ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
Université du Québec
Le génie pour l'industrie

MTI881 - Rapport du challenge

Apprentissage profond : Segmentation du myocarde

Réalisé par :

OLLIVIER Emma

VIERA Baptiste

WALIGORA Paul

LEVESQUE Steve

Hiver 2023

Génie de Technologies de l'information

Ecole de technologie supérieur

Table des matières

1 Introduction **2**
1.1 Motivation . . . 2
1.2 Description du "Challenge" . . . 2
1.3 Contributions . . . 3

2 Revue de littérature **5**
2.1 Les transformeurs . . . 5
2.2 Les transformeurs pour la vision . . . 7
2.3 Le modèle Medformer . . . 7

3 Méthodologie **9**
3.1 L'utilisation du modèle . . . 9
3.1.1 L'apprentissage supervisé . . . 9
3.1.2 L'apprentissage semi-supervisé . . . 9
3.2 La fonction de coût . . . 10
3.2.1 La fonction de coût Entropie-croisée . . . 10
3.2.2 La fonction de coût Dice . . . 10
3.2.3 La consistance . . . 11

4 Paramètres expérimentaux **12**
4.1 Hyperparamètres . . . 12
4.1.1 L'entraînement global . . . 12
4.1.2 L'optimisateur (ADAM) . . . 12
4.1.3 Fonctions de coût (DSC + CE) . . . 12
4.2 Optimisateur . . . 13
4.2.1 Points forts de l'optimisateur ADAM . . . 13
4.2.2 Fonctionnement de l'algorithme . . . 13
4.3 Évaluation . . . 14

5 Résultats **16**
5.1 Résultats des différents modèles implémentés . . . 16
5.2 Résultats du modèle final . . . 17
5.2.1 Résultats en terme de métrique . . . 17
5.2.2 Résultats visuels de la segmentation . . . 18

6 Conclusion **20**
6.1 Résultat du challenge . . . 20
6.2 Perspectives . . . 20

7 Bibliographie **21**

1 Introduction

1.1 Motivation

Utiliser l'intelligence artificielle au niveau de la vision par ordinateur pour faire de la segmentation d'images dans le domaine médicale est d'un intérêt plus fort que jamais. Ceci peut s'avérer complexe lorsque des experts peuvent faire le même travail de manière trivial. Par contre, il y a d'importants avantages à déléguer cette tâche fastidieuse à la machine. Premièrement, segmenter des parties d'organes est l'une des tâches les plus répétitives et chronophage qu'un médecin doit faire lors de ses nombreux mandats d'une importante capitale. De plus, l'erreur humaine à cause de la fatigue, notamment suite à la répétition prolongée de la tâche peut mettre à risque la vie des patients respectifs qui s'attendent à avoir une attention hors paire face à leur dossier (i.e. leurs radiographies en cas de blessures/tumeurs).

Sur ce, le domaine de la segmentation en lien avec la vision par ordinateur et la santé est en demande forte pour régler et automatiser cette tâche faiblement gratifiante mais très importante et indispensable pour les hôpitaux.

Le "challenge" va nous permettre d'en apprendre plus sur le sujet en construisant un modèle capable de faire cette tâche bien plus rapidement qu'un humain. Nous utiliserons des bases de données annotés ainsi que non-annotés pour construire un modèle semi-supervisé.

1.2 Description du "Challenge"

Le challenge consiste à détecter et segmenter toutes les classes respectives (tâche décrite dans la motivation). Les données sont séparées de la manière suivante : Une radiographie d'organe peut avoir une étiquette étant une image de type "ground truth" (GT). Un patient peut avoir plusieurs radiographies, mais les pairs d'images et d'étiquettes sont toujours nommées identiquement dans les dossiers respectifs `/img` et `/GT` (i.e. patientXXX_XX_X.png). Il est possible qu'un patient/une radiographie n'ait pas d'étiquette accompagnée. Le dossier respectif est nommé `/Img-Unlabeled`.

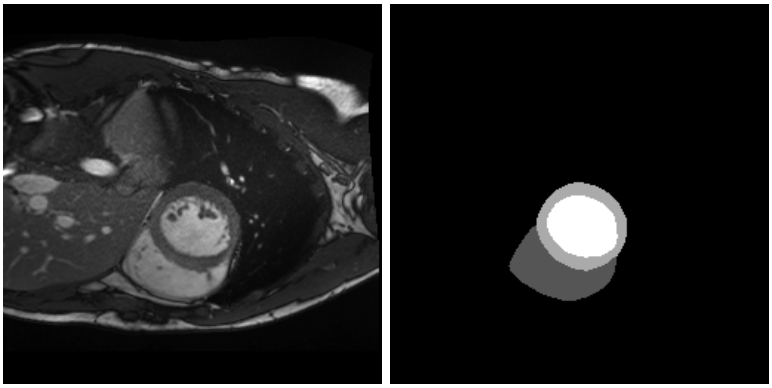

Figure 1 – Exemple d'une radiographie pour "patient078_01_3". La radiographie à gauche et le "ground truth" à droite.

Dans ce challenge, nous avons trois classes à segmenter, le ventricule gauche, qui est ici la première classe, la myocarde, qui est ici la deuxième classe et le ventricule droit qui est la troisième classe. Le reste de l'image est considéré comme étant le fond (le background).

