

RAPPORT PROJET FIN D'ETUDE

Détection Automatique d'IRM pour la Dégénérescence Discale des Brebis

Projet réalisé par

Qianben CHEN

Runming ZHANG

Yang YANG

Projet encadré par

Diana MATEUS

Marion FUSELLIER

Nora BOUHSINA

REMERCIEMENTS

Tout d'abord, nous tenons à remercier tout particulièrement et à témoigner toute notre reconnaissance aux personnes suivantes, pour leur dévouement et leur soutien dans la concrétisation de ce projet ingénieur:

- Mme. Diana MATEUS, tutrice projet, pour ses conseils éclairés, sa patience, sa disponibilité et pour la confiance qu'il nous a accordée dès l'ébauche du projet et tout au long de ces cinq mois et avoir mis à notre disposition tous les moyens disponibles.
- Mme. Marion FUSELLIER, collaboratrice projet, pour sa coopération professionnelle tout au long de cette expérience.
- Mme. Nora BOUHSINA, collaboratrice projet, pour son investissement et sa contribution et pour avoir partagé avec nous, une partie de son savoir-faire et de ses expériences professionnelles.
- M. Said MOUSSAOUI, responsable de l'option DATASIM, pour toute l'aide et les conseils apportés durant ce projet.

SOMMAIRE

I. INTRODUCTION	5
II. BESOINS ET OBJECTIFS DU PROJET.....	5
a) Contexte	5
b) Faisabilité	5
c) Objectifs	6
1. Les objectifs techniques	6
2. Les objectifs économiques	6
3. Les délais	6
III. DEVELOPPEMENT TECHNIQUE.....	6
a) Détection des Patches	7
i. Préparation des données	7
ii. Méthode Traditionnelle	8
1. Modèle	8
2. Entraînement	9
3. Résultats.....	9
iii. Méthode CNN	10
1. Pré-traitement des Données	10
2. Modèle	11
3. Entraînement	13
4. Résultats.....	13
iv. Conclusion	16
b) Segmentation des Régions d'Intérêt	17
i. Préparation des données	17
ii. Méthode Traditionnelle	18
1. Modèle	18
2. Entraînement	18
3. Résultats.....	19
iii. Méthode U-net	20
1. Pré-traitement des Données	20
2. Modèle	20

3. Entraînement	21
4. Résultats	23
a) Première Partie	23
b) Deuxième Partie	24
iv. Conclusion	26
c) Recalage	26
IV. BILAN DU PROJET	28
a) Analyse des Résultats	28
b) Conclusion	29
c) Lien Externe	30
V. BIBLIOGRAPHIE	30

I. INTRODUCTION

Dans le cadre de notre seconde année à DATASIM, il nous est proposé un projet de 5 mois qui nous permet de mettre en pratique nos connaissances et nos compétences professionnelles au travers d'un cahier des charges ayant pour finalité la conception et le développement d'une application industrielle en accord avec nos intérêts professionnels.

Ayant une passion commune pour le traitement des images médicales, notre groupe composé de Qianben CHEN, Runming ZHANG et Yang YANG a saisi l'opportunité d'exploiter cet intérêt commun pour soumettre l'ébauche d'un projet professionnel.

II. BESOINS ET OBJECTIFS DU PROJET

a) Contexte

Le titre de notre projet s'appelle « Développement d'un outil d'analyse automatique d'images IRM quantitatives dans l'évaluation de la dégénérescence discale lombaire chez la brebis ».

La lombalgie discogène est une préoccupation majeure de santé publique fréquemment associée à la maladie dégénérative du disque intervertébral lombaire (DDD). De nombreux outils et modèles animaux ont été utilisés pour nous aider à améliorer notre compréhension de la physiopathologie DDD et à développer des méthodes d'imagerie pour la détecter le plus tôt possible. La brebis s'est avéré être un modèle animal de grande valeur grâce aux similitudes (anatomie globale, propriétés mécaniques) de son disque intervertébral lombaire (DIV) avec celles du DIV lombaire humain. En parallèle, l'imagerie par résonance magnétique quantitative (IRM) semble être un outil cliniquement pertinent pour explorer les DDD précoces, en particulier les mesures du temps de relaxation $T2^*$.

