y}) \propto p(\boldsymbol{Y}|\boldsymbol{X}, \theta)p(\theta). \tag{3.20}$$

On peut donc maximiser la probabilité *a posteriori* comme suit

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\, p(\theta|\boldsymbol{X}, \boldsymbol{Y}) \\ &= \underset{\theta}{\operatorname{argmax}}\, p(\boldsymbol{Y}|\boldsymbol{X}, \theta)p(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(\vec{y}_i|\vec{x}_i, \theta)p(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \underbrace{\log p(\vec{y}_i|f_\theta(\vec{x}_i))}_{\text{Log vraisemblence}} + \log p(\theta).\end{aligned} \tag{3.21}$$

On remarque que le premier terme de l'équation à maximiser est la vraisemblance. Cette portion dépend de la tâche comme il l'a été démontré à la section précédente. Le deuxième terme est l'*a priori* sur les paramètres. Si l'on suppose que les paramètres sont issus d'une distribution gaussienne multivariée centrée sur $\vec{0}$ ayant une matrice de covariance de $\boldsymbol{\Sigma}$, $\mathcal{N}(\vec{0}, \boldsymbol{\Sigma})$ on peut maximiser que cette portion de l'équation qui se simplifie comme suit

$$\begin{aligned}\log p(\theta) &= \log \mathcal{N}(\vec{0}, \boldsymbol{\Sigma}) \\ &= \log \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\theta-0)^T\boldsymbol{\Sigma}^{-1}(\theta-0)} \\ &= \log \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} - \frac{1}{2}\theta^T\boldsymbol{\Sigma}^{-1}\theta.\end{aligned} \tag{3.22}$$

On émet normalement l'hypothèse que $\mathcal{N}(\vec{0}, \boldsymbol{\Sigma})$ est une gaussienne multivariée isotropique et donc que $\boldsymbol{\Sigma}$ est une matrice diagonale ayant une variance constante de α^2, donc $\boldsymbol{\Sigma} = \alpha^2\boldsymbol{I}$. Si on élimine tous les termes qui ne dépendent pas de θ, qu'on pose l'équation 3.21 sous forme de minimisation on obtient

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \underbrace{-\log p(\vec{y}_i|f_\theta(\vec{x}_i))}_{\text{Log vraisemblence}} - \frac{1}{\alpha^2}\theta^T\theta. \tag{3.23}$$

Il est fréquent de poser $\lambda = \frac{1}{\alpha^2}$ et de nommer ce terme la constante de régularisation.

En effet, le terme $\frac{1}{\alpha^2}\theta^T\theta$ a pour effet de régulariser les paramètres en pénalisant ceux qui s'éloignent de 0.

3.2 Modèles linéaires

Jusqu'à présent, le modèle $f_\theta(x)$ est une fonction paramétrique arbitraire. Plusieurs fonctions peuvent être utilisées pour définir $f_\theta(x)$, la plus simple étant le modèle linéaire. Le modèle linéaire est tout simplement un produit scalaire entre le vecteur d'entrée, $\vec{x}$, et un vecteur de paramètres $\vec{w} \in R^D$ (où D est la taille $\vec{x}$) et l'addition d'un biais $b \in R$. Ce modèle a vu le jour sous le nom du Perceptron [44], mais a subi plusieurs modifications depuis.

Régression. Pour la régression, la sortie du modèle linéaire $\hat{y} \in R$ est définie comme suit

$$\begin{aligned} \hat{y} &= w_1 x_1 + w_2 x_2 + ... + w_n x_n + b \\ \hat{y} &= \vec{w}^T \vec{x} + b. \end{aligned} \tag{3.24}$$

Afin d'alléger la notation, il est commun d'ajouter le biais au vecteur $\vec{w}$ et d'ajouter un 1 au vecteur d'entrée. Ceci permet de réécrire $\hat{y}$ comme suit

$$\hat{y} = \vec{w}^T \vec{x}. \tag{3.25}$$

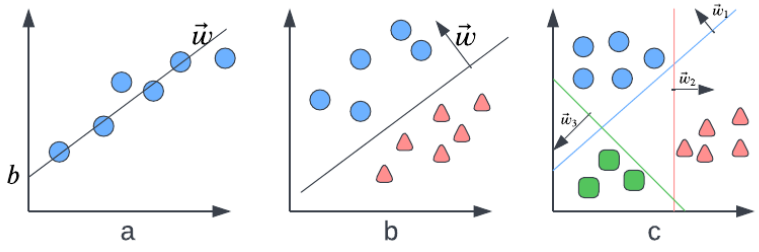


Figure 3.1 – Exemples de modèles linéaires. a) Régression linéaire en 1 dimension. b) Classification binaire linéaire en 2 dimensions c) Classification multiclasse linéaire en 2 dimensions.

Dans ce cas, les paramètres du modèle sont $\theta = \{\vec{w}\}$ (en réalité $\theta = \{\vec{w}, b\}$). Pour la régression, les paramètres $\vec{w}$ et b représentent la droite qui correspond le mieux aux données. En une dimension, le modèle a la forme $y = wx + b$ où w est la pente et b est l'ordonnée à l'origine tel qu'illustré à la figure 3.1 a.

Classification binaire. Dans le cas de la classification binaire, $\vec{w}$ représente la normale d'un hyperplan qui sépare le mieux les deux classes. Un exemple en deux dimensions est illustré à la figure 3.1 b. En effectuant le produit scalaire entre le vecteur d'entrée et le vecteur de paramètres, il est possible de déterminer de quel côté du plan se trouve le vecteur. Si le produit scalaire $\vec{w}^T\vec{x}$ est supérieur à zéro, le point $\vec{x}$ est au-dessus du plan. Sinon, il se trouve en dessous. En y ajoutant une fonction d'activation, le classifieur linéaire est exprimé comme suit

$$\hat{y} = \sigma(\vec{w}^T\vec{x}). \tag{3.26}$$

Ici, la fonction d'activation utilisée est la *sigmoïde*, mais d'autres fonctions d'activation telles que la fonction *heavy-side*, utilisée par le Perceptron, peuvent être utilisées aussi (ceci demande cependant une fonction de coût différente). La fonction *heavy-side*, $h(z)$, est définie comme suit

$$h(z) = \begin{cases} +1 & \text{si } z > 0 \\ -1 & \text{si } z \leq 0. \end{cases} \tag{3.27}$$

Classification multiclasses. Dans le cas de la classification multiclasses, le modèle doit avoir K sorties où K est le nombre de classes. Chaque classe doit avoir un hyperplan $\vec{w}_k$ pour représenter la séparation entre elles. On peut combiner cet ensemble de vecteurs en une matrice $W \in R^{K\times D}$ et effectuer une multiplication matricielle au lieu de plusieurs produits scalaires

$$\hat{y} = \text{Softmax}(\boldsymbol{W}\vec{x}). \tag{3.28}$$

Tel que mentionné à la section 3.1.1, la fonction *Softmax* permet d'obtenir une distribution de probabilités en sortie. Un exemple de classification multiclasses en

deux dimensions est illustré à la figure 3.1 c.

Au moment d'évaluer la classe prédite, la fonction *argmax* est utilisée pour identifier l'indice de la valeur la plus grande dans le vecteur de prédictions.

3.3 Optimisation

Tel qu'il a été mentionné précédemment, le but de l'apprentissage supervisé est de trouver les paramètres, θ^*, permettant à un modèle f_θ de minimiser une fonction de coût $\mathcal{L}$ sur un ensemble de données $\mathcal{D}$. Comme l'espace de recherche pour la solution optimale peut être immense même pour un nombre de paramètres relativement petit, la recherche aléatoire n'est pas une option. On utilise plutôt le gradient de la fonction de coût par rapport aux paramètres.

Définition 1: Gradient

Le gradient d'une fonction multivariée $f(\vec{x}) : R^N \to R$ est un vecteur où chaque valeur est la dérivée partielle de f par rapport à chaque valeur de $\vec{x}$.

$$\nabla_{\vec{x}} f(\vec{x}) : R^N \to R^N = \begin{bmatrix} \frac{\partial f(\vec{x})}{\partial \vec{x}_1} \\ \frac{\partial f(\vec{x})}{\partial \vec{x}_2} \\ \vdots \\ \frac{\partial f(\vec{x})}{\partial \vec{x}_N} \end{bmatrix} \tag{3.29}$$

Il est généralement supposé que la fonction de coût est convexe. Ceci signifie que le minimum de la fonction se trouve où le gradient de la fonction est égal à 0. Il est donc possible de trouver les paramètres optimaux en isolant $\vec{w}$ dans l'équation $\nabla_{\vec{w}} \mathcal{L} = 0$. Cette solution se nomme la solution fermée. Cependant, dans la majorité des cas, tels que les réseaux de neurones, la fonction de coût n'est pas réellement convexe et il n'est pas possible de trouver une solution fermée. On utilise alors le fait que le gradient indique la direction d'ascension ponctuelle la plus rapide dans un processus nommé la descente de gradient.

3.3.1 Solution fermée

Dans le cas de la régression linéaire avec la fonction de coût d'erreur quadratique, il est possible de trouver la solution exacte. Rappelons que nous tentons de minimiser

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\vec{w}^T\vec{x}_i - y_i)^2 \tag{3.30}$$

par rapport à $\vec{w}$. On commence par dériver le gradient de $\vec{w}$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\vec{w}} &= \frac{\partial}{\partial\vec{w}}\frac{1}{N}\sum_{i=1}^{N}(\vec{w}^T\vec{x}_i - y_i)^2 \\ &= \frac{2}{N}\sum_{i=1}^{N}(\vec{w}^T\vec{x}_i - y_i)\vec{x}_i^T \\ &= \frac{2}{N}(\vec{w}^T\sum_{i=1}^{N}\vec{x}_i\vec{x}_i^T - \sum_{i=1}^{N}y_i\vec{x}_i^T).\end{aligned} \tag{3.31}$$

Puisqu'on cherche le point où $\frac{\partial\mathcal{L}}{\partial\vec{w}} = 0$, on l'exprimie comme suit

$$\begin{aligned}0 &= \frac{2}{N}(\vec{w}^T\sum_{i=1}^{N}\vec{x}_i\vec{x}_i^T - \sum_{i=1}^{N}y_i\vec{x}_i^T) \\ 0 &= \vec{w}^T\boldsymbol{X}^T\boldsymbol{X} - \vec{y}^T\boldsymbol{X} \\ 0 &= \boldsymbol{X}^T\boldsymbol{X}\vec{w} - \vec{y}\boldsymbol{X}^T.\end{aligned} \tag{3.32}$$

où $\boldsymbol{X} \in R^{N\times D}$ est une matrice où chaque rangée est un vecteur dans la base de données et $\vec{y} \in R^N$ est un vecteur cible. On isole maintenant $\vec{w}$

$$\begin{aligned}0 &= \boldsymbol{X}^T\boldsymbol{X}\vec{w} - \vec{y}\boldsymbol{X}^T \\ \boldsymbol{X}^T\boldsymbol{X}\vec{w} &= \vec{y}\boldsymbol{X}^T \\ \vec{w} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\vec{y}\boldsymbol{X}^T.\end{aligned} \tag{3.33}$$

Bien que cette équation offre une solution parfaite pour trouver $\vec{w}$, elle n'est pas couramment utilisée comme il peut être très difficile d'inverser une matrice de taille $N \times D$ quand N ou D est très grand.

3.3.2 Descente de gradient

La descente de gradient est un processus d'optimisation itératif. Le but est de minimiser une fonction paramétrée en prenant des pas dans la direction opposée du gradient de la fonction par rapport aux paramètres. Rappelons que le gradient d'une fonction multivariée est un vecteur indiquant la direction de croissance la plus forte. Dans le cas d'optimisation d'un modèle d'apprentissage automatique, le but est de minimiser la fonction de coût $\mathcal{L}$ en modifiant les paramètres du modèle θ. Il est donc nécessaire de calculer le gradient de $\mathcal{L}$ par rapport à θ : $\frac{\partial \mathcal{L}}{\partial \theta}$.

On peut ensuite itérativement modifier les poids de θ à l'étape t comme suit

$$\theta^{t+1} = \theta^t - \alpha \cdot \frac{\partial \mathcal{L}}{\partial \theta}. \tag{3.34}$$

où α est un hyperparamètre nommé le taux d'apprentissage qui contrôle la longueur de chaque pas.

Types de descente de gradient

Il existe trois types de descente de gradient. La descente de gradient classique calcule la moyenne du gradient sur l'ensemble de données complet à chaque pas d'optimisation. Cependant, quand l'ensemble de données est trop grand, il est très difficile de calculer tous les gradients en même temps. Une autre alternative est la descente de gradient stochastique qui effectue un pas d'optimisation pour chaque échantillon de données. Ce type de descente de gradient est cependant très susceptible au bruit ce qui peut ralentir le processus d'entraînement. Un compromis est la descente de gradient par lot. Cette alternative effectue une mise à jour des paramètres avec le gradient calculé sur un sous ensemble des données.

Ces trois types sont décrits par l'algorithme 3.1 en variant la taille de lot b (*batch size* en anglais). Si b est la taille de l'ensemble d'entraînement on retrouve la descente de gradient classique. Si b est 1, l'algorithme décrit la descente de gradient stochastique. Finalement, pour toute autre valeur de b on obtient la descente de gradient par lot. La taille du lot est un des principaux hyperparamètres à définir lors de l'entraînement.

Algorithm 3.1 Descente de gradient

input : $\mathcal{D}, f_\theta, \mathcal{L}, T$
output : θ
procedure Descente de gradient
    for $i = 1...T$ **do**
        $B \leftarrow b$ échantillons tirés de $\mathcal{D}$
        $l \leftarrow \frac{1}{N} \sum_{x,y \in B} \mathcal{L}(f_\theta(x), y)$
        $\theta \leftarrow \theta - \alpha \frac{\partial l}{\partial \theta}$
    end for
end procedure

Variantes de descente de gradient

L'algorithme de descente de gradient comporte certaines limitations. L'algorithme est notamment susceptible aux points de selle et aux minimums locaux. Dans les deux cas, le gradient est zéro. Bien qu'il soit très peu probable que le gradient soit nul dans toutes les directions quand l'espace de paramètres est très grand, ceci ralentit tout de même le processus d'optimisation [9]. De plus, l'optimisation par lots peut induire un certain bruit dans le processus d'optimisation puisque le gradient d'un lot n'est qu'une estimation du gradient de l'ensemble de données complet.

Afin de pallier ces problèmes, plusieurs solutions ont été proposées. La plus simple implique l'ajout d'un terme d'élan (*momentum* en anglais). On calcule une moyenne mobile des gradients qu'on appelle la vélocité, v. Au moment d'effectuer la mise à jour des paramètres, on utilise la moyenne mobile au lieu du gradient instantané. Ceci est exprimé mathématiquement comme suit

$$\begin{aligned} v_{t+1} &= \rho \cdot v_t + \frac{\partial \mathcal{L}}{\partial \theta} \\ \theta^{t+1} &= \theta^t - \alpha \cdot v_t. \end{aligned} \tag{3.35}$$

où ρ est le terme d'élan. Un exemple d'optimisation en une dimension avec et sans élan est illustré à la figure figure 3.2. L'élan permet à la fois de dépasser les minimums locaux et les points de selles, mais aussi d'éviter d'effectuer un pas d'optimisation dans la mauvaise direction en raison d'un lot bruité.

Dans les dernières années, plusieurs améliorations ont été apportées et de nouvelles techniques d'optimisation telles que RMSProp [50], Adagrad [8] et Adam [20] ont été proposées.

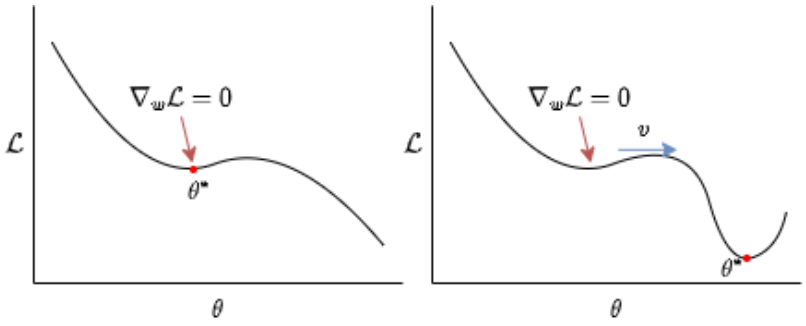


Figure 3.2 – Exemple de minimum local. À droite l'algorithme classique descente de gradient trouve la solution dans le minimum local. À gauche, l'algorithme de descente de gradient avec élan trouve la solution idéale puisque l'élan permet de dépasser le minimum local.

3.4 Réseaux de neurones

Jusqu'à présent, le modèle utilisé est un modèle linéaire. Ce modèle est cependant limité. Pour la classification, par exemple, le modèle linéaire ne peut que classifier des données qui sont linéairement séparables, c'est-à-dire qu'il est possible de correctement séparer les classes par un hyperplan dans l'espace des données. Un exemple de données linéairement séparables et non linéairement séparables est illustré dans la figure 3.3.

Une manière d'effectuer de la classification ou de la régression sur des données non linéaires est d'extraire des caractéristiques avec des fonctions de base. Des fonctions de base sont des fonctions qui prennent en entrée les données et qui les projettent dans un espace que l'on souhaite espère être linéaire. Cette technique est cependant limitée par le fait que la nature des fonctions doit être définie manuellement..

Les réseaux de neurones peuvent être vus comme des fonctions paramétriques permettant de trouver des caractéristiques automatiquement. Vaguement inspirés des neurones du cerveau humain, les réseaux de neurones sont composés de neurones organisés en couches qui transforment des entrées en caractéristiques intermédiaires.