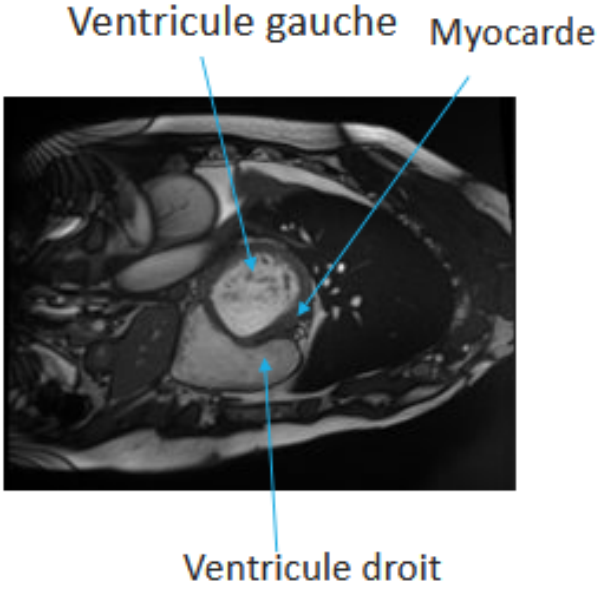

Figure 2 – Image mettant en évidence les différentes classes à segmenter

1.3 Contributions

Le rapport sera divisé en plusieurs parties qui décriront les étapes encourues pour la complétion du "challenge". Il sera question d'une revue de littérature en lien avec l'état de l'art des transformers. Plus précisément pour le domaine de la vision. Une description du modèle utilisé, nommé "MedFormer" fera l'objet de la fin de la revue. Par la suite, la méthodologie regroupera l'utilisation du modèle

choisi pour l'apprentissage supervisé et non-supervisé, la fonction de coût ainsi que la consistance. Les paramètres expérimentaux auront pour but de décrire les hyperparamètres utilisés (autant pour le modèle que l'entraînement et les autres aspects comme la fonction de coût, l'optimisateur) ainsi que l'évaluation au niveau de la topologie de la séparation des données et de la méthodologie d'évaluation des courbes obtenues. Finalement, les résultats vont concrètement dévoiler les performances du modèle par rapport aux étapes précédentes.

Une conclusion brève nous permettra de discuter par rapport à nos réflexions suite aux résultats du "challenge" en plus des possibles travaux futurs qui pourraient être faits pour améliorer notre modèle.

2 Revue de littérature

2.1 Les transformeurs

Parmi l'ensemble des réseaux de neurones, nous distinguons les "Reccurent Neural Network" (RNN) qui constituent un tremplin pour le réseau Long Short-Term Memory (LSTM) et les transformeurs. La spécificité des RNN est leur capacité à prendre en compte les données séquentielles. En d'autres termes, les RNN utilisent la sortie précédente comme entrée actuelle. Cela permet de capturer des relations à assez long terme. Cependant, les RNN sont associés à quelques problèmes. En effet, ils sont très lents au point d'utiliser une version tronquée de la rétro-propagation pour les entraîner ce qui peut demander beaucoup de ressources computationnel. Un autre problème de taille, est la faible capacité à gérer les longues séquences engendrées par la disparition et l'explosion du gradient. Dans ce contexte, nous pouvons introduire le réseau LSTM. Le concept de "long-term memory" permet d'éviter une grande partie de traitement de la cellule actuelle et de passer à la suivante. Cela permet alors de traiter de plus longues séquences bien que celles-ci ne soient limitées à une centaine de mots. À noter que les LSTM sont encore plus lents que les RNN du fait de leur complexité. Tout comme dans les RNN, dans les LSTM, les données d'entrée sont introduites les unes après les autres de manière séquentielle. De plus, pour faire le traitement de l'état actuel, il est nécessaire d'avoir la sortie de l'état précédent. Ces opérations, ne se faisant pas de façon parallèle, n'exploitent pas toutes les capacités des GPUs, ce qui en fait une limite. C'est alors qu'en 2017, l'article « Attention is all you need » a été publié et a introduit la notion des transformeurs qui exploitent quant à eux très largement le calcul parallèle. Les transformeurs sont basés sur une architecture encodeur/décodeur qui est d'une certaine manière similaire aux RNN mais se différencient par cette capacité à transmettre en parallèle la séquence d'entrée. En d'autres termes, si nous prenons une phrase, chaque mot de la phrase sera traité simultanément. Il est intéressant de mentionner que cette architecture des transformeurs présentés dans l'article s'applique plus précisément au traitement du langage comme par exemple pour la traduction d'une phrase. A noter que les encodeurs et décodeurs ne comprennent pas les « mots » en tant que tel. C'est pour cette raison qu'il est nécessaire de transformer les mots en vecteurs. Pour se faire, un mot est associé à un vecteur présent dans un espace dans lequel les mots sémantiquement similaires sont proches physiquement dans l'espace. Par exemple, le vecteur associé au mot « chat » sera proche de celui associé au mot « chien » ou encore celui du mot « rouge » sera proche du mot « bleu ». Cette espace se nomme en anglais « embedding space ». Il existe des espaces déjà pré-entrainés que nous pouvons utiliser. Cependant, il est probable qu'un même mot apparaisse à plusieurs endroits dans une même phrase, ce qui peut changer plus ou moins sensiblement son interprétation. C'est pour cette raison, qu'est présent dans l'architecture d'un transformeur les encodeurs positionnels qui sont des vecteurs qui informent sur la distance des mots

dans une phrase. Le bloc encodeur intégrera donc en entrée des vecteurs contenant non seulement des informations sur les mots eux-mêmes mais aussi sur leur position formant un contexte. Ces vecteurs seront alors les entrées d'un « feed forward network » qui permettra de générer trois vecteurs en sortie appelés « Query », « Key », « Value » dans une couche nommée « Multi-Head Selft Attention ». Le concept d'attention est central dans les transformeurs. Elle peut être définie de manière très simplifiée comme la mesure de la relation entre deux entités. A l'intérieur de cette couche d'attention, des opérations matricielles sont réalisées afin d'obtenir à l'aide d'un « softmax », une distribution probabilistique qui représente la quantité d'attention, c'était à dire de relation de chaque mot d'une phrase, deux à deux. Cette opération est réalisée plusieurs fois comme l'indique le terme « Multi-Head » afin de capturer le maximum d'informations et de contexte pour chaque mot. Les couches d'attentions étant indépendantes, les calculs peuvent être réalisés en parallèles. Une opération de concaténation est ensuite réalisée entre les différentes sorties.