Nous voulons développer un programme d'analyse automatique d'images IRM basé sur une base de données d'images de brebis acquis lors d'une étude préliminaire manuelle. En particulier, les méthodes développées contribueront à la segmentation du disque dans la base de données.

b) Faisabilité

Nous avons constaté que de nombreuses personnes ont utilisé l'apprentissage automatique et l'apprentissage en profondeur pour étudier et traiter des images IRM sur la colonne vertébrale. À l'heure actuelle, les scientifiques ont réussi à atteindre classification et classement de l'IRM sur la colonne vertébrale automatiquement.

Nous avons lu beaucoup d'articles dans ce domaine et nous nous sommes beaucoup inspirés d'eux. En plus d'utiliser les idées d'apprentissage en profondeur, le projet qui nous convient le mieux est l'algorithme HOG [1].

c) Objectifs

i. Les objectifs techniques

Étant pour le moment dans un projet universitaire qui est limité en moyens et en temps et a pour objectif principal de mettre en pratique nos connaissances et nos compétences professionnelles, nous avons décidé de restreindre notre projet domotique, en sélectionnant les solutions à développer parmi toutes les possibilités permises.

ii. Les objectifs économiques

Un des enjeux du projet est économique. Un prix grand est nécessaire si la détection de la dégénérescence discale des brebis doit être fait de manière manuelle. C'est pourquoi nous avons décidé d'étudier à résoudre ce problème de manière automatique.

iii. Les délais

Le projet débute depuis le novembre 2019 et s'achève à la fin du Mars, soit un peu plus de cinq mois. Afin de terminer ce projet ambitieux à temps, il est important de correctement le gérer et de le tenir à jour grâce aux outils de gestion adéquats.

Dans cette optique, nous avons initialement planifié tous les travaux et procédé comme prévu. Et afin de mieux vérifier ce que nous avons perdu et les rattraper, nous maintenons toujours un contact avec notre tuteur et les collaborateurs. Pour les problèmes mineurs quotidiens, nous les contactons par mails. Pour l'avancement d'une phase, nous tenons plusieurs réunions avec les collaborateurs et le tuteur, et également organisé des réunions conjointes avec tout le monde.

III. DEVELOPPEMENT TECHNIQUE

Ce projet étant éducatif, notre première mission fut de définir nous-mêmes un pipeline stratégique prévisionnelle ainsi que les objectifs à atteindre. Bien entendu, ce pipeline a évolué au cours du temps afin de satisfaire nos exigences, mais aussi les contraintes auxquelles nous avons fait face. Éventuellement, nous fixons le pipeline à trois étapes principales : **Détection des Patches**, **Segmentation des Régions d'Intérêt** et **Recalage**.

La première étape se focalise dans les images complètes (qui sont de type T1 SAG et de taille 512x512) et les patches où se trouvent les régions auxquelles nous nous intéressons sont trouvés.

La deuxième étape met les patches trouvés auparavant en entrée et a pour objectif de segmenter les trois régions d'intérêt à l'intérieur des patches.

Par rapport à chaque étape, nous proposons deux types de méthodes : l'un utilise les techniques traditionnelles et l'autre utilise les techniques sur l'apprentissage profond. Ensuite, on compare leur effet quantitatif et qualitatif afin de décider finalement un pipeline le plus efficace selon nos besoins.

Après la localisation des régions d'intérêt, nous faisons ensuite un outil automatique pour la recherche de la valeur moyenne dans les trois régions. Nous avons un tableau de critère pour prédire automatiquement à quel niveau de la dégénérescence discale les valeurs correspondent.

a) Détection des Patches

i. Préparation des données

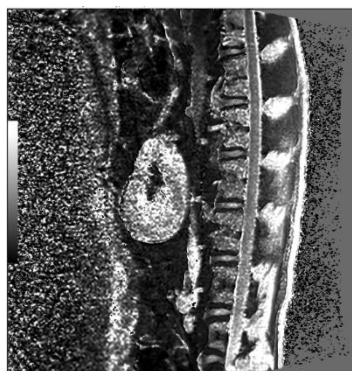
Dans cette partie, nous avons les images sur les disques des brebis données par l'École Nationale Vétérinaire de Nantes. Les images contiennent 15 brebis, dont chacun est mesuré pendant 5 jours (2 brebis parmi eux ont 4 jours). Ainsi, nous obtenons $13 \times 5 + 2 \times 4 = 73$ images au total. De plus, les types des images sont T1 SAG, T2 SAG, T1, T2 et T2*, d'où les images de T1 SAG et T2 SAG sont faciles à opérer mais les images de T1, T2 et T2* ont des valeurs intéressantes. C'est pourquoi nous opérons tout d'abord sur les images pour trouver les régions dans les images SAG et faisons ensuite le recalage afin de localiser les positions des régions dans les images intéressantes pour trouver les valeurs. Nous visualisons la différence des images :