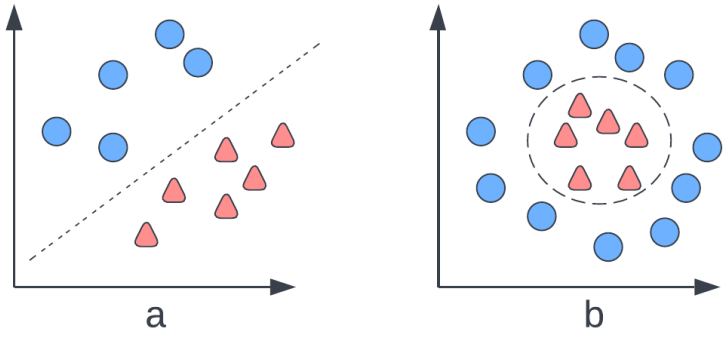


Figure 3.3 – Exemple de données linéairement séparables (a) et non linéairement séparables (b) en 2 dimensions.

Similairement aux fonctions de bases, ces couches, dites couches cachées, projettent les données dans un espace plus grand où elles sont linéairement séparables par une régression logistique par exemple.

À la base du réseau de neurones se trouve le neurone. Le neurone est une opération mathématique qui prend en entrée un vecteur et qui retourne une valeur d'activation. En pratique, ceci est un produit scalaire suivit d'une fonction non linéaire, dite fonction d'activation. La sortie d'un neurone h est donnée par

$$h = \phi(\vec{w}^T \vec{x} + b). \tag{3.36}$$

où ϕ est une fonction d'activation arbitraire. Jusqu'à présent seule la *sigmoïde* a été présentée, mais d'autres fonctions d'activation le seront à la sous-section 3.4.2. La forme de la *sigmoïde* offre une perspective intéressante du fonctionnement du neurone puisque la sortie du neurone est comprise entre 0 et 1 ce qui indique l'activation par rapport au vecteur d'entrée. Le degré d'activation est déterminé par les paramètres $\vec{w}$ et b. Lorsqu'une couche contient plusieurs neurones, on peut paralléliser les produits scalaires par une multiplication vectorielle

$$\vec{h} = \phi(\boldsymbol{W}\vec{x} + \vec{b}). \tag{3.37}$$

On nomme alors $\vec{h}$ la sortie de la couche ou un vecteur d'activation. Les tailles de

$\boldsymbol{W}$ et $\vec{b}$ seront déterminées par le nombre de neurones dans la couche et la taille de l'entrée. Pour une couche de M neurones ayant pour entrée un vecteur de taille D, la matrice W aura la forme $M \times D$ et le vecteur b aura une taille de M.

Si on superpose plusieurs couches de neurones, on retrouve l'architecture du Perceptron multicouches (MLP pour *Multilayer Perceptron* en anglais). Les couches utilisées pour composer le MLP sont nommées couches linéaires ou couches pleinement connectées. Le MLP permet donc de composer plusieurs opérations linéaires pour créer une fonction non linéaire.

La sortie d'une couche arbitraire $\vec{h}^l$ d'un MLP à L couches est donnée par

$$\vec{h}^l = \begin{cases} \phi(\boldsymbol{W}^l \vec{x} + \vec{b}^l) & \text{si } l = 1 \\ \boldsymbol{W}^l \vec{h}^{l-1} + \vec{b}^l & \text{si } l = L \\ \phi(\boldsymbol{W}^l \vec{h}^{l-1} + \vec{b}^l) & \text{sinon.} \end{cases} \tag{3.38}$$

où l est l'indice de la couche. On distingue les différentes couches du MLP comme étant la couche d'entrée ($l = 1$), les couches cachées ($l \in [2, ...L-1]$) et la couche de sortie ($l = L$). On note que la fonction d'activation de la couche de sortie n'est pas indiquée comme la tâche est arbitraire. Les paramètres du modèle sont

$$\theta = \{\boldsymbol{W}^1, \boldsymbol{W}^2, ..., \boldsymbol{W}^L, \vec{b}^1, \vec{b}^2, ...\vec{b}^L\}. \tag{3.39}$$

L'exemple d'un réseau à une couche cachée est donné à l'équation 3.40 (afin d'alléger les équations, on peut encore une fois utiliser la forme abréger de la couche linéaire $\boldsymbol{W}\vec{x}$ afin d'éviter l'utilisation du $\vec{b}$.) Ce réseau est illustré à la figure 3.4 avec une entrée de taille 3, une couche cachée de 6 neurones et une sortie de taille 1. La fonction du réseau prend la forme

$$f_\theta(\vec{x}) = \sigma(\boldsymbol{W}^2 \phi(\boldsymbol{W}^1 \vec{x})). \tag{3.40}$$

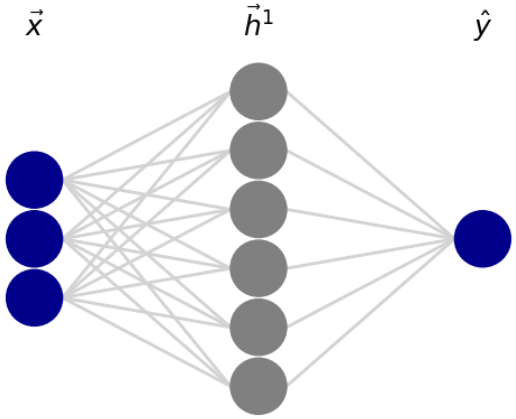


Figure 3.4 – Illustration d'un MLP à deux couches cachées. Illustration générée à l'aide de *Neural Network Visualizer (NNV)*[1].

3.4.1 Rétropropagation

Bien que les réseaux de neurones puissent représenter des fonctions arbitrairement complexes, le processus d'optimisation par descente de gradient est plus sophistiqué que pour les modèles linéaires. Alors que tous les paramètres du modèle linéaire sont directement impliqués dans le calcul de la sortie, ce n'est pas le cas pour le réseau de neurones qui comporte plusieurs couches. Il est donc nécessaire de trouver une manière de déterminer le gradient de toutes les couches du réseau.

L'algorithme de rétropropagation permet de régler ce problème. Cet algorithme utilise la règle de dérivation en chaîne en posant le réseau de neurones comme une composition de fonctions. La dérivation en chaîne d'une fonction composée $f(g(x))$ par rapport à x est définie comme suit

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)}\frac{dg(x)}{dx}. \tag{3.41}$$

La première étape de l'algorithme de rétropropagation est de définir un graphe computationnel. Ce graphe computationnel permet de définir la composition de fonctions du réseau de neurones. Chaque noeud de ce graphe est une opération et les feuilles sont des entrées ou des paramètres de la fonction. Dans le cas de la

1. `https://github.com/renatosc/nnv`

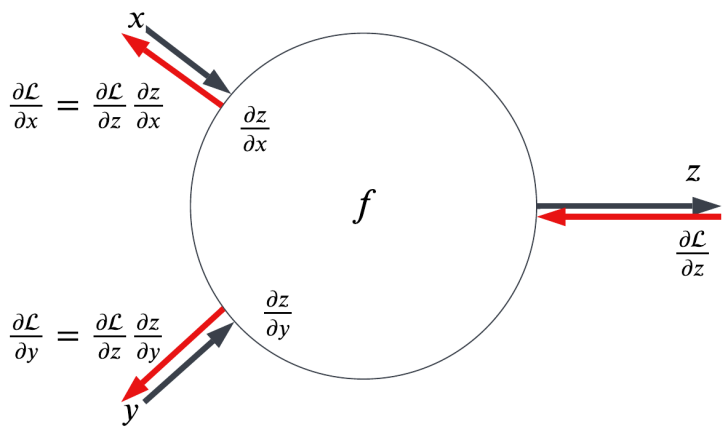


Figure 3.5 – Illustration de la rétropropagation pour un noeud avec des entrées x et y et une sortie z

.

rétropropagation d'un réseau de neurones, le dernier noeud est la fonction de coût.

Pour un noeud arbitraire ayant des entrées x et y, représenté par la fonction $f(x, y) = z$, on désire calculer le gradient de l'erreur par rapport à x et y. Si on connaît le gradient de l'erreur par rapport à z, $\frac{\partial \mathcal{L}}{\partial z}$, il s'agit de calculer le gradient de z par rapport à x et y (le gradient local) et d'appliquer la règle de la dérivation en chaîne. Comme la fonction f est plus simple que la fonction du réseau complet, déterminer le gradient est plus facile. On peut donc calculer le gradient de la fonction de coût par rapport à x,

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial x}. \tag{3.42}$$

La même opération peut être effectuée pour trouver $\frac{\partial \mathcal{L}}{\partial y}$. Cette dérivation est illustrée à la figure. 3.5

En pratique, on définie la propagation avant et la propagation arrière (rétropropagation) du réseau de neurones en définissant les opérations intermédiaires afin de définir les noeuds. Dans ce cas, chaque couche est un noeud. Notons cependant que contrairement à la notation utilisée à l'équation 3.38, on sépare la multiplication matricielle et la fonction d'activation en deux noeuds. Si on reprend en exemple le

réseau de neurones $f_\theta(x) = \hat{y}$ exprimé à l'équation 3.40, la propagation avant est donnée par

$$\begin{aligned}
\vec{z}^1 &= \boldsymbol{W}^1 \vec{x} \\
\vec{h}^1 &= \phi(\vec{z}^1) \\
\vec{z}^2 &= \boldsymbol{W}^2 \vec{h}^1 \\
\hat{y} &= \sigma(\vec{z}^2).
\end{aligned}$$

La propagation arrière est donnée par

$$\frac{\partial \mathcal{L}}{\partial \vec{z}^2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \vec{z}^2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \sigma'(\vec{z}^2)$$

$$\frac{\partial \mathcal{L}}{\partial \vec{h}^1} = \frac{\partial \mathcal{L}}{\partial \vec{z}^2} \frac{\partial \vec{z}^2}{\partial \vec{h}^1}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^2} = \frac{\partial \mathcal{L}}{\partial \vec{z}^2} \frac{\partial \vec{z}^2}{\partial \boldsymbol{W}^2}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{z}^1} = \frac{\partial \mathcal{L}}{\partial \vec{h}^1} \frac{\partial \vec{h}^1}{\partial \vec{z}^1} = \frac{\partial \mathcal{L}}{\partial \vec{h}^1} \phi'(\vec{z}^1)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^1} = \frac{\partial \mathcal{L}}{\partial \vec{z}^1} \frac{\partial \vec{z}^1}{\partial \boldsymbol{W}^1}.$$

Pour ce qui est de $\frac{\partial \mathcal{L}}{\partial \hat{y}}$, la dérivée de la fonction de coût $\mathcal{L}$ par rapport à la sortie du réseau y, son contenu dépendra de la fonction de coût utilisée. Par exemple, dans le cas de l'erreur quadratique moyenne, le gradient est défini comme suit

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{y}} &= \frac{\partial}{\partial \hat{y}} \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \\
&= \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \hat{y}} (\hat{y}_i - y_i)^2 \\
&= \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i) \frac{\partial}{\partial \hat{y}} (\hat{y}_i - y_i) \\
&= \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i).
\end{aligned} \tag{3.43}$$

Il est ensuite nécessaire de définir les gradients locaux. Ceci dépendra de l'opération effectuée à la couche en question. Dans le réseau donné en exemple, il y a deux types des couches : les couches linéaires et les fonctions d'activation. Des exemples de fonctions d'activation ainsi que leurs dérivées seront présentés à la section 3.4.2. Le gradient local de la couche linéaire est défini en fonction de l'opération effectuée. Comme l'opération est une multiplication matricielle, il est nécessaire de déterminer le gradient local par rapport à l'entrée et à la matrice de paramètres. Ceci est démontré dans la section suivante.

Rétropropagation de multiplication matricielle

La rétropropagation de l'opération de la multiplication matricielle est l'une des opérations de base pour la rétropropagation du MLP. Contrairement à la rétropropagation des fonctions d'activation, ce noeud comprend une entrée et des paramètres. Il est nécessaire de calculer le gradient par rapport aux paramètres afin de réaliser l'optimisation et par rapport à l'entrée afin de propager le gradient aux couches précédentes.

Considérons l'opération d'une couche d'un MLP avec une entrée $\vec{x} \in R^D$, une matrice de paramètres $\boldsymbol{W} \in R^{M \times D}$ et l'opération $\vec{z} = \boldsymbol{W}\vec{x} \in R^M$ dont chaque élément est défini par

$$\vec{z}_i = \sum_{j=1}^{D} \boldsymbol{W}_{ij} \cdot \vec{x}_j. \tag{3.44}$$

En supposant que $\frac{\partial \mathcal{L}}{\partial \vec{z}}$ est connu, il est possible de trouver

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}}{\partial \vec{z}} \frac{\partial \vec{z}}{\partial \boldsymbol{W}} \tag{3.45}$$

où $\frac{\partial \vec{z}}{\partial \boldsymbol{W}}$ est une matrice Jacobienne.

Définition 2: Matrice Jacobienne

La matrice jacobienne est une matrice composée de dérivées partielles. Supposons une fonction $f(\vec{x}) : R^n \to R^m$. La matrice jacobienne, J, de f est un matrice de taille $m \times n$ où chaque élément est définit $J_{ij} = \frac{\partial f_i}{\partial x_j}$. C'est a dire que chaque élément de J indique la dérivée partielle d'un élément de f par rapport à un élément de x.

Cependant, dans le cas de la multiplication matricielle comme les entrées ont respectivement des tailles $M \times D$ et $D \times 1$, la matrice jacobienne aura $M \times M \times D$ éléments. Ceci est beaucoup trop gros pour être sauvegardé en mémoire. On remarque cependant que la majorité de $\frac{\partial \vec{z}}{\partial \boldsymbol{W}}$ est composée de 0. Il serait donc possible de simplifier l'opération. Pour un élément de $\boldsymbol{W}$ et un élément de $\vec{z}$

$$\frac{\partial \vec{z}_i}{\partial \boldsymbol{W}_{mn}} = \begin{cases} \vec{x}_n & \text{si } i = m \\ 0 & \text{sinon.} \end{cases} \tag{3.46}$$

Si on examine la dérivée partielle de $\mathcal{L}$ par rapport à un élément de $\boldsymbol{W}$: $\boldsymbol{W}_{mn}$, on obtient

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{mn}} = \sum_{i=1}^{D} \frac{\partial \mathcal{L}}{\partial \vec{z}_i} \frac{\partial \vec{z}_i}{\partial \boldsymbol{W}_{mn}}. \tag{3.47}$$

On remarque que seulement une valeur dans la somme n'est pas nulle quand $m = i$. Donc,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{mn}} = \sum_{i=1}^{D} \frac{\partial \mathcal{L}}{\partial \vec{z}_m} \cdot \vec{x}_n. \tag{3.48}$$

Cette opération se simplifie par un produit matriciel pour trouver chaque element de $\boldsymbol{W}$ est donné par

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}}{\partial \vec{z}} \vec{x}^T. \tag{3.49}$$

Une procédure similaire peut être développée pour constater que

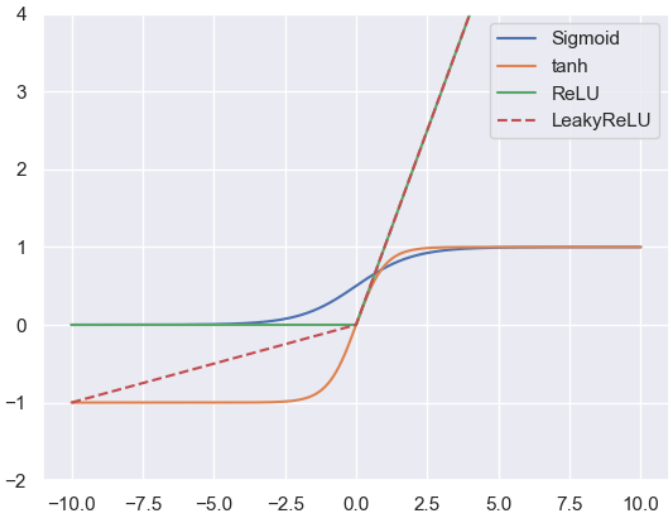


Figure 3.6 – Illustration de différentes fonctions d'activation

$$\frac{\partial \mathcal{L}}{\partial \vec{x}} = \boldsymbol{W}^T \frac{\partial \mathcal{L}}{\partial \vec{z}}. \tag{3.50}$$

3.4.2 Fonctions d'activation

Tel que mentionné plus haut, il est essentiel de séparer les opérations linéaires du réseau de neurones par des fonctions d'activation non linéaires. Sans ces dernières, l'ensemble de multiplications matricielles de chaque couche pourrait être réduit à une seule opération. De plus, la fonction d'activation doit être différentiable afin de permettre la rétropropagation du gradient. Une fonction d'activation déjà présentée est la *sigmoïde*. Cependant, son utilisation comme fonction d'activation pour les couches cachées n'est pas optimale. D'autres alternatives sont donc utilisées. Certaines d'entre elles seront décrites ci-dessous et illustrées à la figure 3.6.

Sigmoïde

La fonction *sigmoïde* telle que présentée précédemment prend la forme suivante

$$\sigma(z) = \frac{1}{1 - e^{-z}}. \tag{3.51}$$

La dérivée de cette fonction est donnée par

$$\sigma^{'}(z) = \sigma(z)(1 - \sigma(z)). \tag{3.52}$$

Si on observe la forme de la sigmoïde, on remarque le gradient est presque nul pour des valeurs positives et négatives qui s'éloignent de zéro. Ceci nuit à l'entraînement du réseau puisque des petites valeurs de gradient ne permettent pas au gradient de se propager vers les premières couches puisqu'il devient de plus en plus petit. D'autre part, il est préférable que la sortie d'une fonction d'activation soit centrée sur 0 puisque l'optimisation sera plus rapide. L'optimisation est plus lente pour une fonction d'activation non centrée sur zéro puisque les valeurs en sortie (et donc les gradients) seront toujours entièrement positives ou négatives. Il faut donc plus d'itérations lors de l'optimisation puisque le chemin parcouru lors de l'optimisation prendre la forme d'un *zigzag* au lieu d'une ligne droite.