Il est également intéressant de préciser que dans la couche de l'encodeur, étant donné que le réseau de neurones (le transformeur) est profond, des connections résiduelles sont utilisées afin d'éviter le problème de la disparation du gradient (vanishing gradient), c'est-à-dire le non apprentissage. De plus, afin d'éviter des valeurs trop larges en magnitude, le réseau présente une couche de normalisation qui peut être fonction des caractéristiques. Concernant le bloc décodeur, dans un contexte de traduction d'une phrase de l'anglais au français par exemple, nous transmettons à celui-ci la phrase française, et plus précisément des vecteurs pour chaque mot qui contiennent des informations contextuelles comme expliqué précédemment. En simplifiant un peu le processus, ces vecteurs sont traités par un bloc d'attention qui se différencie du bloc d'attention de l'encodeur. En effet, dans ce premier bloc d'attention (situé dans le décodeur), celui-ci est associé à un masque qui empêche la possibilité de connaître le mot suivant. Uniquement le mot précédent est connu. En effet, il n'y aurait aucun intérêt à procéder à ces séries d'opérations, si le réseau avait accès au mot suivant étant donné qu'il s'agit de l'élément que l'on souhaite prédire, il n'y aurait aucun apprentissage. Il existe un autre bloc d'attention dans le décodeur qui se différencie des deux autres blocs d'attention précédents qui étaient en mode « self ». Les relations entre les mots sont déterminées sur la phrase elle-même, qu'elle soit en anglais ou en français. Dans ce nouveau bloc d'attention, l'entrée de celui-ci est composée de la sortie du premier bloc d'attention du décodeur et de la sortie du bloc d'attention de l'encodeur. Il s'agit du « Multi-Head Cross Attention ». C'est à cette étape que la relation et donc la mesure d'attention entre des paires de mots anglais et français a lieu. En simplifiant, après avoir passé la sortie de la couche d'attention dans un « feed forward network » puis la sortie du décodeur également dans « un feed forward network », est obtenu une distribution des probabilités après avoir appliqué « softmax ». Le mot suivant qui sera prédit est celui qui est associé à la plus grande probabilité. Ce process est répété jusqu'à atteindre la fin de la phrase à traduire.

2.2 Les transformeurs pour la vision

Maintenant que nous avons expliqué de façon haut niveau les transformeurs pour le traitement du langage présenté dans l'article « Attention is all you need », nous allons présenter brièvement les transformeur pour la vision (ViT) qui ont été introduit en 2019 avec l'article « An image is worth 16x16 words : tranformers for image recognition at scale ». Comme expliqué précédemment, dans le cadre du traitement du langage, chaque mot associé à des vecteurs est comparé deux à deux afin de déterminer un contexte général. Dans une image, nous pourrions alors considérer un pixel comme étant un mot. Cependant, il n'est pas envisageable d'appliquer ce principe sur une image dans sa globalité puisqu'elle peut contenir des centaines de milliers de pixels jusqu'à des millions de pixels pour les images à haute résolution. De plus, il est important de préciser que la complexité associée à la couche d'attention évoquée précédemment possède une complexité quadratique, ce qui rend les calculs impossibles. C'est pour cette raison que l'article a introduit la notion de patch. L'objectif est de diviser l'image en plusieurs patchs de dimension 16x16. Cependant à l'instar d'une phrase, le transformeur ne sait pas à quelle partie de l'image le patch se situe. C'est pour cela que nous devons ajouter des vecteurs portant des informations sur la position. Chaque patch 16x16 « aplatis » va subir une projection linéaire puis les vecteurs de sortie seront associés au vecteur contenant l'information de la position. Les vecteurs résultants joueront alors le rôle d'entrée de l'encoder de notre transformeur dont l'architecture est identique à celui présenté précédemment. La couche d'attention dans l'encoder permettra alors de connaître les relations entre les patchs. Un élément important sur lequel nous reviendrons dans la section suivante est la capacité des transformeurs à mesurer l'attention c'est-à-dire des relations à longue portée dès le début de l'entraînement, à l'inverse des CNNs qui ont une attention très locale, principalement au début. Enfin, la sortie de cet encoder sera l'entrée d'un perceptron multi-couche dans le cas d'une classification multi-classe. En plus du champ récepteur local des neurones dans les CNNs, il paraît important de présenter une autre différence que les CNNs ont avec les transformeur tout comme le réseau LTSM possède pour le traitement du langage. Les CNNs possèdent un bon biais inductif à l'inverse des transformeurs qui sont sans biais. Par exemple dans le cadre des CNNs, si nous prenons un pixel, celui-ci se souciera de son voisin immédiat qui se souciera lui-même de son voisin immédiat et ainsi de suite. Nous connectons les régions proches entre elles. Cela est engendré par le fait que les CNNs sont associés à un estimateur biaisé. Ils performeront généralement mieux sur des faibles jeux de données. A l'inverse, les transformeurs qui sont des modèles sans biais performeront mieux sur de très grands volumes de données.

2.3 Le modèle Medformer

Avant de présenter brièvement le modèle Medformer que nous avons choisi, nous pouvons mentionner à nouveau les limites associées aux transformeurs. Dans

le domaine médical, nous pouvons identifier des défis comme l'acquisition des données, le coût des annotations ou encore la variété des maladies. Dans ce contexte, les tranformeurs sans biais auront des difficultés à généraliser sur un large éventail de tâches médicales même s'ils sont pré-entrainés sur ImageNet. A cela s'ajoute le fait que la complexité de la couche d'attention par rapport à la longueur de la séquence d'entrée est quadratique. Cela engendre des coûts computationnels très élevés, notamment pour les images à haute définition. Enfin, une limite associée aux ViT est le processus de diviser une image en patch dont le processus a été expliqué dans la section précédente. En engendrant la perte de l'information structurelle et détaillée, ce sous-échantillonnage pose naturellement problème à la tâche de segmentation qui se résume par la classification de chaque pixel de l'image. Medformer n'ayant pas besoin d'utiliser des poids pré-entrainé et en utilisant une architecture hybride un peu à l'instar de la famille des modèles hybrides CoAtNets, palie à ces différentes limites. Medformer introduit un biais inductif au travers d'une convolution séparable en profondeur dans le réseau de projection. Cette convolution est présente dans les couches peu profondes. Cela permet de mieux gérer la capture des caractéristiques et des textures locales. De plus, cela favorise également l'analyse des données à petite échelle. Medformer utilise ensuite dans ses couches intermédiaires et profondes un bloc transformeur afin d'améliorer les relations locales mais aussi à longue portée. Dans ce bloc transformeur a été introduit une nouvelle architecture d'attention nommée B-MHA (Bidrectional Multi-Head Attention) qui permet d'éliminer les tokens redondants avec une projection de faible rang, réduit la complexité quadratique à une complexité linéaire, de déterminer à longue portée des relations et ceux même pour les images à haute définition. À noter qu'il existe d'autres méthodes permettant de réduire la complexité, comme celle introduite par le réseau SwinTransformer (« Local window self-attention ») ou encore celle introduite par le réseau CCNet (« Decomposing attention »). Cependant, ces deux techniques sont associées à des inconvénients comme l'altération de la capacité de ces réseaux de déterminer directement des relations à longue portée. Il est également intéressant de préciser qu'il existe d'autres modèles basés sur les transformeurs spécialisés dans la segmentation médicale comme TransUNET ou UNETR qui ajoutent à l'architecture ViT (présentée dans la section précédente) un décodeur convolutionnel. D'autres modèles comme SwinUNet ou VTUNet sont basés sur SwimTransformer ou encore d'autres sont quant à eux basés sur des modèles hydrides comme SwinUNETR. Cependant, tous ces modèles sont limités étant donné qu'ils doivent être pré-entrainés sur de très grand jeux de données afin d'obtenir de bonnes performances.