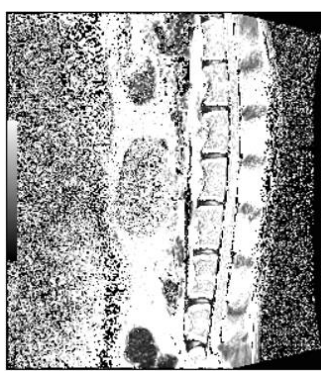
T1 SAG



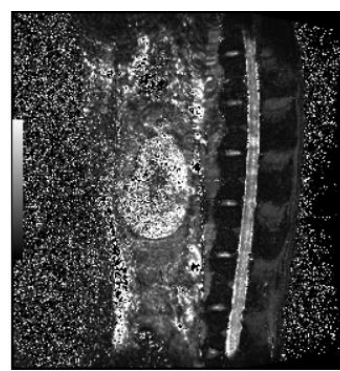
T2 SAG



T1



T2



T2*

Figure 1 Différence des images de différents types

Donc, nous concentrons sur les images SAG afin de trouver les patches. Afin de faire la détection automatique, il faut tout d'abord obtenir manuellement certains patches dans les images SAG qui aident nos modèles à apprendre les caractéristiques des patches. Nous les visualisons comme ci-dessous :

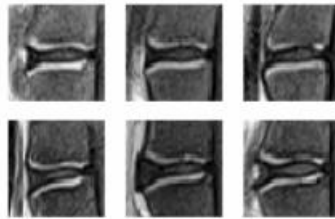


Figure 2 Exemples des patches qui nous intéressent

ii. Méthode Traditionnelle

1. Modèle

Après l'étude sur le pré-traitement des images et sur les papiers publiés, nous voyons la possibilité d'utiliser des méthodes traditionnelles pour détecter des patches. Donc, nous proposons 3 étapes pour la détection des patches:

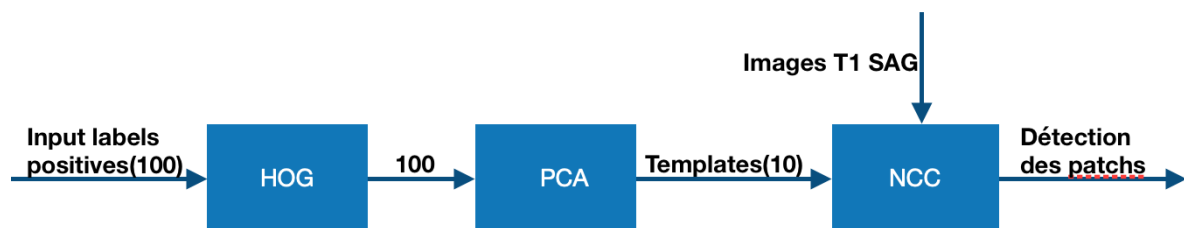


Figure 3 Étapes pour la détection des patches

- *HOG (Histogram of Oriented Gradients):*
 - *Entrée : 100 patches déjà trouvés manuellement*
 - *Sortie : 100 patches de forme d'histogramme orienté*
 - Cette technique peut aider à représenter les caractéristiques des patches dans l'image, afin que le modèle puisse détecter de tels patches.
- *PCA (Principal Component Analysis):*
 - *Entrée : 100 patches de forme d'histogramme orienté*
 - *Sortie : 10 composantes principales*
 - Cette technique peut réduire les dimensions des données et trouver les caractéristiques qui contiennent la plus grande quantité d'informations.
- *NCC (Normalized Cross Correlation):*

- *Entrée : 10 composantes principales*
- *Sortie : des patches détectés*
- Cette technique peut aider à faire des jugements de similitude en calculant la covariance de deux images. Si l'image a un sous-ensemble qui correspond aux autres données d'échantillon, sa valeur NCC est 1, indiquant une corrélation élevée, et si elle est -1, elle est complètement hors de propos.