Tangente hyperbolique

Une fonction d'activation qui est centrée à zéro est la tangente hyperbolique (*tanh*). Cette fonction et son gradient sont définis comme suit

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.53}$$

et

$$tanh^{'}(z) = 1 - tanh^{2}(z). \tag{3.54}$$

Cette fonction est centrée sur zéro, mais comme la *sigmoïde*, la dérivée est presque nulle pour des valeurs qui s'éloignent de zéro.

ReLU

Afin de régler le problème de perte de gradient, la fonction ReLU est couramment utilisée [25]. La fonction et son gradient sont définis comme suit

$$ReLU(z) = \begin{cases} 0 & \text{si } x \leq 0 \\ z & \text{sinon} \end{cases} \tag{3.55}$$

et

$$ReLU^{'}(z) = \begin{cases} 0 & \text{si } x \leq 0 \\ 1 & \text{sinon.} \end{cases} \tag{3.56}$$

Cette fonction est non seulement simple à implémenter, mais la dérivée n'est jamais 0 pour des valeurs positives. Bien que cette fonction soit très utilisée, le dérivée est nul pour des valeurs négatives et elle n'est pas centrée sur 0. Des variantes de la fonction ReLU ont donc été proposées. Par exemple, la fonction $LeakyReLU$ est plus centrée sur 0 et à un gradient non nul sur tout son domaine d'entrée [31]. La fonction et son gradient sont définies par :

$$LReLU(z) = \begin{cases} cz & \text{si } x \leq 0 \\ z & \text{sinon} \end{cases} \tag{3.57}$$

et

$$LReLU^{'}(z) = \begin{cases} c & \text{si } x \leq 0 \\ 1 & \text{sinon.} \end{cases} \tag{3.58}$$

3.5 Sous et sur apprentissage

Avec la possibilité de rajouter un nombre arbitraire de couches et donc un nombre arbitraire de paramètres, une attention particulière doit être apportée pour éviter le sur-apprentissage. Le sur-apprentissage se produit quand le modèle apprend une correspondance trop spécifique des données d'entraînement. Le modèle généralise donc

mal à de nouvelles données telles que celles dans les données de test. Ceci est souvent causé par un modèle avec un trop grand nombre de paramètres.. Inversement, le sous-apprentissage se produit lorsque le modèle n'est pas assez complexe et qu'il ne réussit pas à adéquatement trouver la correspondance entre les entrées et les sorties.

Le surapprentissage peut être détecté à l'aide de l'ensemble de validation. Lors du sur-apprentissage, on remarque que le coût de validation augmente alors que le coût d'entraînement continue à descendre tel qu'illustré à la figure 3.7. La meilleure façon de réduire le sur-apprentissage est d'augmenter le nombre de données dans l'ensemble d'entraînement, ce qui est difficile, surtout en imagerie médicale.. Une autre méthode utilisée est la régularisation déjà introduite dans le contexte de maximum *a posteriori*. La régularisation pénalise les paramètres qui s'éloigne de zéro forçant le modèle à rester simple. Une autre alternative est l'ajout de décrochage (*Dropout* en anglais).

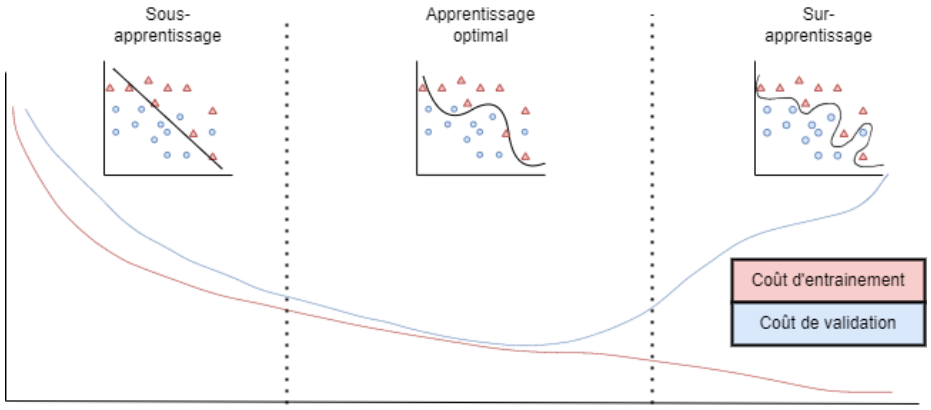


Figure 3.7 – Illustration de sous et sur apprentissage

3.5.1 Décrochage

Le décrochage, ou *Dropout*, est un opération appliquée sur les réseaux de neurones pour réduire le sur-apprentissage en forçant le réseau à apprendre des caractéristiques plus générales au lieu d'une correspondance un pour un entre les données et les cibles d'entraînement [47]. Le décrochage est réalisé lors de la phase d'entraînement alors que chaque neurone a une probabilité p d'être mis à zéro. Lors de chaque propagation avant, plusieurs neurones sont désactivés ce qui force le réseau à apprendre différents

"chemins". Lors de la phase de test, tous les neurones sont activés. Afin de compenser les neurones supplémentaires qui augmentent la valeur des activations, on multiplie les activations par p afin d'avoir des valeurs similaires à celles obtenues en entraînement.

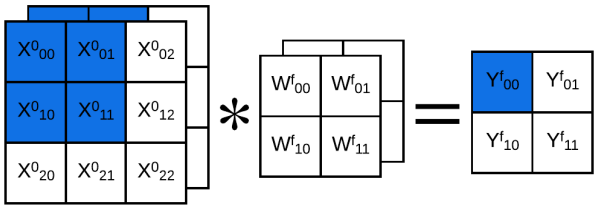
3.6 Réseaux convolutifs

Bien que les MLP soient très performants sur certaines tâches, ils souffrent tout de mÊme de certains inconvénients. Par exemple, la première couche pleinement connectée devient particulièrement grande lorsque les vecteurs d'entrée sont grands. Ceci est fréquent pour les images à résolution élevée. Pour cette raison d'autres types de couches peuvent être utilisées afin d'extraire des caractéristiques. images à résolution élevée. La solution est de réduire le nombre de paramètres indépendants dans le réseau. Ceci peut être réalisé avec une opération de convolution. Cette opération, très utilisée en traitement d'image, permet d'extraire des cartes d'activations avec un très petit nombre de paramètres tout en préservant la structure spatiale de l'image.

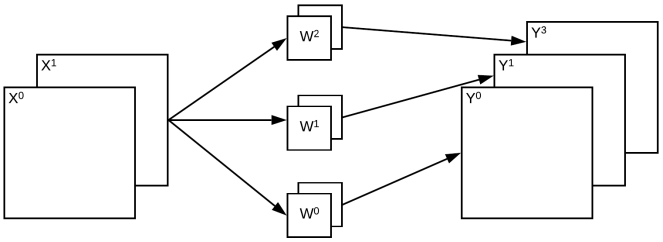
3.6.1 Convolution

La convolution est une opération qui implique une succession de produits scalaires et de translations. La convolution se fait en glissant un filtre de gauche à droite et de haut en bas dans une image. À chaque position, un produit scalaire est effectué entre les paramètres du filtre et les pixels de l'image correspondant à la position du filtre. Le résultat de ce produit scalaire devient un pixel contenu dans ce qu'on appelle une carte d'activation. Cette opération est illustrée à la figure 3.8a. Lors de la translation du filtre vers la droite et vers le bas, le filtre peut être déplacé d'un ou de plusieurs pixels. Le nombre de pixels de translation se nomme la foulée, s pour *stride* en anglais). Afin de préserver les dimensions de l'image, celle-ci peut être rembourrée d'un nombre p de pixels autour de l'image originale.

La couche convolutive utilisée dans les réseaux convolutifs peut se faire à l'aide de plusieurs filtres, dénombré par F. Chaque filtre a une largeur et hauteur, K_h et K_w respectivement, et doit avoir le même nombre de canaux que l'image d'entrée. Chaque convolution entre une image et un filtre crée un canal dans l'image de sortie. Ceci est



(a) Illustration du calcul d'un canal f de Y



(b) Illustration de la convolution d'une image à deux canaux et trois filtres

Figure 3.8 – Illustration de la convolution 2D

illustré à la figure 3.8b

Exprimé de manière mathématique la convolution est définie comme suit. Pour une image en entrée de taille $H \times W$ ayant C canaux, $\boldsymbol{X} \in R^{C \times H \times W}$ et un filtre $\boldsymbol{W} \in R^{F \times C \times K_h \times K_w}$ et un biais $\vec{b} \in R^F$, la sortie est une nouvelle image

$$\boldsymbol{Y} = \boldsymbol{W} * \boldsymbol{X} \in R^{F \times H_2 \times W_2} \tag{3.59}$$

où $H_2 = \frac{H+2p-k_h}{s} + 1$ et $W_2 = \frac{W+2p-k_w}{s} + 1$. La valeur de chaque pixel est donnée par l'expression suivante

$$\boldsymbol{Y}_{fij} = \sum_{m=0}^{k_h} \sum_{n=0}^{k_w} \boldsymbol{X}_{i+m,j+n} \cdot \boldsymbol{W}_{fmn} + \vec{b}_f. \tag{3.60}$$

3.6.2 Architecture standard du réseau convolutif

Similairement aux réseaux linaires multicouches, l'architecture d'un réseau convolutif implique la combinaison de plusieurs couches cachées. Le but étant toujours de

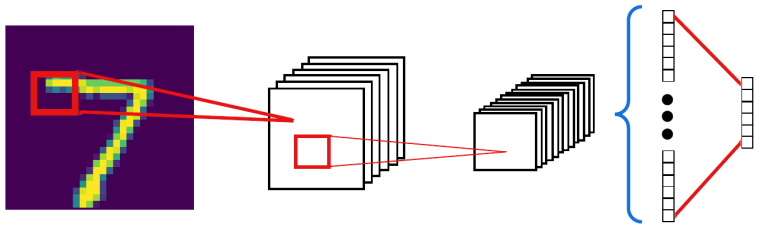


Figure 3.9 – Illustration d'un réseau convolutif simple appliqué sur une image de la base de données MNIST [27]. Les liens rouges indiquent des couches avec paramètres (convolutives et linéaires) et les liens bleus indiquent des couches sans paramètres (aplatissement).

projeter les données vers un espace où la classification (ou la régression) linéaire est possibles. Par exemple, un réseaux convolutif à trois couches est exprimé comme suit

$$f_\theta(\boldsymbol{X}) = \boldsymbol{W}^3 F(\phi(\boldsymbol{W}^2 * \phi(\boldsymbol{W}^1 * \boldsymbol{X}))). \tag{3.61}$$

Ici, $\boldsymbol{W}^2$ et $\boldsymbol{W}^1$ sont les paramètres de deux couches convolutives, $\phi(\cdot)$ est une fonction d'activation arbitraire, $F(\cdot)$ est une couche d'aplatissement (*Flatten*) et $\boldsymbol{W}^3$ est la matrice de paramètres d'une couche linéaire de sortie. Ce réseau est illustré à la figure 3.9

Plus généralement, la structure classique du réseau convolutif est réalisée en empilant des couches convolutives séparées par des fonctions d'activation. Les cartes d'activation sont ensuite reformées en un seul vecteur avant de passer par une ou plusieurs couches linéaires. En général, la taille des cartes d'activation est réduite et le nombre de canaux est augmenté à chaque couche convolutive.

Ce type d'architecture a été peaufiné aux fil des années. Le premier réseau convolutif fut LeNet publié en 1998 [27]. Ce réseau a été utilisé pour la classification de caractères. En 2012, la révolution de l'apprentissage profond est déclenchée par la publication de AlexNet [25]. Ce réseau a drastiquement réduit le taux d'erreur pour gagner le concours de classification *Imagenet* [7]. Ce réseau a été le premier à exploiter les cartes graphiques pour paralléliser l'entraînement d'un réseau. Dans les années suivantes, les réseaux sont devenus de plus en plus profonds. Le réseau VGG par exemple utilise 16 ou 19 couches contrairement à huit pour AlexNet [48]. Le modèle VGG a notamment

réussi à augmenter le nombre de couches en réduisant la taille des filtres.

Échantillonnage (*pooling*)

Une couche qui accompagne souvent la convolution, notamment dans les réseaux énumérés précédemment et la couche d'échantillonnage (*Pooling* en anglais). Cette couche permet de réduire la taille des images ou cartes d'activations et donc de réduire le nombre de paramètres du réseau. L'opération d'échantillonnage utilise aussi un filtre de taille $K_h \times K_W$ et une foulée de s pixels. L'échantillonnage se fait en appliquant une opération de réduction sur la région de l'image couverte par le filtre et en effectuant une translation de s pixels comme pour la convolution. L'opération de réduction peut prendre plusieurs formes. Deux opérations fréquemment utilisées sont le maximum des valeurs (*Max-Pooling*) et la moyenne (*Average-Pooling*). Un exemple du *Max-Pooling* est illustré à la figure 3.10

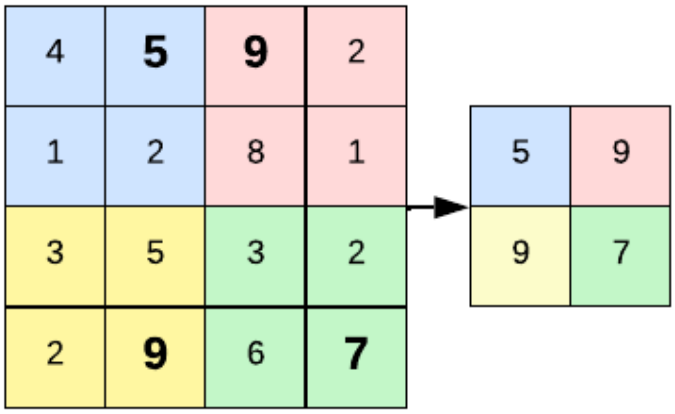


Figure 3.10 – Illustration de l'opération max-pooling avec un filtre 2×2 et une foulée de 2.

3.6.3 Architecture moderne du réseau convolutif

Contrairement au MLP les réseaux convolutifs permettent d'avoir des réseaux très profonds avec un nombre relativement faible de paramètres (nombre relatif à la

capacité de calcul qui augmente sans arrêt). Cependant quand les réseaux deviennent trop profonds, le gradient devient très faible lors de la rétropropagation puisque cette opération implique la multiplication successive de plusieurs valeurs inférieures à un. Ce problème se nomme le problème de la disparition du gradient (*vanishing gradient problem* en anglais). Afin de régler ce problème, les réseaux neuronaux résiduels (*Resnet*) ont été proposés [17]. Ces réseaux sont composés de blocs dits résiduels qui additionnent leur entrée à leur sortie. Ce passage en parallèle, nommé connexion résiduelle, permet, permet à la fois au réseau d'apprendre de meilleurs caractéristiques et au gradient de mieux se propager vers l'entrée du réseau.

Un autre type de réseau similaire est le réseau dense (*Densenet*) qui comporte aussi des connexions parallèles, mais qui se concatènent à la sortie au lieu d'être additionnées [15]. La concaténation se fait par la dimension des canaux. La sortie a donc des canaux qui proviennent de plusieurs résolutions du réseau. Ceci permet aussi au réseau de mieux propager le gradient, mais aussi de combiner et d'opérer sur de l'information provenant de différentes résolutions. Les blocs résiduels et denses sont illustrés à la figure 3.11.

3.7 Segmentation

Bien que la classification et la régression permettent de réaliser certaines tâches, l'imagerie médicale demande souvent des tâches plus complexes. Il est souvent nécessaire d'identifier et de localiser des structures complexes dans les images médicales. Pour cette raison, la tâche de segmentation est utilisée.

La segmentation est une extension de la tâche de classification où il est nécessaire de classifier chaque pixel dans K classes prédéfinies. La base de données est composée d'images $\boldsymbol{X} \in R^{C \times H \times W}$ et de cartes de segmentation $\boldsymbol{Y} \in R^{K \times H \times W}$. Chaque élément spatial de $\boldsymbol{Y}$ est un vecteur *one-hot* indiquant la classe du pixel dans l'image. Cette tâche demande une modification aux architectures présentées jusqu'à présent ainsi que de nouvelles couches.

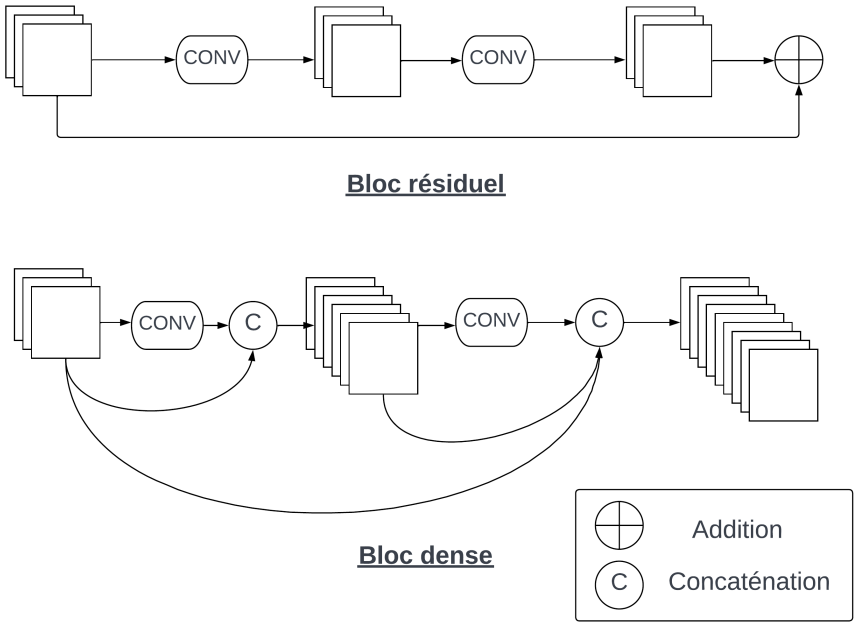


Figure 3.11 – Illustration de blocs résiduels et denses

3.7.1 Méthodes

La méthode la plus simple pour effectuer la segmentation est de séparer l'image en sous-régions et de classifier le pixel central [52]. Ceci est cependant très lent, car il requiert un nombre important d'opérations. De plus, le réseau ne peut qu'utiliser l'information locale contenue dans la sous-région et non l'information globale contenue dans le reste de l'image.