3 Méthodologie

3.1 L'utilisation du modèle

3.1.1 L'apprentissage supervisé

La première étape de l'apprentissage de notre modèle s'est fait uniquement avec le jeu d'entraînement des données étiquetés, c'est à dire celles pour lesquelles nous possédons le "ground truth". La procédure mise en place a été la suivante. Après chaque epoch, il fallait valider la pertinence du modèle obtenu avec le jeu de données de validation, puis après dix epochs, il a été constaté que les résultats de performance en terme d'accuracy n'étaient pas meilleurs que ceux obtenus à l'epochs précédent. Et si nous continuions l'apprentissage malgré tout, une performance légèrement inférieure était même observable, il s'agit du phénomène de sur-apprentissage. Nous avons donc enregistré le modèle obtenu au bout de 10 epochs pour lequel la précision mesurée est de 96%.

3.1.2 L'apprentissage semi-supervisé

A partir du modèle enregistré, nous avons à présent voulu utiliser les données non supervisées. Pour cela, la méthode dite de l'élève et du professeur (student-teacher) a été utilisée. Cette méthode consiste à dupliquer un réseau de neurones puis à faire des prédictions avec l'un à partir des données non étiquetées classiques et avec l'autre à partir des données non étiquetées auxquelles un bruit a été rajouté. L'apprentissage se fait en minimisant la différence entre les prédictions faites avec les données bruitées et les prédictions faites avec les données non bruitées.

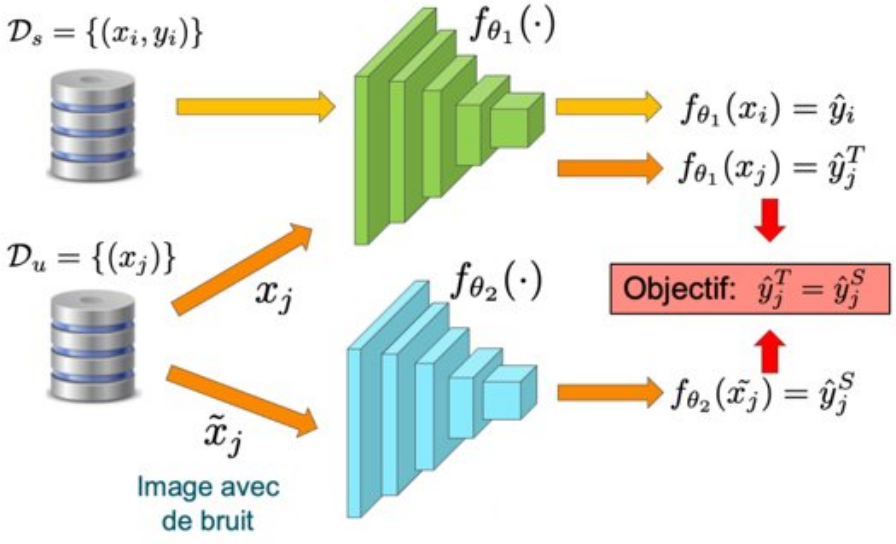

Figure 3 – Schéma expliquant le fonctionnement de la méthode professeur-élève pour les jeux de données semi-supervisées

Ici, seul le réseau de neurones recevant les images non bruitées subit une rétropropagation pour la mise à jour de ses poids. L'autre réseau de neurones est mis à jour avec ce qui est appelé l'"exponential moving average" (EMA), qui prend en compte les valeurs de ses poids et celles de l'autre réseau de neurones (celui sur lequel est appliqué la rétro-propagation).

$$\theta'_t = \alpha * \theta'_{t-1} + (1 - \alpha) * \theta_t$$

$$\alpha = min(1 - \frac{1}{n+1}, \alpha)$$

avec
θ'_t : nouveau paramètre du réseau qui ne fait pas la rétropropagation
θ'_{t-1} : ancien paramètre du réseau qui ne fait pas la rétropropagation
θ'_t : paramètre actuel du réseau qui fait la rétropropagation

Dans notre cas, nous sommes parties directement avec le modèle obtenu avec les données supervisées. De la même manière qu'avec les données supervisées, il a été nécessaire d'arrêter l'apprentissage une fois que les performances obtenues avec le jeu de données de validation à une epoch donnés étaient moins pertinentes que celles de l'epoch précédente dans le but de ne pas "tomber dans le sur-apprentissage" (overfitting). L'entraînement a ici été arrêté au bout de cinq epochs, avec une précision (accuracy) de 98%.

3.2 La fonction de coût

3.2.1 La fonction de coût Entropie-croisée

Afin que l'apprentissage du modèle se déroule correctement, nous avons besoin d'une fonction de coût à minimiser. Dans un premier temps nous nous sommes tournés vers l'entropie croisée qui est une fonction de coût assez classique et souvent utilisé. Malheureusement, bien qu'ayant, tout d'abord, l'impression d'avoir de bons résultats en terme d'accuracy avec celle-ci nous nous sommes aperçu que les modèles obtenu ainsi classifient en fait tous les pixels comme appartenant au fond (au "background"). Cela s'explique en effet par le fait que comparativement au fond les différentes zones à segmenter sont plutôt minoritaire, d'autant plus que certaines images ne sont en réalité constituées que d'un fond. Ainsi en prédisant tous les points comme appartenant au fond on obtient une accuracy de 97% mais cela n'est bien évidemment pas pertinent.