2. Entraînement

➤ *HOG (Histogram of Oriented Gradients):*

Nous choisissons la taille de chaque cellule est 2×2 , puis les labels de taille 36×36 est divisées en 18×18 cellules et nous définissons 2×2 cellules dans chaque bloc. Les histogrammes de gradient de toutes les cellules à l'intérieur de chaque bloc sont combinés et normalisés. Les caractéristiques de tous les blocs sont combinées pour former la fonction HOG finale.

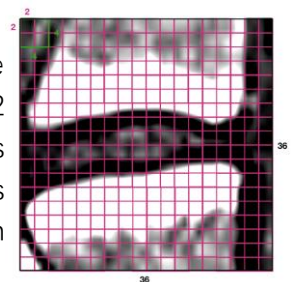


Figure 4 HOG

➤ *PCA (Principal Component Analysis):*

Nous faisons PCA sur les 100 labels positives après le traitement HOG et nous définissons le nombre des caractéristiques sur 10. Nous obtenons donc 10 images contenant le plus d'informations.

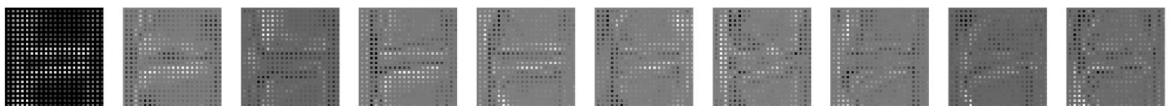


Figure 4 10 labels positifs après le traitement HOG et PCA

➤ *NCC (Normalized Cross Correlation):*

Nous allons parcourir les dix images obtenues après PCA sur les données de validation, afin de trouver la région avec la plus forte corrélation avec ces dix images, qui est la région d'intérêt d'abord.

Nous pouvons facilement penser que dans le processus de la détection, il y aura des zones qui se chevauchent, elles représentent un même patch. Par conséquent, nous définissons que lorsque la zone coïncidente est supérieure à 60%, un seul des patches est conservé.

1. Résultats

Selon le pré-traitement des images, les labels positifs sont obtenus de manière manuelle. Nous avons mis leur vérité terrain à 1. Nous comparons nos patches détectés avec des labels positives. Si nos patches détectés est correct, nous les cochons en couleur verte. En revanche, si le résultat est incorrect, nous les cochons en couleur orange.

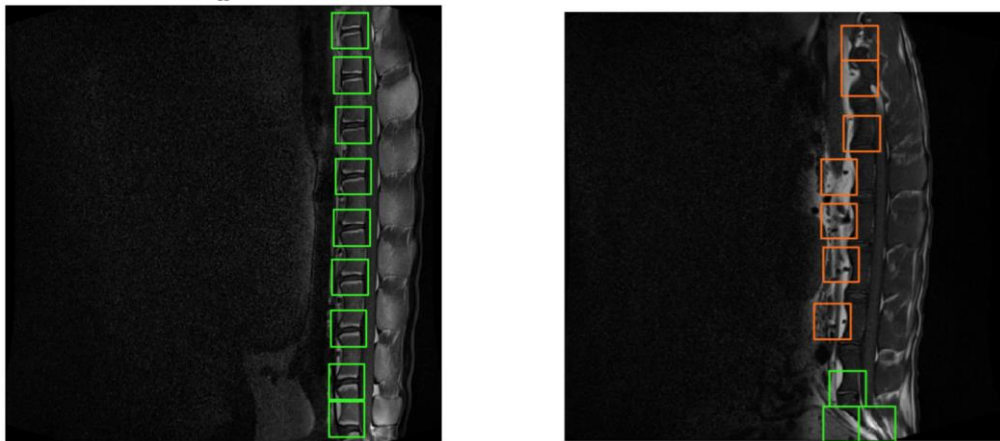


Figure 5 Les régions d'intérêt détectés

Nous avons constaté que pour certaines images, la méthode traditionnelle fonctionne bien et trouve tous les patches, mais pour d'autres images, la méthode ne fonctionne pas bien.

Nous évaluons les performances du modèle :

TP	FP	precision score
470	251	0.65

Tableau 1 Résultat quantitatif

Donc, ce modèle n'est pas assez robuste.