Long *et al.* ont proposé un réseau pleinement convolutif afin d'effectuer la segmentation d'une image avec une seule propagation avant [29]. Ils ont utilisé des réseaux convolutifs existants, tels que AlexNet et VGG, et ont remplacé les couches pleinement connectées par des convolutions afin d'avoir une sortie avec un nombre de canaux égal au nombre de classes. Cette méthode est cependant limitée puisque les couches d'échantillonnage réduisent la taille de l'image. Il est aussi possible d'utiliser une architecture entièrement convolutive tout en gardant des cartes d'activation de la même taille que l'image originale en ajustant le rembourrage aux opérations de convolutions. Cependant, ceci est très coûteux en termes de calculs.

Comme la carte de segmentation contient moins d'information que l'image d'entrée puisque plusieurs pixels ont la même valeur, il est possible de réaliser la segmentation sans devoir garder la résolution spatiale complète tout au long du réseau. Dans cette optique et afin de réduire la quantité de calcul, les architectures encodeur-décodeur ont été proposées. Dans cette architecture un réseau encodeur, ayant une forme similaire aux réseaux de convolutions introduits précédemment, transforme l'image d'entrée en une représentation plus petite spatialement, mais ayant plus de canaux. Les opérations de convolutions avec une foulée de plus de un et les opérations d'échantillonnage sont utilisées pour réduire la résolution spatiale tout en augmentant le nombre de canaux. Le décodeur prend en entrée la représentation et la transforme en carte de segmentation par une succession de couches de suréchantillonnage afin d'augmenter la résolution. Les couches de suréchantillonnage seront expliquées à la section 3.7.1

Noh *et al.* ont proposé une architecture basée sur le modèle VGG utilisé comme encodeur et décodeur composé du modèle VGG inversé avec des couches de suréchantillonnage au lieu de couche d'échantillonnage [34]. Le réseau SegNet a été proposé ayant une architecture similaire, mais en ajoutant un lien entre les couches correspondantes d'échantillonnage et de suréchantillonnage nommées *Max-unpooling* [3] (voir prochaine section).

Finalement, le modèle U-Net a été présenté et est depuis un des modèles les plus populaires en segmentation [42]. Le modèle utilise des couches denses entre les blocs d'encodeur et décodeur afin de propager l'information de haute résolution au décodeur ce qui permet une meilleure segmentation des détails. Depuis, plusieurs variations ont été proposées telles que le V-Net pour la segmentation en trois dimensions [33] et le Enet, une version plus petite et efficace du UNet [38].

Suréchantillonnage

Les couches de suréchantillonnage ont pour but d'augmenter la résolution spatiale de l'image dans le décodeur du réseau de segmentation. Plusieurs méthodes peuvent être utilisées telles que le *un-pooling*, le suréchantillonnage bilinéaire et la convolution transposée.

Le *un-pooling* est la méthode la plus simple pour augmenter la résolution. Cette opération effectue le contraire de la couche *pooling*. Il existe cependant quelques

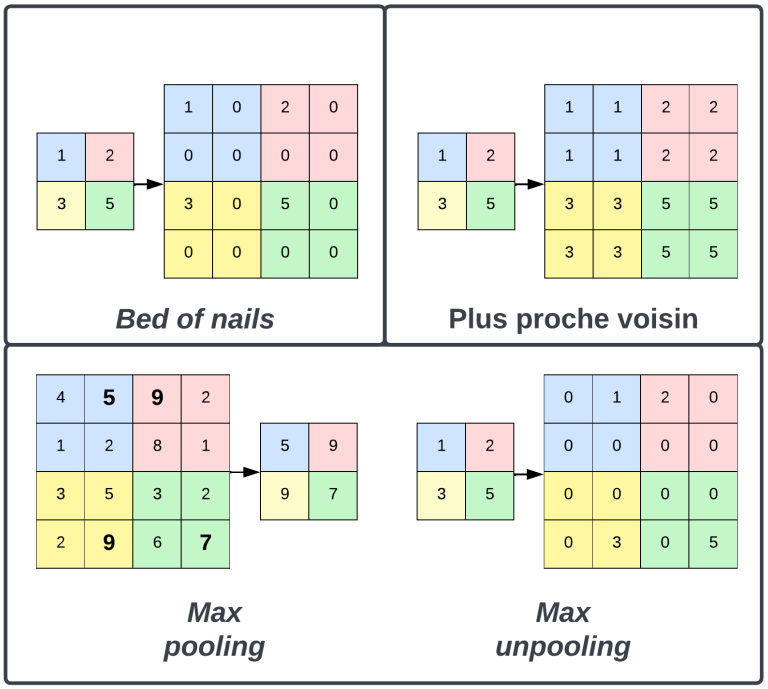


Figure 3.12 – Illustration des différentes opérations de *unpooling*

variantes. Premièrement, la méthode du «plus proche voisin» duplique le pixel d'une carte d'activation plus petite dans une sous-région de la carte de sortie en fonction de la foulée et de la taille du filtre. La méthode *bed of nails* ajoute seulement la valeur une fois dans la région dans une position prédéterminée et remplit les autres pixels par une constante, souvent zéro. Finalement, le *max unpooling* utilise l'information de couche *pooling* de l'encodeur pour placer les éléments dans la carte de sortie [3]. Ces trois variantes sont illustrées à la figure 3.12.

Le suréchantillonnage bilinéaire utilise une interpolation bilinéaire, soit une interpolation linéaire dans les dimensions de hauteur et largeur, pour augmenter la résolution.

Finalement, la convolution transposée modifie la convolution standard afin d'avoir une carte d'activation plus grande. L'avantage de la convolution transposée est que le suréchantillonnage peut être appris lors de la rétropropagation. Un exemple de convolution transposée est illustré à la figure 3.13.

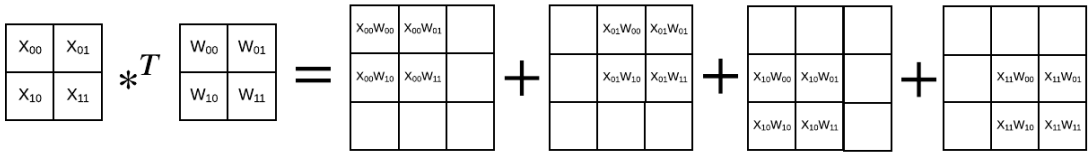


Figure 3.13 – Illustration de l'opération de convolution transposée

Chapitre 4

Estimation d'incertitude

Les réseaux de neurones présentés dans les derniers chapitres atteignent des performances en moyenne aussi bonne que les experts humains sur de nombreuses tâches. Notamment en segmentation cardiaque, des travaux récents ont démontré que les réseaux de neurones peuvent atteindre la variabilité inter-experts pour la segmentation d'IRM [4] et d'ultrason [30]. Atteindre la variabilité inter-experts signifie que l'erreur de prédiction du réseau de neurones est en moyenne plus faible que la différence de segmentation entre les experts. Cependant, malgré ces performances globalement intéressantes, les réseaux de neurones produisent des erreurs ponctuelles. Bien qu'il soit très difficile, voire impossible, d'éliminer ces erreurs avec les algorithmes actuels, il serait bien de pouvoir identifier les cas d'erreur automatiquement. Pour cette raison, les techniques d'estimation d'incertitude sont d'une grande importance.

Au courant de ce chapitre, les termes **incertitude** et **confiance** seront utilisés. Dans certains cas, il est préférable de parler de confiance alors qu'il est parfois plus clair d'exprimer l'incertitude. Pour ce mémoire, ces deux termes seront considérés des inverses. De plus, si les deux sont bornés entre zéro et un, on peut utiliser l'égalité suivante $c = 1 - i$, où c est la confiance et i est l'incertitude.

4.1 Évaluation de l'incertitude

Avant d'explorer les différentes techniques d'estimation d'incertitude, il est judicieux de définir la méthode utilisée pour évaluer la qualité de l'estimation. Le but premier de l'estimation d'incertitude est d'identifier les erreurs effectuées par l'algorithme. La quantification de l'erreur ainsi que la prédiction de probabilité vont dépendre de la tâche. Dans le cas de la classification, on peut prédire une valeur de confiance que la prédiction est bonne. Si cette valeur de confiance est une probabilité, il est possible de quantifier la calibration du modèle, soit une measure du degré à laquelle la confiance correspond à la probabilité d'avoir une bonne prédiction. Dans le cas de la régression, on ne peut pas déterminer si une prédiction est bonne ou non. On peut cependant donner une mesure de distance entre la valeur prédite et la valeur réelle. De plus, on ne peut pas prédire la probabilité d'avoir une bonne prédiction. On peut cependant estimer l'erreur de la prédiction. Puisqu'une grande valeur d'erreur estimée devrait indiquer une plus grande erreur, on parle alors d'incertitude au lieu de confiance.

Classification. Considérons un algorithme de classification arbitraire et une méthode qui évalue la confiance (une valeur de probabilité). L'algorithme peut s'agir d'un modèle qui retourne la prédiction et la confiance ou deux modèles distincts.

On dit qu'un algorithme de classification est calibré si la probabilité prédite, $\hat{p}$, correspond à la probabilité réelle que la prédiction soit bonne. Par exemple, pour 100 prédictions ayant une probabilité de 0.5, 50 d'entre elles devraient être bien classifiées. Mathématiquement, ceci est exprimé comme suit [13]

$$P(\hat{y} = y|\hat{p} = p) = p, \quad \forall p \in [0, 1] \tag{4.1}$$

où $\hat{y}$ est la prédiction et y est la vérité terrain. La calibration d'un modèle peut être visualisée par un diagramme de fiabilité (*reliability diagram* en anglais) [35]. Ce diagramme est construit à partir de N prédictions classées en M groupes.

L'appartenance à un groupe B_m est déterminée par la probabilité $\hat{p}$ et l'intervalle du groupe $I_m =]\frac{m-1}{M}, \frac{m}{M}]$. Pour chaque groupe, on calcule la justesse (*accuracy* en Anglais) et la confiance moyenne

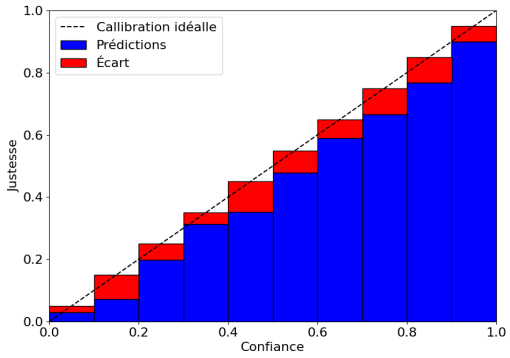


Figure 4.1 – Exemple d'un diagramme de fiabilité. La hauteur des barres bleues indique la justesse du groupe et la barre rouge indique l'écart avec la calibration parfaite.

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(y_i = \hat{y}_i), \tag{4.2}$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i. \tag{4.3}$$

Un exemple de diagramme de fiabilité est montré à la figure 4.1.

L'erreur de calibration (ECE pour *expected calibration error* en anglais) est une valeur qui représente la calibration du modèle sur l'ensemble des prédictions. Il s'agit de l'espérance mathématique de la différence entre la justesse et la confiance, soit

$$\text{ECE} = \mathbb{E}_{\hat{p}}[P(\hat{y} = y | \hat{p} = p) - p]. \tag{4.4}$$

Puisqu'il y a un nombre fini M de groupes, on calcule l'erreur de calibration comme suit

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|. \tag{4.5}$$

Régression. L'erreur de calibration dans le cas de la régression donc calculée en prenant la différence entre l'erreur et l'incertitude moyenne de chaque groupe

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{err}(B_m) - \text{uncert}(B_m)|. \tag{4.6}$$

L'erreur et l'incertitude moyenne peuvent être calculées de différentes manières. On doit cependant s'assurer d'avoir les mêmes unités. Par exemple, on peut utiliser l'erreur quadratique moyenne et la variance de la prédiction comme suit

$$\text{err}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} (y_i - \hat{y}_i)^2, \tag{4.7}$$

$$\text{uncert}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{\sigma}^2. \tag{4.8}$$

Il est aussi possible d'utiliser l'erreur absolue, mais il faudrait alors utiliser l'écart-type comme valeur d'incertitude.

4.2 Approche probabiliste

Rappelons qu'un réseau de neurones qui effectue de la classification multi-classes produit un vecteur de *logits* $\vec{z} \in R^K = f_\theta(x)$ qui est ensuite passé dans la fonction *softmax* afin d'avoir un vecteur de probabilités $\vec{p}$. Ce réseau exprime la probabilité conditionnelle de chaque classe en fonction de l'image d'entrée. Une manière très simple d'estimer la confiance de la prédiction est de prendre la valeur de probabilité associée à la classe prédite, soit celle avec la plus grande probabilité

$$c = \hat{p} = \max(\vec{p}). \tag{4.9}$$

Plus cette probabilité se rapproche de 1, plus le réseau est confiant de sa prédiction. Inversement, si la probabilité se rapproche de $1/K$, plus le réseau est incertain puisque plusieurs probabilités sont grandes.

Guo *et al.* ont cependant démontré que les modèles de classification profonds sont mal calibrés même si leurs performances en terme de justesse sont bonnes [13]. Ceci

est causé par le surapprentissage sur la fonction de coût d'entropie croisée. Bien que la justesse ne change pas puisque la classe ayant la plus grande probabilité reste identique, la valeur de cette probabilité augmente lors du surapprentissage. Ceci fait en sorte que lorsque vient le temps de généraliser à de nouvelles données, les réseaux sont systématiquement surconfiants.

Afin de régler ce problème, Guo *et al.* ont proposé d'ajouter une variable de température lors du calcul du *softmax*. Cette valeur, τ, est optimisée sur l'ensemble de validation après l'entraînement du réseau. Avec cette variable, la fonction *softmax* prend la forme suivante

$$\text{softmax}(\vec{z})_i = \frac{e^{\vec{z}_i/\tau}}{\sum_{j=1}^{K} e^{\vec{z}_j/\tau}}. \tag{4.10}$$

On remarque que la valeur de τ ne changera pas la prédiction, mais peut réduire la valeur de la confiance si une grande valeur est apprise. La différence entre la fonction softmax avec et sans température est illustrée à la figure 4.2.

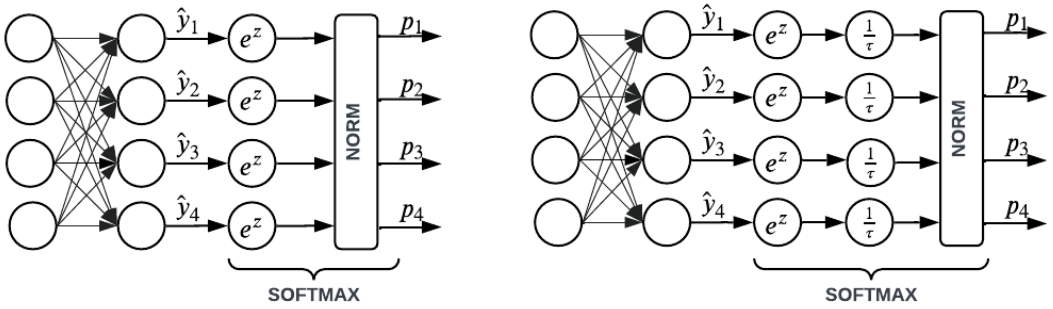


Figure 4.2 – À gauche, la représentation graphique de la fonction Softmax. À droite, la représentation graphique de la fonction Softmax avec le paramètre de température appris, τ

4.3 Réseau de neurones bayésien

Kendal *et al.* ont présenté un formalisme plus sophistiqué pour quantifier l'incertitude présente dans les réseaux de neurones. Ils ont séparé l'incertitude en deux catégo-

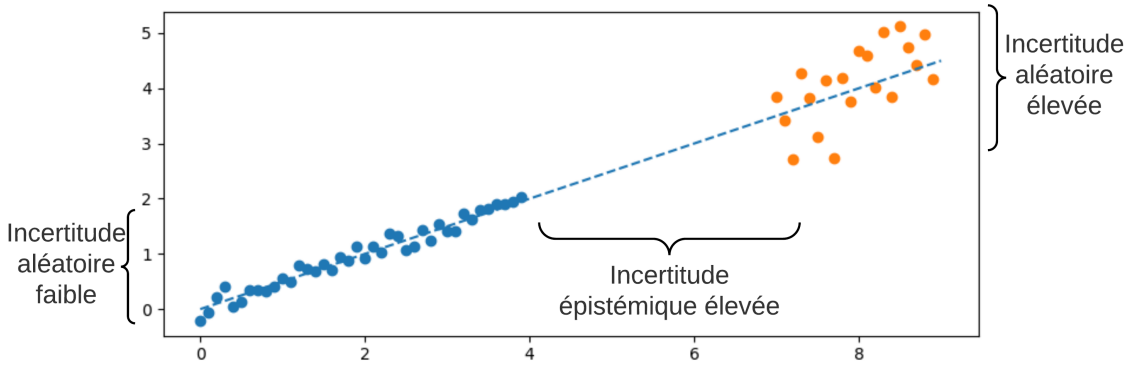


Figure 4.3 – Illustration des deux types d'incertitude.

ries ; l'incertitude aléatoire et l'incertitude épistémique [21]. L'incertitude aléatoire est l'incertitude intrinsèquement présente dans les données d'entrées. Ce type d'incertitude peut être séparé encore une fois en incertitude homoscédastique (constante pour tous les échantillons) et hétéroscédastique (varie en fonction de l'entrée). L'incertitude épistémique est l'incertitude présente dans le modèle lui-même. Cette incertitude est causée par le fait que les données d'entraînement sont disponibles en nombre limité. Ce type d'incertitude peut donc être réduit en augmentant la quantité de données d'entraînement ce qui n'est pas le cas pour l'incertitude aléatoire. Les deux types d'incertitude sont illustrés à la figure 4.3.