3.2.2 La fonction de coût Dice

Il a alors été nécessaire de se tourner vers une autre fonction de coût que l'entropie croisée. Nous avons alors tenté d'utiliser la fonction de coût Dice. Dans notre cas, ayant trois zones à segmenter, il fallait implémenter une fonction de Dice capable de gérer plusieurs classes simultanément, nous en avons alors utilisé une

développée par Iakubovskii. Toutefois en utilisant seulement la fonction de coût Dice, l'accuracy peinait à atteindre des valeurs intéressantes lors de la phase de validation. Nous avons alors eu l'idée de combiner par une somme la fonction de coût Dice et l'entropie croisé, cette dernière obtenant quant à elle un bon score d'accuracy. Grâce à cette nouvelle fonction de coût créée, les résultats de performance en terme d'accuracy et de dice sont bien meilleur. Il reste alors à trouver le terme de régularisation, entre les deux éléments additionnés, le plus avantageux. Après plusieurs essaies avec le jeu de validation nous sommes arrivés à la conclusion que la fonction de coût optimal pour l'apprentissage impliquant uniquement les données étiquetées est la suivante :

$$Loss = \frac{3}{4} * DiceLoss + \frac{1}{4} * CrossEntropy$$

En parallèle de la combinaison de la fonction de coût loss et de l'entropie croisée, d'autres méthodes ont été envisagées comme l'utilisation de la fonction de coût focal, mais les résultats étaient moins intéressants.

3.2.3 La consistance

Lors de la phase où les données non étiquetées ont été utilisées, il a fallu redéfinir la fonction de coût. Avec la méthode de l'élève et du professeur (student-teacher), le but est de minimiser la différence entre les prédictions obtenus à partir des données non supervisées non modifiées et des données non supervisées pour lesquelles du bruit a été ajouté. Cette minimisation est traduite par une fonction de coût dite de consistance, prenant dans notre cas la forme suivante :

$$ConsistencyLoss = ||yj^T - yj^S||^2$$

Dans notre approche, nous utilisons en parallèle les données étiquetées et non étiquetées, il s'agit donc là encore de trouver le bon terme de régularisation entre la fonction de coût pour l'apprentissage avec les données étiquetées et la fonction de coût de consistance. Il a alors fallu refaire plusieurs essais avec le jeu de données de validation pour parvenir au meilleur résultat. Ainsi, la fonction de coût obtenu est la suivante :

$$Loss = \frac{4}{7} * LabeledDataLoss + \frac{3}{7} * ConsistencyLoss$$

4 Paramètres expérimentaux

4.1 Hyperparamètres

Les hyperparamètres sont les coefficients reliés aux composantes du modèle d'apprentissage machine, notamment l'algorithme, l'optimisateur, des sections de la fonction de coût si applicable, etc. Ceux-ci peuvent être changés avant le début de l'entraînement pour potentiellement obtenir de meilleurs résultats.

4.1.1 L'entraînement global

`epoch`
Les epochs correspondent au nombre d'itérations que l'on choisit pour obtenir la convergence souhaité. Pour un epoch l'ensemble du jeu de données d'entraînement a été parcouru. Dans notre cas le modèle présenté résulte de **15** epochs répartis entre le supervisé et le non supervisé.

`batch_size`
Le "batch size" correspond à la grosseur des lots pour chaque epoch. Pour la phase d'entraînement notre taille de lot est de **15** pour la phase d'entraînement est de **1** pour la phase de validation.

4.1.2 L'optimisateur (ADAM)

`learning_rate`
Le taux d'apprentissage correspond en quelque sorte à la grandeur du pas utilisé pour faire les mises à jour des paramètres lors de l'apprentissage. Il est important que celui-ci ne soit pas trop petit pour que la convergence ne soit pas trop lente, ni trop grand pour éviter de diverger. Nous avons choisi un taux d'apprentissage de **0.001** en début d'apprentissage puis nous avons réduit à **0.0001** à la fin.

4.1.3 Fonctions de coût (DSC + CE)

La fonction de coût pour le supervisé
Notre fonction de coût pour l'apprentissage supervisé est la somme d'une fonction de coût dice et d'une fonction de coût d'entropie croisée. Il est alors nécessaire d'avoir des hyperparamètres λ_1 et λ_2 qui permettent d'accorder une importance spécifique pour la fonction de coût dice et la l'entropie croisée de manière à maximiser l'efficience.

$$Loss = \lambda_1 * DSC + \lambda_2 * CE$$

Dans notre cas $\lambda_1 = \frac{3}{4}$ et $\lambda_2 = \frac{1}{4}$, la pertinence de ces hyperparamètres a été trouvé grâce aux données du jeu de validation.

La fonction de coût totale
La fonction de coût totale prend à la fois en compte les données étiquetées et les données non étiquetées. Elle a été fait en prenant en compte la consistance. Là également il faut avoir deux hyperparamètres λ_3 et λ_4 afin de choisir l'importance donnée à chaque composante de la fonction de coût.

$$Loss = \lambda_3 * LabeledDataLoss + \lambda_4 * ConsistencyLoss$$

De la même manière qu'avec la fonction de coût des données étiquetées, les hyperparamètres sont trouvées avec le jeu de données de validation, et nous avons choisi les suivant : $\lambda_3 = \frac{4}{7}$ et $\lambda_4 = \frac{3}{7}$

4.2 Optimisateur

L'optimisateur de choix par l'équipe pour accomplir la tâche est ADAM. Celui-ci est couramment utilisé pour entraîner des réseaux de neurones profonds. Il combine le Stochastic Gradient Descent (SGD) et le Momentum résultant en une version améliorée : "Adaptatitive Moment Estimation Optimizer" (ADAM).

4.2.1 Points forts de l'optimisateur ADAM

- Il permet de trouver un minimum global de manière très efficace.

- Il est populaire pour les réseaux de neurones profonds et les modèles en général en raison de sa capacité à ajuster efficacement la taille de pas de mise à jour des poids.

- Il peut converger plus rapidement et plus efficacement vers un minimum global grâce à son estimation de premier et deuxième ordre ainsi que sa stabilisation pour éviter les arrêts directs (avant même qu'il ait la chance de commencer à trouver un minimum).

4.2.2 Fonctionnement de l'algorithme

ADAM utilise l'estimation adaptative du moment du premier ordre consistant à faire la moyenne mobile exponentielle des gradients ainsi que l'estimation du moment du deuxième ordre en lien avec les carrés des gradients pour ajuster la taille de la mise à jours des poids lors de la descente du gradient stochastique.

Ceci permet d'éviter potentiellement les minimums locaux que le gradient normal aurait fort probablement détecté comme une solution viable. Aussi, l'estimation adaptative permet de donner un mouvement plus fluide, rapide et "diagonale"

contrairement aux version plus ancienne qui sont bien entendue plus performante que le gradient normal, mais ayant une direction plus linéaire et "rectangulaire".

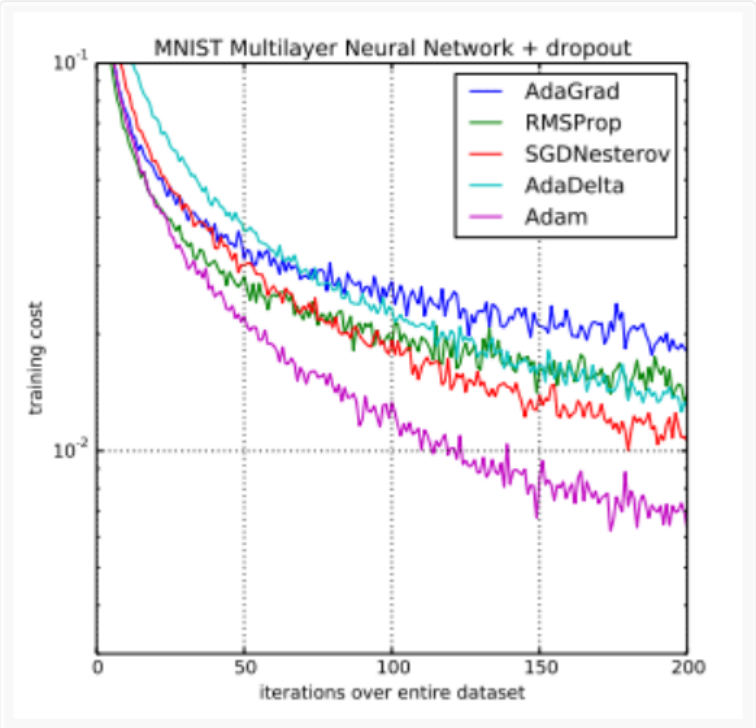

Figure 4 – Comparaison de la méthode ADAM avec d'autre méthode d'optimisation sur la tâche de classification pour le jeu de données MNIST [9]

Le schéma ci-dessus met bien en lumière la pertinence de la méthode ADAM par rapport aux autres méthode et c'est pour cela que nous avons choisi d'entraîner notre modèle avec celle-ci.

4.3 Évaluation

- **Séparation de la base de données** :

La séparation de la base de données est faite respectivement en lien avec les données étiquetées ainsi que les données non étiquetées. Voici la séparation proposée de manière plus précise :

– 204 images étiquetées pour l'entraînement

– 1004 images non étiquetées pour l'entraînement

– 74 images étiquetées pour la validation

Ceci va nous permettre d'avoir une courbe avec les données d'entraînement et de validation. Parfait pour l'évaluation décrite plus tard dans cette section.

- **Méthodologie d'évaluation au niveau de la progression de l'entraînement** :

– Utilisation des ressources et convergence du modèle : Basé sur le code fournit au cours, il est possible de sauvegarder uniquement le modèle (i.e. les poids) lorsqu'il est meilleur que son précédent. Avec cette information, ainsi que les statistiques obtenues après chaque epoch, il est possible de détecter lorsque l'entraînement arrive à conclusion si la convergence reste la même et qu'aucune amélioration n'est obtenue après N epoch.

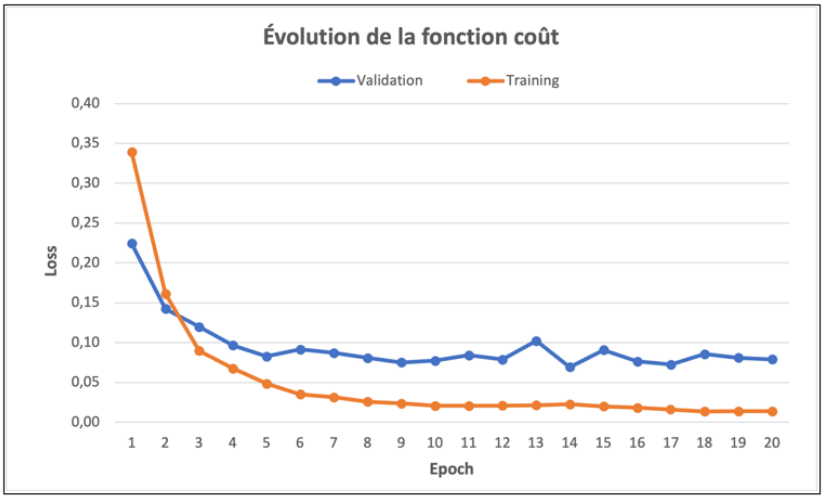

Figure 5 – Schéma représentant un exemple d'entraînement et de validation obtenu avec notre modèle

– Courbe de validation : Suite à la figure ci-dessus, il est possible d'avoir un jugement critique au niveau de l'évolution de la fonction de coût lors de l'entraînement envers la base de données d'entraînement et validation. Une manière d'y arriver serait d'utiliser notre figure 4 pour y faire un diagnostique par rapport à l'epoch qui est la plus performante.