4.3.1 Incertitude aléatoire

Pour estimer l'incertitude aléatoire, on revient à la notion de maximisation de la vraisemblance. Cependant, on modélise maintenant la distribution complète en prédisant tous les paramètres. Dans le cas de la régression avec la distribution gaussienne, on prédit la moyenne et la variance associées à une entrée x. Notons ici que ceci modélise donc l'incertitude hétéroscédastique puisque l'incertitude, la variance dans le cas présent, dépendra de l'entrée. Le réseau prend la donc la forme $f_\theta(x) = (\mu(x), \sigma(x))$. Soulignons qu'il est possible d'utiliser un modèle avec plusieurs sorties ou deux modèles

séparés. On reprend l'équation 3.9,

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \frac{1}{\sigma(x_i)\sqrt{2\pi}} e^{-\frac{-(\mu(x_i)-y_i)^2}{2\sigma(x_i)^2}} \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \frac{1}{\sigma(x_i)\sqrt{2\pi}} - \frac{-(\mu(x_i)-y_i)^2}{2\sigma(x_i)^2}.\end{aligned} \tag{4.11}$$

Maintenant, puisque $\sigma(x_i)$ dépend aussi de θ, on n'élimine pas ce terme. On peut exprimer la maximisation sous forme de fonction de coût

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \frac{1}{2\sigma(x_i)^2}(\mu(x_i) - y_i)^2 + \frac{1}{2}\log \sigma(x_i). \tag{4.12}$$

On remarque ici que cette équation ressemble à l'erreur quadratique moyenne, mais aveec un facteur multiplicatif devant l'erreur au carré et un second terme. Ces deux termes impliquent la variance qui est prédite par le modèle. On peut remarquer qu'en prédisant une grande variance, la portion du coût associée à l'erreur au carré sera diminuée. Cependant, cette augmentation de variance vient au prix d'une augmentation du deuxième terme. Cette fonction de coût impose donc au réseau d'apprendre à balancer la prédiction de variance. En effet, ceci peut être vu comme l'apprentissage de l'atténuation de l'erreur du réseau quand le réseau est incertain [21].

4.3.2 Incertitude epistémique

L'incertitude épistémique est modélisée par des variations dans les paramètres du réseau. Plusieurs options sont possibles pour modéliser ce type d'incertitude telles que les réseaux bayésiens, le *Monte-Carlo Dropout* et les ensembles. Dans tous les cas, l'incertitude est normalement prédite en calculant la variance de plusieurs prédictions. Les différentes méthodes sont expliquées ci-dessous et illustrées à la figure 4.4.

Réseaux bayésiens

Jusqu'à présent, les estimations par maximum de vraisemblance et *a posteriori* ont produit des estimations ponctuelles des paramètres. Cependant, il est possible

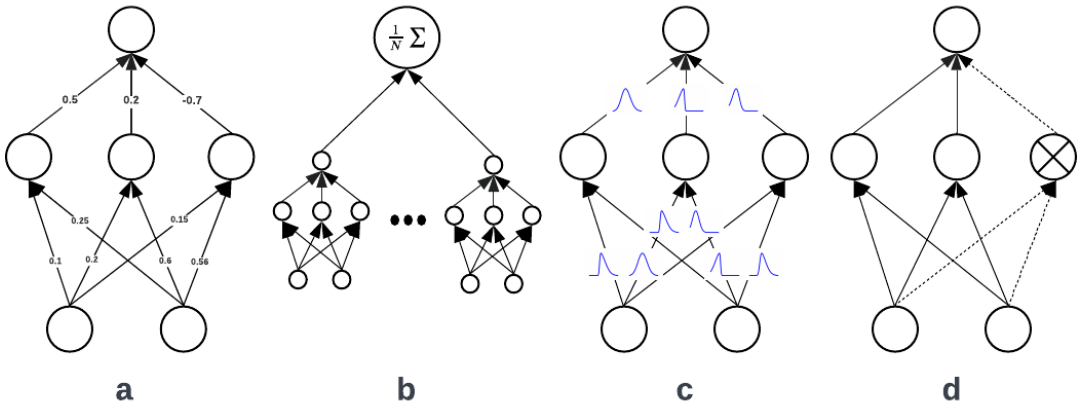


Figure 4.4 – Illustration des techniques d'estimation d'incertitude épistémique. a) Estimation ponctuelle (MLE ou MAP) b) Ensemble de réseaux c) Réseau bayésien d) *Monte-Carlo Dropout*

de modéliser la distribution complète [1]. En reprenant le théorème de Bayes, on remarque qu'il est possible de modéliser la distribution *a posteriori*

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\theta p(\mathcal{D}|\theta)p(\theta)d\theta}. \tag{4.13}$$

Considérant que la distribution a posteriori $p(\theta|\mathcal{D})$ est connue, il est possible de calculer la distribution prédictive de $\hat{y}^*$ sur une nouvelle entrée x^*. La distribution prédictive est donnée par

$$p(y^*|x^*, \mathcal{D}) = \int_\theta p(y^*|x^*, \theta)p(\theta|\mathcal{D})d\theta. \tag{4.14}$$

En pratique, la distribution prédictive est approximée en échantillonnant les paramètres $\theta^t \sim p(\theta|\mathcal{D})$

$$p(y^*|x^*, \mathcal{D}) \approx \frac{1}{T}\sum_{i=1}^{T} \underbrace{p(y^*|x^*, \theta^t)}_{f_{\theta^t}(x^*)}. \tag{4.15}$$

Cependant, la solution analytique pour $p(\theta|\mathcal{D})$ est intraitable pour les réseaux de neurones puisqu'il est impossible d'évaluer $p(\mathcal{D})$ en raison de l'intégrale $\int_\theta p(\mathcal{D}|\theta)p(\theta)d\theta$. La distribution est donc approximée par une distribution variationnelle paramétrée par

ϕ, $q(\theta|\phi)$. On minimise la diverge Kullback–Leibler (KL) afin de trouver les paramètres ϕ. Le problème d'inférence est donc transformé en problème d'optimisation comme suit

$$\begin{aligned} KL(q(\theta|\phi)||p(\theta|\mathcal{D})) &= \int q(\theta|\phi) \log \frac{q(\theta|\phi)}{p(\theta|\mathcal{D})} \\ &= \mathbb{E}_{q(\theta|\phi)} \log \frac{q(\theta|\phi)}{p(\theta|\mathcal{D})} \\ &= \mathbb{E}_{q(\theta|\phi)} \Big[\log q(\theta|\phi) - \log p(\mathcal{D}|\theta) - \log p(\theta) + \log(\mathcal{D}) \Big] \\ &= \underbrace{\mathbb{E}_{q(\theta|\phi)} \Big[\log q(\theta|\phi) - \log p(\theta) \Big]}_{KL(q(\theta|\phi)||p(\theta))} - \underbrace{\mathbb{E}_{q(\theta|\phi)} \Big[\log p(\mathcal{D}|\theta) \Big]}_{\text{Vraisemblance}} + \log(\mathcal{D}). \end{aligned} \tag{4.16}$$

On remarque que le terme intraitable $p(\mathcal{D})$ est toujours présent. Cependant comme la KL divergence est toujours positive, on constate qu'on peut tout de même minimiser cette équation en minimisant que les deux premiers termes. On nomme la somme de ces termes la *ELBO loss*, pour *Evidence lower bound*. On dénote ce terme $\mathcal{F}(\mathcal{D}, \theta)$. Si on réorganise les termes, on obtient

$$\begin{aligned} \mathcal{F}(\mathcal{D}, \theta) &= KL(q(\theta|\phi)||p(\theta)) - \mathbb{E}_{q(\theta|\phi)} \Big[\log p(\mathcal{D}|\theta) \Big] \\ &= \mathbb{E}_{q(\theta|\phi)} \Big[\log q(\theta|\phi) \Big] - \mathbb{E}_{q(\theta|\phi)} \Big[\log p(\theta) \Big] - \mathbb{E}_{q(\theta|\phi)} \Big[\log p(\mathcal{D}|\theta) \Big]. \end{aligned} \tag{4.17}$$

Comme tous les termes de $\mathcal{F}(\mathcal{D}, \theta)$ sont des espérances mathématiques par rapport à $q(\theta|\phi)$, on peut approximer $\mathcal{F}(\mathcal{D}, \theta)$ en échantillonnant des valeurs de θ à partir de $q(\theta|\phi)$

$$\mathcal{F}(\mathcal{D}, \theta) \approx \frac{1}{T} \sum_{i=1}^{T} \log q(\theta^t|\phi) - \log p(\theta^t) - \log p(\mathcal{D}|\theta^t). \tag{4.18}$$

Pour entrainer le un réseau de neurones bayésien, il faut choisir une distribution pour l'*a posteriori* $p(\theta|\mathcal{D})$. On utilise normalement une distribution gaussienne. Les

paramètres ϕ représentent donc la moyenne et la variance de la distribution de paramètre θ. La distribution prend la forme

$$p(\theta|\mathcal{D}) = \mathcal{N}(\mu, \sigma\boldsymbol{I}) \tag{4.19}$$

avec $\phi = \{\mu, \sigma\}$. Lors de l'entraînement, on doit échantillonner une valeur pour chaque paramètre $\theta^t \sim \mathcal{N}(\mu, \sigma\boldsymbol{I})$. Cependant, il n'est pas possible d'effectuer la rétropropagation dans un noeud stochastique tel que l'échantillonnage. On utilise donc astuce de reparamétrage [26]. On obtient

$$\theta^t \sim \mathcal{N}(\mu, \sigma I) \equiv \theta^t = \mu + \sigma \odot \epsilon, \tag{4.20}$$

où $\odot$ est le produit matriciel de Hadamard (multiplication point par point) et $\epsilon \sim \mathcal{N}(0, I)$. Comme il n'est pas nécessaire d'avoir le gradient pour ϵ, cette valeur peut être traitée comme une constante lors de la rétropropagation et le noeud devient une addition et une multiplication par rapport à μ et σ.

Monte-Carlo Dropout

Bien que les réseaux de neurones bayésiens soient un formalisme intéressant, ils ne sont pas faciles à implémenter et entraîner. Gal *et al.* ont proposé un alternative en utilisant la couche *Dropout* [12]. Ils ont démontré que l'utilisation de la couche *Dropout*, normalement désactivé en phase de test, lors de plusieurs propagations avant sur de nouvelles données permet d'approximer les réseaux bayésiens. La distribution de *Dropout* devient donc une approximation distribution $q(\theta|\phi)$ et une estimation d'incertitude peut être réalisée avec l'équation 4.15.

Ensembles

Une autre alternative est l'utilisation d'ensemble d'un réseaux. Il est possible d'évaluer l'incertitude épistémique en entrainant plusieurs réseaux, un ensemble, avec différents hyperparamètres et sur différent sous-ensemble de l'ensemble d'entraînement. On peut ensuite analyser la variance de leurs différents sorties pour estimer l'incertitude [28]. Bien que cette méthode soit performante en termes de performance en prédiction

et pour estimer l'incertitude, elle est très coûteuse, surtout quand la ou les architectures de réseaux utilisées sont grandes.

Chapitre 5

Article - Estimation d'incertitude pour la segmentation d'images médicales

Le chapitre précédent a introduit les bases de l'estimation d'incertitude pour la classification et la régression. L'article présenté dans ce chapitre abordera l'estimation d'incertitude pour la tâche de segmentation. Tel que mentionné au chapitre 3 , la segmentation est une extension de la classification. Les méthodes présentées au chapitre 4 peuvent donc être appliquées à la segmentation en fournissant une prédiction d'incertitude par pixel. Ceci donne lieu à des cartes d'incertitude, une matrice de la même taille que l'image dans laquelle chaque valeur indique le niveau d'incertitude à cette position de l'image.

Au cours du développement de ce projet, nous avons constaté que l'utilisation de méthodes de classification adaptées pour la segmentation produisent des résultats sous optimaux. L'incertitude est prédite pour chaque pixel de manière indépendante ce qui ne garantit aucune cohérence dans la carte d'incertitude. Il a rapidement été établi que ces méthodes ne pouvaient donc pas prédire adéquatement l'erreur des cartes de segmentation même si les erreurs étaient considérables.

Nous avons donc mis au point une méthode qui fait fi de l'interprétation probabiliste des réseaux de segmentation, mais s'appuie plutôt sur l'apprentissage d'un

espace de représentations conjointes qui encode la distribution de formes anatomiques possibles ainsi que leur relation avec l'image d'entrée correspondante. Nous nous sommes inspirés de la méthode CLIP qui apprend un espace conjoint entre du texte et des images [43]. Nous avons modifié cette méthode pour apprendre un espace conjoint entre les cartes de segmentation et les images ultrasonores. Nous avons ensuite utilisé cet espace conjoint pour comparer de nouvelles cartes de segmentation prédites à un sous-ensemble de cartes de segmentation plausibles en fonction de l'image d'entrée.

Nous avons démontré que l'approche proposée surpasse les performances des méthodes de l'état de l'art pour identifier les régions ayant la plus grande probabilité d'erreur. Ces résultats ont été démontrés sur deux bases de données de segmentation d'échocardiographique et deux bases de données de segmentation de poumons par rayon X.

Les contributions de l'article sont les suivantes :

— Nous identifions les problèmes avec les techniques d'estimation d'incertitudes de segmentation qui sont des extensions de techniques de classification.
— Nous proposons une adaptation de la méthode CLIP que nous appliquons à l'estimation d'incertitude en comparant les prédictions à un ensemble de cartes de segmentation plausibles dans l'espace latent conjoint.
— Nous proposons d'utiliser l'information mutuelle entre la carte d'incertitude et la carte de différence par pixel entre la prédiction et la vérité terrain.

Les contributions des auteurs sont :

— Définition du cadre méthodologique (Thierry Judge, Olivier Bernard et Pierre-Marc Jodoin)
— Développement du code informatique (Thierry Judge)
— Rédaction de l'article (Thierry Judge, Olivier Bernard et Pierre-Marc Jodoin)
— Relecture et correction de l'article (Olivier Bernard, Mihaela Porumb, Agisilaos Chartsias, Arian Beqiri et Pierre-Marc Jodoin)

Cet article a été publié à la conférence *Medical Image Computing and Computer Assisted Intervention* (MICCAI) 2022.

CRISP - Reliable Uncertainty Estimation for Medical Image Segmentation

Thierry Judge[1], Olivier Bernard[3], Mihaela Porumb[2], Agisilaos Chartsias[2], Arian Beqiri[2] and Pierre-Marc Jodoin[1]

[1]Department of Computer Science, University of Sherbrooke, Canada
[2]Ultromics Ltd., Oxford, OX4 2SU, UK
[3]University of Lyon, CREATIS, CNRS UMR5220, Inserm U1294, INSA-Lyon, University of Lyon 1, Villeurbanne, France

Abstract

Accurate uncertainty estimation is a critical need for the medical imaging community. A variety of methods have been proposed, all direct extensions of classification uncertainty estimations techniques. The independent pixel-wise uncertainty estimates, often based on the probabilistic interpretation of neural networks, do not take into account anatomical prior knowledge and consequently provide sub-optimal results to many segmentation tasks. For this reason, we propose *CRISP* a ContRastive Image Segmentation for uncertainty Prediction method. At its core, *CRISP* implements a contrastive method to learn a joint latent space which encodes a distribution of valid segmentations and their corresponding images. We use this joint latent space to compare predictions to thousands of latent vectors and provide anatomically consistent uncertainty maps. Comprehensive studies performed on four medical image databases involving different modalities and organs underlines the superiority of our method compared to state-of-the-art approaches. Code is available at: `https://github.com/ThierryJudge/CRISP-uncertainty`.

5.1 Introduction

Deep neural networks are the *de facto* solution to most segmentation, classification and clinical metric estimation. However, they provide no anatomical guarantees nor any safeguards on their predictions. Error detection and uncertainty estimation methods are therefore paramount before automatic medical image segmentation systems can be effectively deployed in clinical settings.

In this work, we present a novel uncertainty estimation method based on joint representations between images and segmentations trained with contrastive learning. Our method, *CRISP* (ContRastive Image Segmentation for uncertainty Prediction), uses this representation to overcome the limitations of state-of-the-art (SOTA) methods which heavily rely on probabilistic interpretations of neural networks as is described below.

Uncertainty is often estimated assuming a probabilistic output function by neural networks. However, directly exploiting the maximum class probability of the *Softmax* or *Sigmoid* usually leads to suboptimal solutions [13]. Some improvements can be made by considering the entire output distribution through the use of entropy [46] or by using other strategies such as temperature scaling [13].

Uncertainty may also come from Bayesian neural networks, which learn a distribution over each parameter using a variational inference formalism [21]. This enables weight sampling, which produces an output distribution that can model the prediction uncertainty. As Bayesian networks are difficult to train, they are often approximated by aggregating the entropy of many dropout forward runs [12]. Alternatively, a network ensemble trained with different hyper-parameters can also estimate uncertainties through differences in predictions [28].

In addition to modeling weight uncertainty, referred to as epistemic uncertainty, uncertainty in the data itself (aleatoric) can also be predicted [22]. However, it has been shown that these methods are less effective for segmentation [19].

Other methods explicitly learn an uncertainty output during training. DeVries and Taylor [10] proposed Learning Confidence Estimates (LCE) by adding a confidence output to the network. The segmentation prediction is interpolated with the ground truth according to this confidence. This confidence can also be learned after training

by adding a confidence branch and finetuning a pre-trained network. This enables learning the True Class Probability which is a better confidence estimate than the maximum class probability [6].

Recent works have modeled the disagreement between labelers for ambiguous images [24, 5]. Both these methods use a form of variational sampling to make their output stochastic. However, these methods require datasets with multiple labels per image to perform at their best. As these datasets are rarely available, we consider these methods out of scope for this paper.

With the exception of methods modeling disagreement, all other methods can be applied to classification and, by extension, to segmentation tasks with an uncertainty prediction at each pixel. In theory, uncertainty maps should identify areas in which the prediction is erroneous. However, as these methods produce per-pixel uncertainties, they do not take into account higher-level medical information such as anatomical priors. Such priors have been used in segmentation [54, 36], but are yet to be exploited in uncertainty estimation. For instance, Painchaud et al. [41] remove anatomical errors by designing a latent space dedicated to the analysis and correction of erroneous cardiac shapes. However, this approach does not guarantee that the corrected shape matches the input image.

To this end, we propose *CRISP*, a method which does not take into account the probabilistic nature of neural networks, but rather uses a joint latent representation of anatomical shapes and their associated image. This paper will describe the *CRISP* method and propose a rigorous evaluation comparing *CRISP* to SOTA methods using four datasets.

5.2 *CRISP*

The overarching objective of our method is to learn a joint latent space, in which the latent vector of an input image lies in the vicinity of its corresponding segmentation map's latent vector in a similar fashion as the "CLIP" method does for images and text [43]. As such, a test image x whose latent vector does not lie close to that of its segmentation map y is an indication of a potentially erroneous segmentation. Further details are given below.

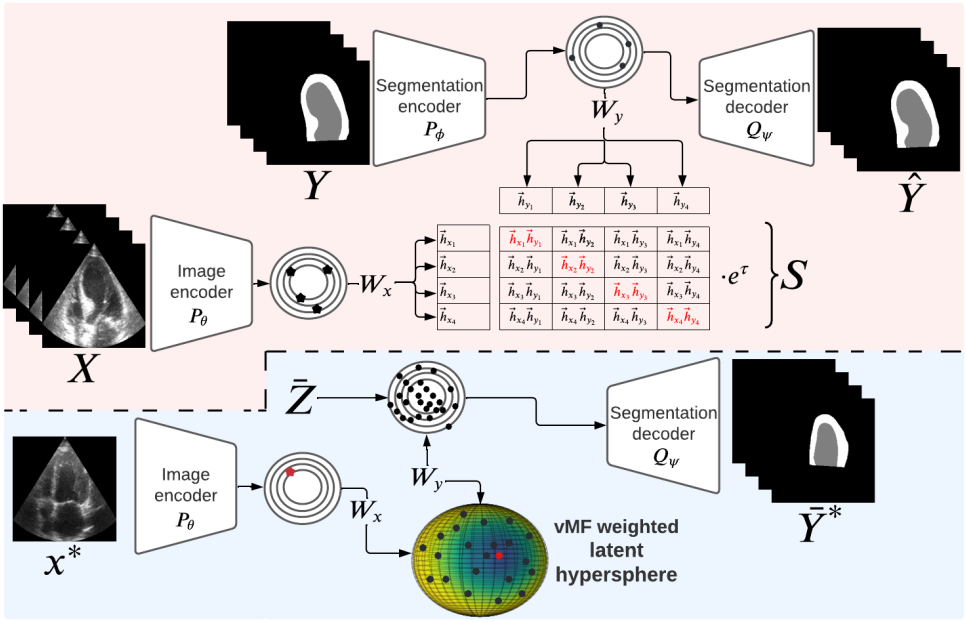


Figure 5.1 – Schematic representation of our method. Top depicts the training phase and bottom illustrate the uncertainty estimation on an input-prediction pair (x^*, y^*).

Training. As shown in Figure 5.1, at train time, *CRISP* is composed of two encoders: the image encoder P_θ and the segmentation encoder P_ϕ. They respectively encode an image x_i and its associated segmentation groundtruth y_i into latent vectors $\vec{z}_{x_i} \in \Re^{D_x}$ and $\vec{z}_{y_i} \in \Re^{D_y}$ $\forall i$. Two weight matrices $W_x \in \Re^{D_h \times D_x}$ and $W_y \in \Re^{D_h \times D_y}$ linearly project the latent vectors into a joint D_h-dimensional latent space where samples are normalized and thus projected onto a hyper-sphere. As such, the image latent vector $\vec{z}_{x_i}$ is projected onto a vector $\vec{h}_{x_i} = \frac{W_x \cdot \vec{z}_{x_i}}{||W_x \cdot \vec{z}_{x_i}||}$ and similarly for $\vec{z}_{y_i}$. A successful training should lead to a joint representation for which $\vec{h}_{x_i} \approx \vec{h}_{y_i}$.

During training, images and groundtruth maps are combined into batches of B elements, $\boldsymbol{X} = [x_1 x_2 ... x_B] \in \Re^{B \times C \times H \times W}$ and $\boldsymbol{Y} = [y_1 y_2 ... y_B] \in \{0,1\}^{B \times K \times H \times W}$ for images with C channels and K segmentation classes. As mentioned before, these batches are encoded by P_θ and P_ϕ into sets of latent vectors Z_X and Z_Y and then projected and normalized into sets of joint latent vectors H_X and H_Y.

At this point, a set of $2 \times B$ samples lie on the surface of a unit hyper-sphere of the joint latent space. Much like CLIP [43], the pair-wise distance between these joint latent

vectors is computed with a cosine similarity that we scale by a learned temperature factor τ to control the scale of the logits. This computation is done by taking a weighted product between H_X and H_Y which leads to the following square matrix: $S = (H_X \cdot H_Y^T)e^\tau \in \Re^{B\times B}$. As shown in Figure 5.1, the diagonal of S corresponds to the cosine similarity of the latent image vectors with their corresponding latent groundtruth vector while the off-diagonal elements are cosine similarity of unrelated vectors.

The goal during training is to push S towards an identity matrix, such that the latent vectors $\vec{h}_{x_i}$ and $\vec{h}_{y_j}$ lie on the same spot in the joint latent space when $i = j$ and are orthogonal when $i \neq j$. This would lead to similarities close to 1 on the diagonal and close to 0 outside of it. To enforce this, a cross-entropy loss on the rows and columns of S is used as a contrastive loss [43],

$$\mathcal{L}_{cont} = -\frac{1}{2}\left(\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} I_{ij}\log S_{ij} + \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} I_{ji}\log S_{ji}\right). \tag{5.1}$$

CRISP also has a segmentation decoder Q_ψ to reconstruct segmentation latent vectors, a critical feature for estimating uncertainty. This decoder is trained with a reconstruction loss $\mathcal{L}_{rec}$ which is a weighted sum of the Dice coefficient and cross-entropy loss. The model is trained end-to-end to minimize $\mathcal{L} = \mathcal{L}_{cont} + \mathcal{L}_{rec}$.

Uncertainty prediction. Once training is over, the groundtruth segmentation maps $\boldsymbol{Y}$ are projected one last time into the Z and H latent spaces. This leads to a set of N latent vectors $\bar{Z} \in \Re^{N\times D_y}$ and $\bar{H} \in \Re^{N\times D_h}$ which can be seen as latent anatomical prior distributions that will be used to estimate uncertainty.

Now let x^* be a non-training image and y^* its associated segmentation map computed with a predetermined segmentation method (be it a deep neural network or not). To estimate an uncertainty map, x^* is projected into the joint latent space to get its latent vector $\vec{h}_{x^*} \in \Re^{D_h}$. We then compute a weighted dot product between $\vec{h}_{x^*}$ and each row of $\bar{H}$ to get $\bar{S} \in \Re^N$, a vector of similarity measures between $\vec{h}_{x^*}$ and every groundtruth latent vector. Interestingly enough, the way *CRISP* was trained makes $\bar{S}$ a similarity vector highlighting how each groundtruth map fits the input image x^*.

Then, the M samples of $\bar{Z}$ with the highest values in $\bar{S}$ are selected. These samples

are decoded to obtain $\bar{Y}^*$, *i.e.* various anatomically valid segmentation maps whose shapes are all roughly aligned on x^*. To obtain an uncertainty map, we compare these samples to the initial prediction y^*. We compute the average of the pixel-wise difference between y^* and $\bar{Y}^*$ to obtain an uncertainty map

$$U = \frac{1}{M} \sum_{i=1}^{M} w_i (\bar{y}_i^* - y^*). \tag{5.2}$$

As not all samples equally correspond to x^*, we add a coefficient w_i which corresponds to how close a groundtruth map y_i is from x^*. Since the joint latent space is a unit hyper-sphere, we use a *von Mises-Fisher distribution* (vMF) [32] centered on $\vec{h}_{x^*}$ as a kernel to weigh its distance to $\vec{h}_{y_i}$. We use Taylor's method [49] to define the kernel bandwidth b (more details are available in the supplementary materials). We define the kernel as:

$$w_i = e^{\frac{1}{b}\vec{h_i}^T \vec{h}_{x^*}} / e^{\frac{1}{b}\vec{h}_{x^*}^T \vec{h}_{x^*}} = e^{\frac{1}{b}(\vec{h_i}^T \vec{h}_{x^*} - 1)}. \tag{5.3}$$

5.3 Experimental setup

5.3.1 Uncertainty metrics

Correlation. Correlation is a straightforward method for evaluating the quality of uncertainty estimates for a full dataset. The absolute value of the Pearson correlation score is computed between the sample uncertainty and the Dice score. In this paper, sample uncertainty is obtained by dividing the sum of the uncertainty for all pixels by the number of foreground pixels. Ideally, the higher the Dice is, the lower the sample uncertainty should be. Therefore, higher correlation values indicate more representative uncertainty maps.

Calibration. A classifier is calibrated if its confidence is equal to the probability of being correct. Calibration is expressed with Expected Calibration Error (ECE) computed by splitting all n samples into m bins and computing the mean difference between the accuracy and average confidence for each bin. Please refer to the following paper for more details [39].

Uncertainty-error mutual information. Previous studies have computed Uncertainty-error overlap by obtaining the Dice score between the thresholded uncertainty map and a pixel-wise error map between the prediction and the ground-truth segmentation map [19]. As the uncertainty error overlap requires the uncertainty map to be thresholded, much of the uncertainty information is lost. We therefore propose computing the mutual information between the raw uncertainty map and the pixel-wise error map. We report the average over the test set weighted by the sum of erroneous pixels in the image.

5.3.2 Data

CAMUS. The CAMUS dataset [30] consists of cardiac ultrasound clinical exams performed on 500 patients. Each exam contains the 2D apical four-chamber (A4C) and two-chamber view (A2C) sequences. Manual delineation of the endocardium and epicardium borders of the left ventricle (LV) and atrium were made by a cardiologist for the end-diastolic (ED) and end-systolic (ES) frames. The dataset is split into training, validation and testing sets of 400, 50 and 50 patients respectively.

HMC-QU. The HMC-QU dataset [11] is composed of 162 A4C and 130 A2C view recordings. 93 A4C and 68 A2C sequences correspond to patients with scarring from myocardial infarction. The myocardium (MYO) of 109 A4C (72 with myocardial infarction/37 without) recordings was manually labeled for the full cardiac cycle. These sequences were split into training, validation and testing sets of 72, 9 and 28 patients.

Shenzen. The Shenzen dataset [18] is a lung X-ray dataset acquired for pulmonary tuberculosis detection. The dataset contains 566 postero-anterior chest radiographs and corresponding manually segmented masks to identify the lungs. The dataset was split into training and validation sets of 394 and 172 patients.

JSRT. We use the Japanese Society of Radiological Technology (JSRT) [45] lung dataset which contains images and segmentation maps for 154 radiographs with lung nodules, and corresponding segmentation masks.

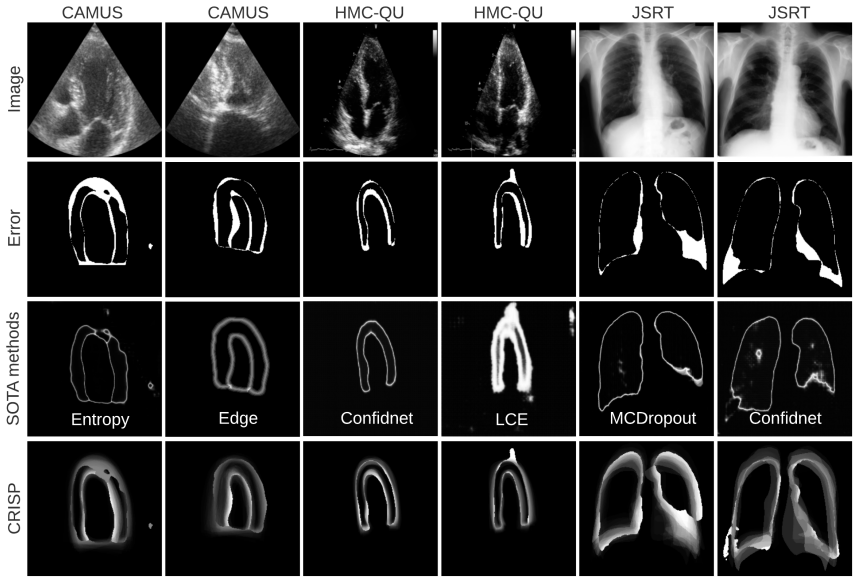


Figure 5.2 – From top to bottom: raw images, corresponding error maps, uncertainty estimation of SOTA methods and CRISP uncertainty. White indicates erroneous pixels in the error maps [row 2] and high uncertainty in the uncertainty maps [rows 3 and 4].

5.3.3 Implementation details

CRISP was compared to several SOTA methods mentioned before. To make comparison fair, every method use the same segmentation network (an Enet [38] in our case). All methods were trained with a batch size of 32 and the Adam optimizer [20] with a learning rate of 0.001 and weight decay of 1e-4. We added early stopping and selected the weights with the lowest validation loss. The **Entropy** method was tested using the baseline network. We tested **MC Dropout** by increasing the baseline dropout value from 10% to 25% and 50% (we report best results with respect to the Dice score) and computing the average of 10 forward passes. For **LCE**, we duplicated the last bottleneck of the Enet to output confidence. The **Confidnet** method was trained on the baseline Enet pre-trained network. The full decoder was duplicated to predict the True Class Probability. For methods or metrics that require converting pixel-wise confidence (c) to uncertainty (u), we define the relationship between the two as $u = 1 - c$ as all methods produce values in the range $[0, 1]$.

Table 5.1 – Uncertainty estimation results (average over 3 random seeds) for different methods. Bold values indicate best results.

Training data	CAMUS			CAMUS			Shenzen		
Testing data	CAMUS			HMC-QU			JSRT		
Method	Corr. ↑	ECE ↓	MI ↑	Corr. ↑	ECE ↓	MI ↑	Corr. ↑	ECE ↓	MI ↑
Entropy	0.66	0.12	0.02	0.34	0.27	0.02	**0.89**	0.08	0.02
ConfidNet [1]	0.34	0.08	0.04	0.36	0.17	0.04	0.69	0.09	0.01
CRISP	**0.71**	0.09	**0.20**	**0.41**	0.14	**0.06**	0.83	0.19	**0.11**
McDropout [3]	0.67	0.13	0.03	0.26	0.26	0.02	**0.82**	**0.06**	0.03
CRISP-MC	**0.78**	**0.11**	**0.26**	**0.29**	**0.14**	**0.06**	**0.82**	0.21	**0.08**
LCE [2]	0.58	0.44	0.08	**0.35**	0.37	**0.07**	**0.87**	0.37	0.06
CRISP-LCE	**0.59**	**0.08**	**0.15**	0.34	**0.13**	**0.07**	0.85	**0.18**	**0.11**

To highlight some limitations of SOTA methods, we also added a naïve method for computing uncertainty which we referred to as **Edge**. The uncertainty map for *Edge* amounts to a trivial edge detector applied to baseline predicted segmentation maps. The resulting borders have a width of 5 pixels.

As our **CRISP** method can be used to evaluate any image-segmentation pair, regardless of the segmentation method, we tested it on all the segmentation methods that produce different results (baseline, MC Dropout, LCE). This allows for a more robust evaluation as the evaluation of uncertainty metrics is directly influenced by the quality of the segmentation maps [19]. The value of M was determined empirically and kept proportional to the size of $\bar{Z}$. It can be noted, that the vMF weighting in the latent space attenuates the influence of M.

5.3.4 Experimental setup

We report results on both binary and multi-class segmentation tasks. As our datasets are relatively large and homogeneous, Dice scores are consistently high. This can skew results as methods can simply predict uncertainty around the prediction edges. Thus, as mentioned below, we tested on different datasets or simulated domain shift through data augmentation.

Tests were conducted on the CAMUS dataset for LV and MYO segmentation. We simulated a domain shift by adding brightness and contrast augmentations (factor=0.2) and Gaussian noise ($\sigma^2 = 0.0001$) with probability of 0.5 for all test images. We used the 1800 samples from the training and validation sets to make up the $\bar{Z}$ set and used $M = 50$ samples to compute the uncertainty map.