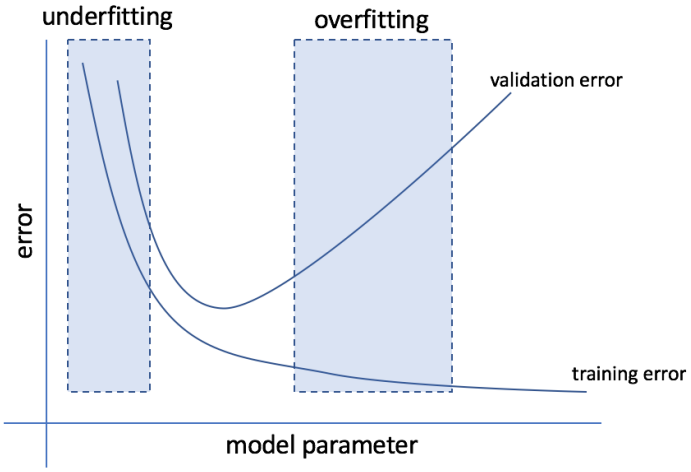

Figure 6 – Graphique montrant le sous apprentissage et le sur apprentissage sur l'ensemble de données d'apprentissage et de validation [1]

Comme on peut le voir sur la figure ci-dessus, il est pertinent d'avoir le modèle le plus au centre pour éviter le sous apprentissage et le sur apprentissage. Respectivement, on évitera ainsi le biais et la variance forte. Dans le même ordre d'idées, nous voulons aussi un taux d'erreur d'entraînement qui se rapproche le plus possible de la validation puisque cela signifie que l'entraînement est juste. La partie gauche représente un niveau de biais haut puisque la performance est nulle et l'apprentissage n'est pas significatif. La partie droite représente une variance élevée puisque l'erreur de validation est haute, ce qui implique qu'on apprend trop des données d'entraînement et que l'on va avoir des problèmes à généraliser par la suite.

5 Résultats

5.1 Résultats des différents modèles implémentés

Avant de choisir le modèle présenté ici, nous en avons testé d'autres qui se sont révélés moins performant. Pour ce qui est de la fonction de coût nous avons d'abord essayé avec l'entropie croisé et avons obtenue une accuracy de 97% mais le modèle prédisait l'ensemble des pixels comme appartenant au "background". Nous avons également utilisé une fonction de coût focal mais l'accuracy était seulement de 53% ce qui n'était vraiment pas bon. Toutefois il se peut que ce mauvais résultat vienne d'une mauvaise implémentation de cette fonction. Enfin nous avons utilisé la fonction de coût utilisant à la fois la fonction de coût dice et la cross entropie et avec

celle-ci nous avons obtenu une accuracy de 93%. Les résultats précédents sont données en utilisant un réseau de neurone de type "SwinNet" qui est un type de réseau de neurones pour la segmentation dérivant de "Unet".

Après avoir obtenu ces résultats avec SwinNet, nous avons implémenter le réseau de neurones MedFormer. Et avec la fonction de coût prenant en compte le dice et l'entropie croisée nous obtenons une accuracy de 98%. Nous avons donc gardé cette fonction de coût (à laquelle il a été ajouté la consistance) pour notre modèle final ainsi que le réseau de neurone de type MedFormer. Il faut aussi noter que les résultats pertinents étaient obtenus beaucoup plus rapidement avec MedFormer qu'avec Swinet.

5.2 Résultats du modèle final

5.2.1 Résultats en terme de métrique

Pour mesurer la performance du modèle de segmentation final obtenu nous nous sommes appuyés sur plusieurs métriques. Tout d'abord avec le jeu de données de validation nous obtenons une accuracy de 98% et un dice de 93%, ce qui est tout de même correct bien qu'il y ait une grosse marge d'amélioration. Toutefois si nous entrons un peu plus dans le détail des différentes métriques que nous pouvons obtenir par classe, on peut réaliser que les performances ne sont pas réellement homogènes.

Table 1 – Tableau mettant en lumière la précision, le recall et l'accuracy pour les trois classes à segment

	Précision	Recall	Accuracy
classe 1	93%	95%	99%
classe 2	59%	93%	99%
classe 3	41%	97%	99%

On remarque alors que l'accuracy et le recall sont plutôt bons pour les trois classes mais que la vraie lacune de ce modèle est la précision. En effet, bien que correcte pour la première classe, elle n'est que de 41% pour la troisième classe. En d'autres termes, pour interpréter ces résultats, un pixel classifié réellement dans la troisième classe sera très certainement classifié comme tel par le modèle. Mais d'autre part, si un pixel est classifié comme étant de la classe 3, il se peut qu'il soit en réalité issu d'une autre classe ou du fond (background).

Etiquette \ Prédiction	classe 0	classe 1	classe 2	classe 3
classe 0	4612852	2893	27544	52311
classe 1	527	62667	2947	0
classe 2	503	2012	45766	986
classe 3	339	7	853	37457

Figure 7 – Matrice de confusion obtenue à partir de la validation du modèle

Lors du calcul de la matrice de confusion, le même phénomène est observable. Le modèle ne produit pas beaucoup de faux négatifs, mise à par peut être légèrement pour le fond qui représente par la classe 0 mais qui a tendance à prédire beaucoup de faux positifs principalement pour la deuxième et la troisième classe. On peut constater que c'est principalement les pixels du fond qui sont prédits à tort comme étant de la troisième classe.

5.2.2 Résultats visuels de la segmentation

Si on regarde visuellement les résultats de notre modèle, on constate en effet que celui-ci a beaucoup de difficulté à segmenter la troisième classe, c'est à dire le ventricule droit. Et comme cela a pu être observé avec l'analyse des différentes métriques et de la matrice de confusion, le modèle aura tendance à segmenter une zone plus large que la réalité pour le ventricule droit. Toutefois, dans l'ensemble les autres classes semblent être relativement bien segmentées.