We also tested all methods trained on the CAMUS dataset on the HMC-QU dataset for myocardium segmentation. We added brightness and contrast augmentations (factor=0.2) and RandomGamma (0.95 to 1.05) augmentations during training and normalized the HMC-QU samples using the mean and variance of the CAMUS dataset. We used the A4C samples from the CAMUS dataset (along with interpolated samples between ES and ED instants) to create the set of latent vectors $\bar{Z}$. This corresponds to 8976 samples, of which $M = 150$ were selected to compute the uncertainty map.

Finally, we tested our method on a different modality and organ by using the lung X-ray dataset. We trained all the methods on the Shenzen dataset and tested on the JSRT dataset. We normalized JSRT samples with the mean and variance of the Shenzen dataset. We used the 566 samples from the Shenzen dataset to form the $\bar{Z}$ set and used $M = 25$ samples to compute the uncertainty.

5.4 Results

Uncertainty maps are presented in Figure 5.2 for samples on 3 datasets. As can be seen, *Entropy*, *ConfidNet*, and *MCDropout* have a tendency to work as an edge detector, much like the naive *Edge* method. As seen in Table 5.1, different methods perform to different degrees on each of the datasets. However, CRISP is consistently the best or competitive for all datasets for the correlation and MI metrics. ECE results for *CRISP* are also competitive but not the best. Interestingly, the trivial *Edge* method often reports the best ECE results. This is probably due to the fact that errors are more likely to occur near the prediction boundary and the probability of error decreases with distance. These results might encourage the community to reconsider the value of ECE for specific types of segmentation tasks.

Figure 5.3 shows the distribution of pixel confidence according to the well-classified and misclassified pixels. This figure allows for a better understanding of the different

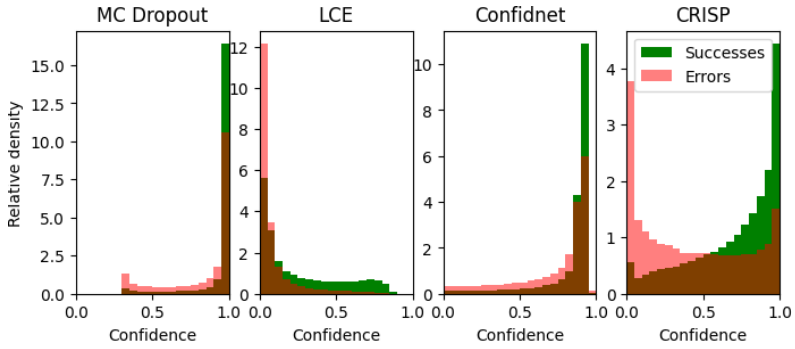


Figure 5.3 – Histograms of well classified pixels (Successes) and mis-classified pixels (Errors) for different methods on the HMC-QU dataset.

shortcomings of each method. It clearly shows that both MC Dropout and *Confidnet* methods produce over-confident results. On the other hand, LCE appears to produce slightly under-confident predictions which explains the higher mutual information value. Finally, *CRISP* is the only method that can clearly separate certain and uncertain pixels. These results are consistent with what is observed in Figure 5.2 as both MC Dropout and *Confidnet* produce very thin uncertainty and LCE predicts large areas of uncertainty around the border. Only *CRISP* produces varying degrees of uncertainty according to the error.

It is apparent that there is a slight decrease in performance for *CRISP* on the JSRT dataset. This is most likely caused by the fact that the latent space is not densely populated during uncertainty estimation. Indeed, the 566 samples in $\bar{Z}$ might not be enough to produce optimal uncertainty maps. This is apparent in Figure 5.2 where the uncertainty maps for the JSRT samples are less smooth than the other datasets that have more latent vectors. Different techniques such as data augmentation or latent space rejection sampling [41] are plausible solutions.

5.5 Discussion and conclusion

While empirical results indicate that all methods perform to a certain degree, qualitative results in Figure 5.2 show that most SOTA methods predict uncertainty around the prediction edges. While this may constitute a viable uncertainty prediction when the predicted segmentation map is close to the groundtruth, these uncertainty

estimates are useless for samples with large errors. Whereas in other datasets and modalities, the uncertainty represents the probability of a structure being in an image and at a given position, lung X-Ray and cardiac ultrasound structures are always present and are of regular shape and position. This makes the task of learning uncertainty during training challenging as few images in the training set produce meaningful errors. Compared to other approaches, *CRISP* leverages the information contained in the dataset to a greater degree and accurately predicts uncertainty in even the worst predictions.

To conclude, we have presented a method to identify uncertainty in segmentation by exploiting a joint latent space trained using contrastive learning. We have shown that SOTA methods produce sub-optimal results due to the lack of variability in segmentation quality during training when segmenting regular shapes. We also highlighted this with the naïve *Edge* method. However, due to its reliance on anatomical priors, *CRISP* can identify uncertainty in a wide range of segmentation predictions.

5.6 Supplementary materials

5.6.1 Ablation study

Table 5.2 – Uncertainty estimation results (average over 3 random seeds) for different values of M for our *CRISP* method.

Training data	CAMUS			CAMUS			Shenzen		
Testing data	CAMUS			HMC-QU			JSRT		
Method	M	Corr. ↑	MI ↑	M	Corr. ↑	MI ↑	M	Corr. ↑	MI ↑
CRISP	25	0.72	0.20	50	0.41	0.06	5	0.83	0.10
CRISP	50	0.71	0.20	100	0.41	0.06	10	0.84	0.11
CRISP	100	0.69	0.20	150	0.41	0.06	25	0.84	0.11
CRISP	150	0.68	0.20	250	0.41	0.06	50	0.84	0.11
CRISP	250	0.67	0.19	500	0.41	0.06	100	0.84	0.11

5.6.2 von Mises-Fisher kernel

To define the kernel used at the end of section 2, we find the maximum likelihood parameters for the mean direction, μ, and the concentration parameter, κ, describing the vMF of all the samples $\bar{H}$. The mean direction of the distribution is defined with

$$\vec{h}_m = \frac{1}{N}\sum_{i}^{N}\vec{h}_i \quad \text{and} \quad \mu = \frac{\vec{h}_m}{r_m} \tag{5.4}$$

where $r_m = ||\vec{h}_m||$. We estimate the concentration parameter for a D_h-dimensional space using the following equation [2]:

$$\kappa = \frac{r_m(D_h - r_m^2)}{1 - r_m^2}. \tag{5.5}$$

We use these values to define the kernel bandwidth following Taylor's method [49]:

$$b = \kappa^{-\frac{1}{2}}(\frac{40\sqrt{\pi}}{N})^{\frac{1}{5}}. \tag{5.6}$$

5.6.3 Edge uncertainty

Edge is a simple morphological edge-detection method. Let δ and ξ be dilation and erosion operations, $f^n(\cdot)$ a set of n successive applications of a morphological operator f ($f^0(\cdot)$ is no operation) and I the predicted segmentation map, the *Edge* uncertainty map is computed as

$$U_{edge} = \sum_{n=1}^{5}\Big(\xi^n(I) - \xi^{n-1}(I) + \delta^n(I) - \delta^{n-1}(I)\Big)\cdot\Big(1 - n/5\Big) \tag{5.7}$$

5.6.4 Supplementary results

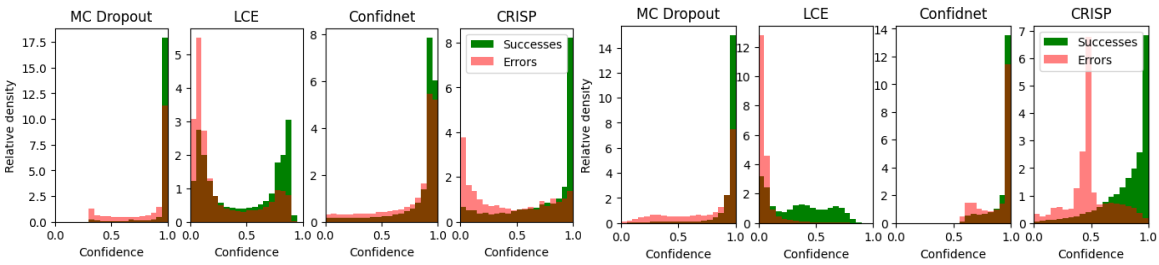


Figure 5.4 – Histograms on the CAMUS (5.4a) and JSRT datasets. (5.4b).

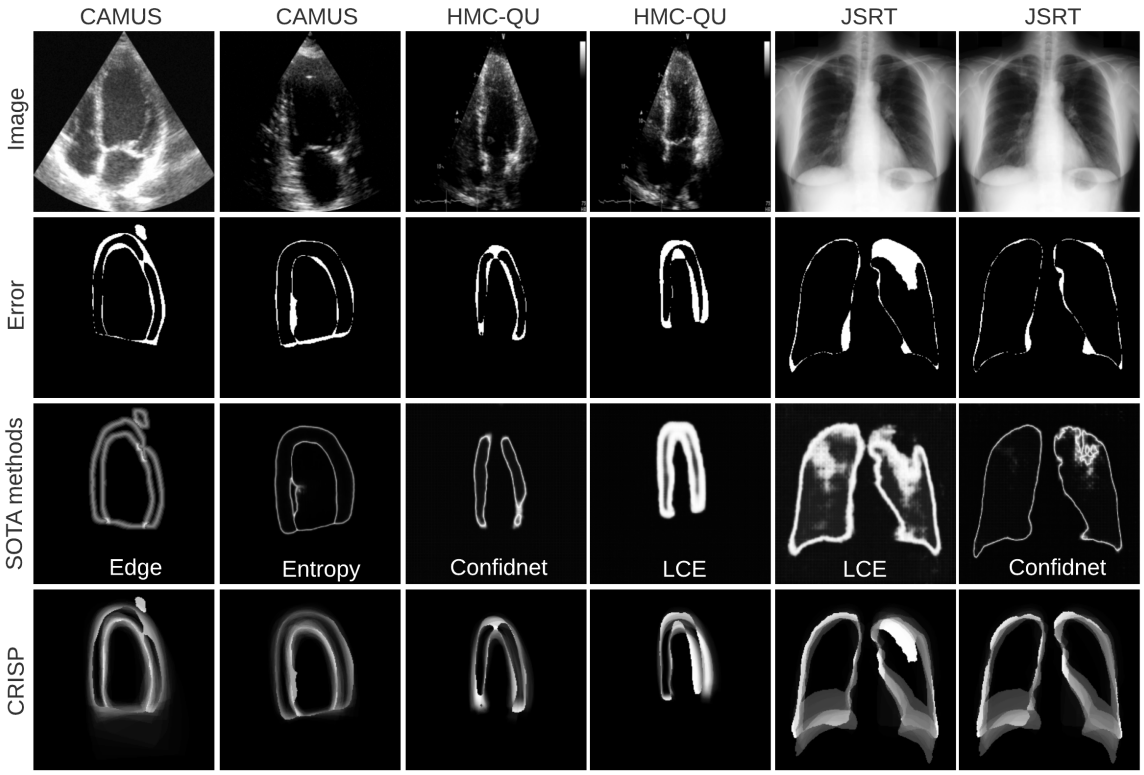


Figure 5.5 – Supplementary samples for Fig. 2.

Chapitre 6

Article - Apprentissage semi-supervisé avec contraintes anatomiques pour la segmentation échocardiographique

Cet article fait suite à des travaux publiés par notre laboratoire, le VITALAB, ayant pour but d'identifier et corriger des erreurs anatomiques dans des segmentations cardiaques à l'aide de post-traitement [37, 40]. Ces travaux ont défini des métriques anatomiques afin d'identifier les cartes de segmentation ayant des erreurs anatomiques (trous, concavités, *etc.*) pour des images d'IRM et d'ultrasons cardiaques. Les travaux ont présenté une méthode de post-traitement à base d'auto-encodeurs variationnels pour corriger ces segmentations erronées.

Le but de l'article présenté dans ce chapitre était de réduire le nombre d'erreurs anatomiques sans avoir recours à du post-traitement en imposant les contraintes anatomiques durant l'entraînement. Puisque les métriques anatomiques définies dans les travaux précédents ne sont pas différentiables, il est impossible de les optimiser avec une fonction de coût standard et la descente de gradient. Nous avons donc proposé une méthode qui utilise un deuxième réseau de neurones afin d'apprendre la fonction des métriques anatomiques dans le but d'émuler le gradient de cette fonction pour le

réseau de segmentation.

Nous avons appliqué cette méthode à la segmentation d'ultrason cardiaque. Dans un premier temps, nous avons démontré que les réseaux de neurones produisent un nombre important d'erreurs anatomiques, en particulier quand la quantité de données annotées est faible. Cependant, avec l'ajout de contraintes anatomiques, le nombre d'erreurs anatomiques est réduit même pour de très petites quantités de données annotées.

De plus, puisque l'optimisation de métriques anatomiques ne requiert pas une vérité terrain, la méthode proposée une est candidate idéale pour l'apprentissage semi-supervisé, un mode d'apprentissage qui utilise des données annotées et non-annotées. Dans ce cas, le nombre d'erreurs anatomiquesréduit encore plus.

Les contributions de l'article sont les suivantes :

— Nous proposons une méthode qui permet d'optimiser des métriques non-différentiables sur des données annotées et non-annotées.

— Nous démontrons que notre méthode permet d'augmenter les performances de segmentation et de réduire le nombre d'erreurs anatomiques avec une très petite quantité de données annotées.

Les contributions des auteurs sont :

— Définition du cadre méthodologique (Thierry Judge)

— Développement du code informatique (Thierry Judge et Arnaud Judge)

— Rédaction de l'article (Thierry Judge)

— Relecture et correction de l'article (Pierre-Marc Jodoin et Arnaud Judge)

Cet article a été publié à la conférence *Medical Imaging with Deep Learning* (MIDL) 2022.

Anatomically Constrained Semi-supervised Learning for Echocardiography Segmentation

Thierry Judge, Arnaud Judge, Pierre-Marc Jodoin

Department of Computer Science, University of Sherbrooke, Sherbrooke, QC, Canada

Abstract

Deep convolutional neural networks (CNNs) have had great success for medical imaging segmentation. Many methods attained nearly perfect Dice scores, sometimes within inter-expert variability. However, CNNs require large amounts of labeled data and are not immune to producing anatomically implausible results, especially when applied to ultrasound images. In this paper, we propose a method that tackles both of these problems simultaneously. Our method optimizes anatomical segmentation metrics on both labeled and unlabeled data using a training scheme analogous to adversarial training. Our method allows the optimization of several hand-made non-differentiable metrics for any segmentation model and drastically reduces the number of anatomical errors. The code is available at `https://github.com/ThierryJudge/anatomically-constrained-ssl`.

6.1 Introduction

Convolutional neural networks (CNNs) are the go-to solution for echocardiography segmentation capable of producing results within inter-expert variability [30]. They however required large amounts of labeled data and are prone to making predictions that are not consistent with the underlying anatomy. These anatomical errors often hinder the credibility that neural networks may have in clinical practice. Recently,

[37] have shown to what extent these errors occur in state-of-the-art segmentation methods. While their post-processing method can scrub out anatomical errors, it nonetheless requires a large number of labeled images and constitutes a non-negligible processing overhead which raises questions regarding its real-life usability.

Techniques for enforcing anatomical constraints are essential, especially when little labeled data is available. In this paper, we propose a novel training method to directly optimize non-differentiable anatomical metrics and leverage them to enforce anatomical prior for both labeled and unlabeled data. We do so by using a classification neural net whose back-propagated gradients emulate those that would have been produced by an anatomical loss. We also show that by its very nature, our method adapts to partly annotated data and show that adding unannotated data further improves results. The system is also trained on a scheme that prevents from falling into class imbalance problems.

6.2 Method

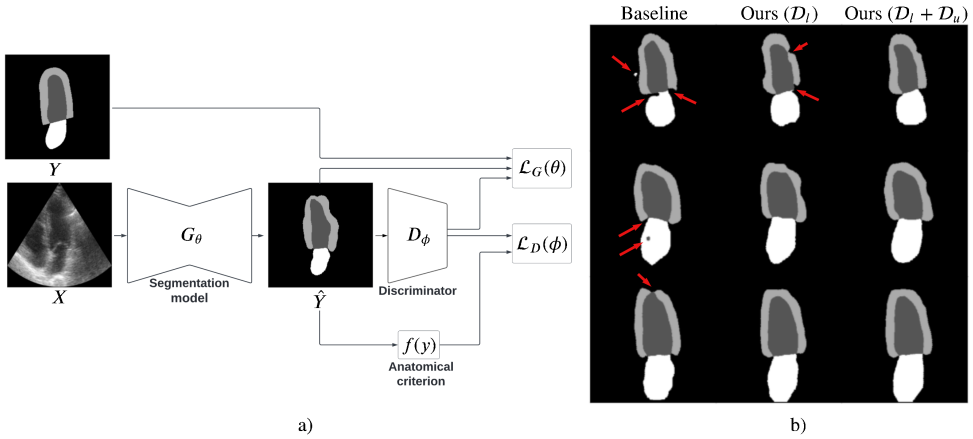


Figure 6.1 – a) Illustration of the proposed method. Generator and discriminator training are shown in blue and red blocks respectively. b) Examples of predictions for different methods. Anatomical errors are indicated with red arrows.

Lets consider $\mathcal{D}_l = \{(x_l^{(1)}, y_l^{(1)}), ..., (x_l^{(n)}, y_l^{(n)})\}$, where $x \in \Re^{C\times H\times W}$ is an input image, $y \in \{0,1\}^{K\times H\times W}$ a ground truth segmentation map, and $G_\theta(x) : \Re^{C\times H\times W} \rightarrow$

$\{0,1\}^{K\times H\times W}$ a segmentation network trained with a loss function $\mathcal{L}_{sup}(x,y)$. While segmentation networks generalize well, they offer no anatomical guarantee what so ever.

At the core of our method is an anatomical constraint function based on 12 anatomical criteria (region connections, sizes, concavities, etc.) outlined in [37]. This function $f(y) : \{0,1\}^{K\times H\times W} \rightarrow \{0,1\}$ returns 0 when the segmentation map y is anatomically invalid (i.e. whenever it violates at least one anatomical criterion) and 1 when every criterion is satisfied.

Since the constraint function is non-differentiable, one cannot use it as a loss function. As a workaround, a follow up network $D_\phi(y) : [0,1]^{K\times H\times W} \rightarrow [0,1]$ is used to emulate it. As shown in Figure 6.1, $D_\phi(y)$ predicts if the input segmentation map y is anatomically valid or not. Since the system is trained end-to-end, the back-propagated gradients force the segmentation network to learn anatomical concepts better adapted to the task at hand than those a network gets to learn when trained solely with a Dice or a cross-entropy loss.

Since $f(y)$ does not rely on groundtruth data, one can use unlabeled sets of data $\mathcal{D}_u = \{x_u^{(1)}, x_u^{(2)}, ..., x_u^{(m)}\}$ to train both networks. This results in the following loss equations:

$$\mathcal{L}_G(\theta) = \sum_{\mathcal{D}_l} \mathcal{L}_{sup}(x_l^{(i)}, y_l^{(i)}) - \lambda_1 \log\left(D_\phi(G_\theta(x_l^{(i)}))\right) - \lambda_2 \sum_{\mathcal{D}_u} \log\left(D_\phi(G_\theta(x_u^{(i)})\right) \tag{6.1}$$

$$\mathcal{L}_D(\phi) = -\sum_{\mathcal{D}_l+\mathcal{D}_u} f(G_\theta(x^{(i)})) \log\left(D_\phi(G_\theta(x^{(i)}))\right) + (1 - f(G_\theta(x^{(i)}))) \log\left(1 - D_\phi(G_\theta(x^{(i)}))\right) \tag{6.2}$$

with $\lambda_1 = 0.025$ and $\lambda_2 = 0.01$. As the discriminator is trained on samples generated by $G_\theta(x)$, there is a high chance of class imbalance with respect to the valid and in-valid segmentations. To alleviate this problem, we use a replay buffer in which is saved an equal number of valid and in-valid samples at every step. The discriminator is trained with batches made of samples randomly selected from the buffer.

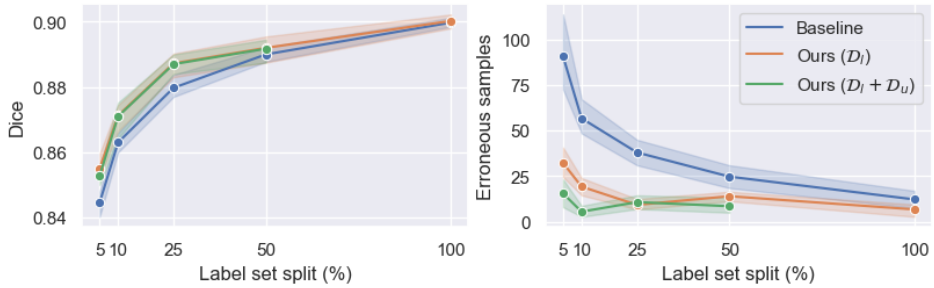


Figure 6.2 – Dice and number of anatomically erroneous samples in the test set (200 samples). Methods trained on datasets containing 5% to 100% of the full training set labels.

6.3 Results

We trained and tested our method on the CAMUS [30] dataset which contains cardiac ultrasound images of 500 patients. The left ventricle, myocardium and left atrium are manually labeled for end-diastolic and end-systolic instants for both the 2D four-chamber and two-chamber apical views.

A standard Enet [**?**] trained with a batch size of 32 was used for all methods. We trained a baseline Enet with a combination of Dice and cross-entropy loss using a constant learning rate of 0.001 with weight decay of 1e-4. We tested our method with labeled data ($\mathcal{D}_l$) and with labeled+unlabeled data ($\mathcal{D}_l + \mathcal{D}_u$). G_θ was initialized with the baseline weights and D_ϕ was pre-trained for 1000 steps to help it better converge. Results in Figure 6.2 show an increase in Dice score when a small amount of labeled data is used to train the system and, most importantly, an important decrease in the number of anatomical errors. The decrease is even more drastic when unlabeled images are included.

6.4 Conclusion

In this paper, we proposed a novel method for optimizing a non-differentiable anatomical prior. As the computation of the anatomical prior does not require the ground truth, our method is suitable for semi-supervised learning.

We show that our method drastically reduces the number of anatomical errors in the test set predictions without requiring any post-processing. We also show that our method improves the dice score as a result of the regularization induced by the anatomical constraints.

Acknowledgments This work was funded by the NSERC Graduate Scholarships – Master's program, the NSERC Discovery Grant and by MITACS Accelerate.

Conclusion et perspectives

Dans un premier temps, ce mémoire a introduit les notions nécessaires à la compréhension des contributions scientifiques présentées. Nous avons introduit les bases de l'anatomie de la physiologie cardiaque. Nous avons ensuite abordé les fondements de l'imagerie ultrasonore, plus particulièrement l'échocardiographie. Suite à ceci, les bases de l'apprentissage profond nécessaires pour segmentation les images échocardiographie ont été présentées. Finalement, les principes théoriques de l'estimation d'incertitude ont été énoncés. Dans un deuxième temps, ce mémoire a présenté deux contributions scientifiques. La première est une technique d'estimation d'incertitude basée sur l'apprentissage d'une représentation conjointe entre images et segmentations. Nous avons démontré que ce type d'approche est plus performant que les méthodes de l'état de l'art basées sur l'interprétation probabiliste des réseaux de neurones. Grâce à la représentation conjointe, notre méthode peut comparer les prédictions à des milliers d'exemples tirés de la base de données et prédire les régions erronées des segmentations les plus probables. La deuxième contribution est une technique d'optimisation de métriques non-différentiables appliquée à l'optimisation de métriques de plausibilité anatomique. Nous avons démontré que cette approche permet de réduire le nombre d'erreurs anatomiques en exploitant les données non-annotées sans avoir recours à des techniques de post-traitement. Malgré ces avancées, ainsi que celles citées tout au long de ce mémoire, la segmentation d'imagerie cardiaque par réseaux de neurones profonds est encore loin d'être prête à l'utilisation en pratique clinique. La pratique clinique requiert des techniques qui fonctionnent dans tous les cas sur une grande variété de patients. Bien que nous avons présenté des méthodes pour estimer l'incertitude et réduire le nombre d'erreurs anatomique dans ce mémoire, ces méthodes sont encore entrainées et testées sur de très petits sous-ensembles des données réellement rencontrées en pratique clinique. Sans des bases de données de plusieurs ordres de magnitude plus grandes, ces techniques ne pourront être appliquées efficacement en pratique clinique. Un autre problème intrinsèquement lié à l'estimation d'incertitude qui devra sans doute être abordé est l'analyse de la variabilité inter et intra

experts. Ce sujet est inévitable pour le futur de la recherche en estimation d'incertitude puisque la modélisation de la variabilité des experts permettra d'effectuer une validation concrète des méthodes d'estimation d'incertitude par rapport à l'incertitude entre humains. Il faut cependant noter que les défis qui accompagnent cette modélisation vont au-delà des détails techniques puisque ce type d'analyse requiert des données annotées par plusieurs experts, ce qui n'est pas disponible publiquement à ce jour.

Bibliographie

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, et D. Wierstra, « Weight Uncertainty in Neural Network, » dans *Proceedings of the 32nd International Conference on Machine Learning*, série Proceedings of Machine Learning Research, F. Bach et D. Blei, éditeurs, vol. 37. Lille, France : PMLR, 07–09 Jul 2015, pp. 1613–1622.

[2] A. Banerjee, I. S. Dhillon, J. Ghosh, et S. Sra, « Clustering on the Unit Hypersphere using von Mises-Fisher Distributions, » *Journal of Machine Learning Research*, vol. 6, no. 46, pp. 1345–1382, 2005.

[3] V. Badrinarayanan, A. Kendall, et R. Cipolla, « SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, » *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[4] O. Bernard, A. Lalande, C. Zotti, F. Cervenansky, X. Yang, P.-A. Heng *et al.*, « Deep Learning Techniques for Automatic MRI Cardiac Multi-Structures Segmentation and Diagnosis : Is the Problem Solved ? » *IEEE Transactions on Medical Imaging*, vol. 37, no. 11, pp. 2514–2525, 2018.

[5] C. F. Baumgartner, K. C. Tezcan, K. Chaitanya, A. M. Hötker, U. J. Muehlematter, K. Schawkat, A. S. Becker, O. Donati, et E. Konukoglu, « PHiSeg : Capturing Uncertainty in Medical Image Segmentation, » dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*. Cham : Springer International Publishing, 2019, pp. 119–127.

[6] C. Corbière, N. Thome, A. Bar-Hen, M. Cord, et P. Pérez, « Addressing Failure Prediction by Learning Model Confidence, » dans *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 2902–2913.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, et L. Fei-Fei, « Imagenet : A large-scale hierarchical image database, » dans *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[8] 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.

[9] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, et Y. Bengio, « Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, » *NIPS*, vol. 27, 06 2014.

[10] T. DeVries et G. W. Taylor, « Learning Confidence for Out-of-Distribution Detection in Neural Networks, » *arXiv preprint arXiv :1802.04865*, 2018, manuscrit non publié.

[11] A. Degerli, M. Zabihi, S. Kiranyaz, T. Hamid, R. Mazhar, R. Hamila, et M. Gabbouj, « Early Detection of Myocardial Infarction in Low-Quality Echocardiography, » *IEEE Access*, vol. 9, pp. 34 442–34 453, 2021.

[12] Y. Gal et Z. Ghahramani, « Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning, » dans *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, série ICML'16. JMLR.org, 2016, p. 1050–1059.

[13] C. Guo, G. Pleiss, Y. Sun, et K. Q. Weinberger, « On Calibration of Modern Neural Networks, » dans *Proceedings of the 34th International Conference on Machine Learning*, série Proceedings of Machine Learning Research, D. Precup et Y. W. Teh, éditeurs, vol. 70. PMLR, 06–11 Aug 2017, pp. 1321–1330.

[14] J. E. Hall et M. E. Hall, *Guyton and Hall Textbook of Medical Physiology , 14th Edition*. Elsevier, 2021.

[15] G. Huang, Z. Liu, L. van der Maaten, et K. Q. Weinberger, « Densely Connected Convolutional Networks, » dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[16] P. Hoskins, K. Martin, et A. Thrush, *Diagnostic Ultrasound : Physics and Equipment*. Cambridge University Press, 2010.

[17] 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)*, 2016, pp. 770–778.

[18] S. Jaeger, S. Candemir, S. Antani, Y.-X. J. Wáng, P.-X. Lu, et G. Thoma, « Two public chest X-ray datasets for computer-aided screening of pulmonary diseases, » *Quantitative Imaging in Medicine and Surgery*, vol. 4, p. 475, 2014.

[19] A. Jungo et M. Reyes, « Assessing Reliability and Challenges of Uncertainty Estimations for Medical Image Segmentation, » dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, et A. Khan, éditeurs. Cham : Springer International Publishing, 2019, pp. 48–56.

[20] D. P. Kingma et J. Ba, « Adam : A Method for Stochastic Optimization, » dans *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio et Y. LeCun, éditeurs, 2015.

[21] A. Kendall et Y. Gal, « What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ? » dans *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett, éditeurs, vol. 30. Curran Associates, Inc., 2017, pp. 5574–5584.

[22] A. Kendall et Y. Gal, « What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ? » dans *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett, éditeurs, vol. 30. Curran Associates, Inc., 2017.

[23] A. Kosaraju1, A. Goyal, Y. Grigorova, et A. N. Makaryus, « Left Ventricular Ejection Fraction, » 1 mai 2022. Dans National Center for Biotechnology Information. Disponible à https://www.ncbi.nlm.nih.gov/books/NBK459131/.

[24] S. Kohl, B. Romera-Paredes, C. Meyer, J. De Fauw, J. R. Ledsam, K. Maier-Hein, S. M. A. Eslami, D. Jimenez Rezende, et O. Ronneberger, « A Probabilistic U-Net for Segmentation of Ambiguous Images, » dans *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[25] A. Krizhevsky, I. Sutskever, et G. E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks, » dans *Advances in Neural Information*

Processing Systems, F. Pereira, C. Burges, L. Bottou, et K. Weinberger, éditeurs. Curran Associates, Inc.

[26] D. P. Kingma et M. Welling, « Auto-Encoding Variational Bayes, » dans *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[27] Y. Lecun, L. Bottou, Y. Bengio, et P. Haffner, « Gradient-based learning applied to document recognition, » *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] B. Lakshminarayanan, A. Pritzel, et C. Blundell, « Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, » dans *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett, éditeurs, vol. 30. Curran Associates, Inc., 2017.

[29] J. Long, E. Shelhamer, et T. Darrell, « Fully convolutional networks for semantic segmentation, » dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[30] S. Leclerc, E. Smistad, J. Pedrosa, A. Østvik, F. Cervenansky, F. Espinosa *et al.*, « Deep Learning for Segmentation Using an Open Large-Scale Dataset in 2D Echocardiography, » *IEEE Transactions on Medical Imaging*, vol. 38, no. 9, pp. 2198–2210, 2019.

[31] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, « Rectifier nonlinearities improve neural network acoustic models, » dans *Proc. icml*, vol. 30, no. 1. Atlanta, Georgia, USA, 2013, p. 3.

[32] K. V. Mardia et P. E. Jupp, *Directional Statistics*. Wiley, 1999.

[33] F. Milletari, N. Navab, et S. Ahmadi, « V-Net : Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation, » dans *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.

[34] H. Noh, S. Hong, et B. Han, « Learning Deconvolution Network for Semantic Segmentation, » dans *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520–1528.

[35] A. Niculescu-Mizil et R. Caruana, « Predicting good probabilities with supervised learning, » dans *2005 International Conference on Machine Learning (ICML*, 2005, p. 625–632.

[36] O. Oktay, E. Ferrante, K. Kamnitsas, M. Heinrich, W. Bai, J. Caballero *et al.*, « Anatomically Constrained Neural Networks (ACNNs) : Application to Cardiac Image Enhancement and Segmentation, » *IEEE Transactions on Medical Imaging*, vol. 37, no. 2, pp. 384–395, 2018.

[37] N. Painchaud *et al.*, « Cardiac Segmentation With Strong Anatomical Guarantees, » *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3703–3713, 2020.

[38] A. Paszke, A. Chaurasia, S. Kim, et E. Culurciello, « ENet : A Deep Neural Network Architecture for Real-Time Semantic Segmentation, » *CoRR*, vol. abs/1606.02147, 2016, manuscrit non publié.

[39] M. Pakdaman Naeini, G. Cooper, et M. Hauskrecht, « Obtaining Well Calibrated Probabilities Using Bayesian Binning, » *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015.

[40] N. Painchaud, Y. Skandarani, T. Judge, O. Bernard, A. Lalande, et P.-M. Jodoin, « Cardiac MRI Segmentation with Strong Anatomical Guarantees, » dans *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, et A. Khan, éditeurs. Cham : Springer International Publishing, 2019, pp. 632–640.

[41] N. Painchaud, Y. Skandarani, T. Judge, O. Bernard, A. Lalande, et P.-M. Jodoin, « Cardiac Segmentation With Strong Anatomical Guarantees, » *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3703–3713, 2020.

[42] O. Ronneberger, P. Fischer, et T. Brox, « U-Net : Convolutional Networks for Biomedical Image Segmentation, » dans *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 234–241.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, et I. Sutskever, « Learning Transferable Visual Models From Natural Language Supervision, » *CoRR*, vol. abs/2103.00020, 2021, manuscrit non publié.

[44] F. Rosenblatt, « The perceptron : A probabilistic model for information storage and organization in the brain, » *Psychological Review*, vol. 65, pp. 386–408, 11.

[45] J. Shiraishi *et al.*, « Development of a digital image database for chest radiographs with and without a lung nodule : receiver operating characteristic analysis of radiologists' detection of pulmonary nodules, » *American Journal of Roentgenology*, vol. 174, pp. 71–74, 2000.

[46] B. Settles, « Active Learning Literature Survey, » University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[47] 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.

[48] K. Simonyan et A. Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition, » dans *The 3rd International Conference on Learning Representations (ICLR2015)*.

[49] C. C. Taylor, « Automatic bandwidth selection for circular density estimation, » *Computational Statistics and Data Analysis*, vol. 52, no. 7, pp. 3493–3500, 2008.

[50] T. Tieleman et G. Hinton, « Lecture 6.5-rmsprop : divide the gradient by a running average of its recent magnitude. » *COURSERA : Neural networks for machine learning*, vol. 4, no. 2, p. 26–31, 2012.

[51] « Chapter 1 - Introduction, » dans *Diagnostic Ultrasound Imaging : Inside Out (Second Edition)*, T. L. Szabo, éditeur. Boston : Academic Press, 2014, pp. 1–37.

[52] « Interactive deep learning method for segmenting moving objects, » *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017, scene Background Modeling and Initialization.

[53] C. Wiggers, *Circulation in Health and Disease*. Lea & Febiger., 1905.

[54] C. Zotti, O. Humbert, A. Lalande, et P.-M. Jodoin, « GridNet with automatic shape prior registration for automatic MRI cardiac segmentation, » *MICCAI - ACDC Challenge*, 2017.