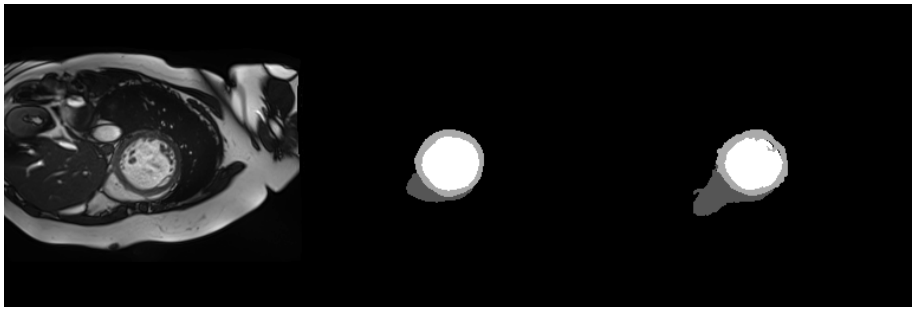

Figure 8 – Résultat de la segmentation pour l'image "patient012_01_4" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite)

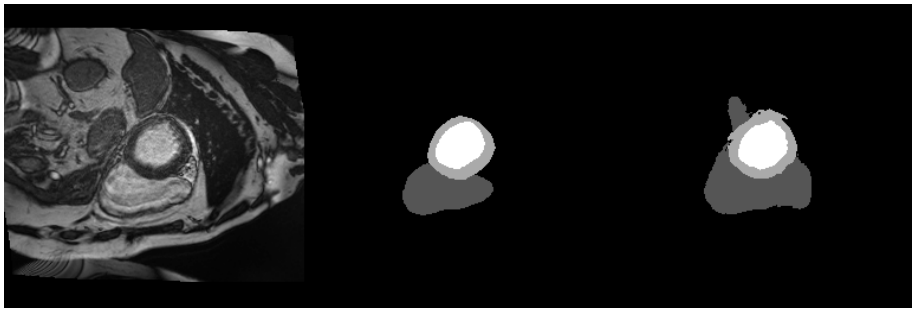

Figure 9 – Résultat de la segmentation pour l'image "patient052_01_1" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite)

Nous pouvons également noter que lorsqu'une image ne possède aucune des classes à segmenter, le modèle aura tendance à vouloir, malgré tout, classifier des pixels comme appartenant à ces classes. Cela explique aussi que le modèle a de mauvais résultats en terme de précision.

Figure 10 – Résultat de la segmentation pour l'image "patient061_10_9" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite)

6 Conclusion

6.1 Résultat du challenge

Après avoir obtenu les résultats obtenu par notre modèle avec le jeu de données de test, il s'est avéré que nous sommes arrivés troisième au challenge, ce qui démontre que notre modèle a encore une grande marge de progression. Bien que certains éléments tels que l'architecture Medformer pour le réseau de neurones ou encore la fonction de coût utilisant à la fois la fonction de coût dice et l'entropie croisées nous on permis d'effectuer un apprentissage correct au niveau de l'accuracy, les mauvais résultats au niveau de la précision ont dû nous faire perdre beaucoup de points.

6.2 Perspectives

Plusieurs perspectives d'amélioration peuvent par conséquent être envisagées. Il serait par exemple pertinent de pouvoir faire des tests avec plus d'hyperparamètres. Pour cela, il serait pas exemple possible d'utiliser un "Grid Search", bien que cela soit tout de même très coûteux en terme de ressource. On pourrait également utiliser une méthode de validation croisée, ce qui pourrait permettre une meilleur performance lors de l'étape de validation. D'autre part, il pourrait être aussi nécessaire de se concentrer d'avantage sur les transformations possibles qui pourraient être effectuées sur les images étiquetées afin d'augmenter de manière artificielle leur quantité. Enfin nous n'avons pas normalisé les images dans notre étape d'apprentissage, il serait alors pertinent de recommencer celui-ci en les normalisant.

7 Bibliographie

[1] Jordan, J., Evaluating a Machine Learning model
https ://www.jeremyjordan.me/evaluating-a-machine-learning-model/

[2] HILAB-GIT. SSL4MIS : Self-supervised learning for medical image segmentation. Github Repository, 2021. https ://github.com/HILAB-GIT/SSL4MIS.

[3] Qubvel. Segmentation Models Pytorch. Github Repository, 2021.
https ://github.com/qubvel/segmentation_models.pytorch

[4] Dolz, J (2023), cours de MTI881, ETS Montreal

[5] Zhang, Y., Tian, Y., Kong, X., Liu, B., Fu, Y. "A Data-scalable Transformer for Medical Image Segmentation : Architecture, Model Efficiency, and Benchmark." IEEE Transactions on Medical Imaging, vol. 40, no. 3, pp. 1223-1232, 2021.

[6] Gao, Y. CBIM-Medical-Image-Segmentation : Convolutional Block Attention Module for Medical Image Segmentation. Github Repository, 2021.
https ://github.com/yhygao/CBIM-Medical-Image-Segmentation.

[7] Wikipedia, Stochastic Gradient Descent ADAM,
https ://en.wikipedia.org/wiki/Stochastic_gradient_descentAdam

[8] Great Learning, What is Cross Validation in Machine learning ? Types of Cross Validation, https ://www.mygreatlearning.com/blog/cross-validation/

[9] Machine Learning Mastery, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, https ://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Table des figures

1 Exemple d'une radiographie pour "patient078_01_3". La radiographie à gauche et le "ground truth" à droite. 3
2 Image mettant en évidence les différentes classes à segmenter 3
3 Schéma expliquant le fonctionnement de la méthode professeur-élève pour les jeux de données semi-supervisées 9
4 Comparaison de la méthode ADAM avec d'autre méthode d'optimisation sur la tâche de classification pour le jeu de données MNIST [9] . . 14
5 Schéma représentant un exemple d'entraînement et de validation obtenu avec notre modèle 15
6 Graphique montrant le sous apprentissage et le sur apprentissage sur l'ensemble de données d'apprentissage et de validation [1] 16
7 Matrice de confusion obtenue à partir de la validation du modèle . . . 18
8 Résultat de la segmentation pour l'image "patient012_01_4" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite) 18
9 Résultat de la segmentation pour l'image "patient052_01_1" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite) 19
10 Résultat de la segmentation pour l'image "patient061_10_9" - l'image (à gauche), le groundtruth (au milieu), la segmentation obtenue (à droite) 19