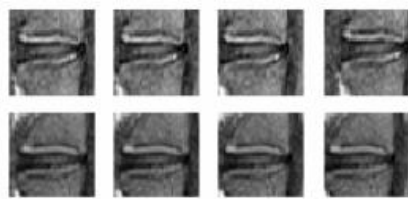
iii. Méthode CNN

1. Pré-traitement des données

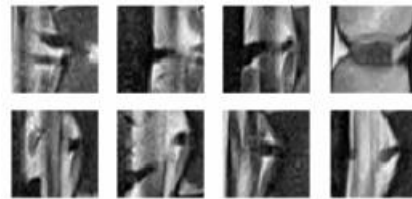
Nous faisons manuellement les labels positifs qui sont des patches dans les images SAG, comme indiqué dans la partie précédente. Cependant, nous devons aussi préparer les labels négatifs qui se divisent en trois groupes : un groupe avec les labels négatifs aléatoires dans l'image complète **(a)**, un groupe avec les labels négatifs autour des labels positifs **(b)**, et un autre groupe avec les labels "négatifs difficiles" **(c)** :

- Les labels positifs sont les labels que nous voulons.
- Les labels négatifs **(a)** aident les modèles à distinguer les labels positifs et les bruits.
- Les labels négatifs **(b)** aident les modèles à se concentrer plutôt au centre des labels positifs
- Les labels négatifs **(c)** aident les modèles à avoir plus de capacité de distinguer les labels positifs et les labels négatifs qui ressemblent aux labels positifs.

Nous visualisons la différence des labels positifs et des labels "négatifs difficiles" :



Exemplaires des labels positifs



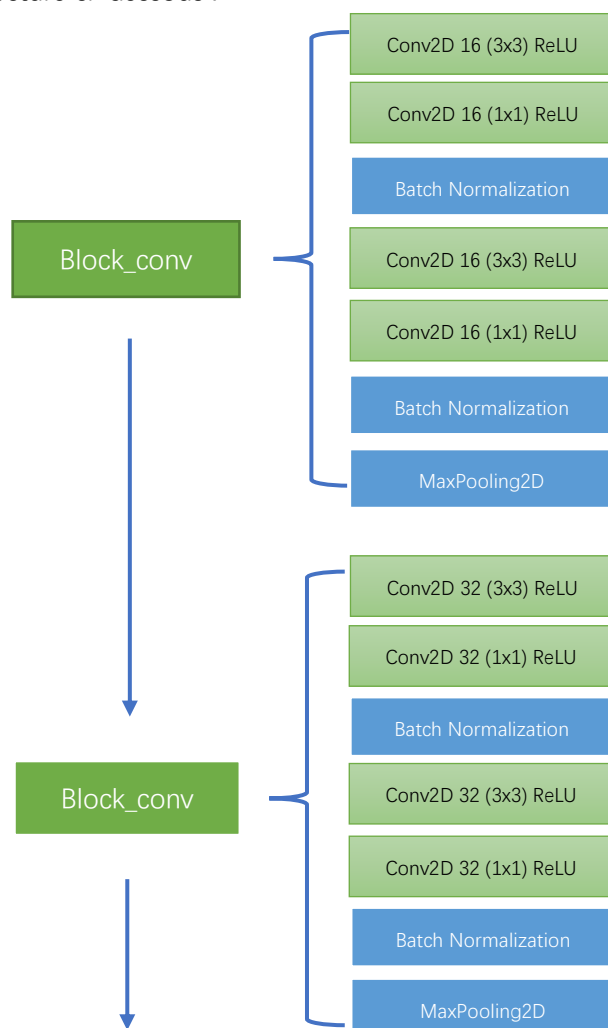
Exemplaires des labels "négatifs difficiles"

Figure 6 Différence des labels positifs et des labels "négatifs difficiles"

2. Modèle

En constatant que la méthode traditionnelle ne permet pas d'avoir un résultat précis, nous nous intéressons ensuite aux méthodes d'apprentissage profond. Nous proposons CNN [3] qui a pour objectif de faire la classification des labels positifs et négatifs (donc le nombre de classes est 2).

L'architecture du CNN se base principalement sur VGG. On utilise l'ensemble de trois blocks composés des couches convolutionnelles, Batch Normalization et Maxpooling et ensuite deux blocks composés des couches denses, Batch Normalization et Dropout. Nous visualisons l'architecture ci-dessous :



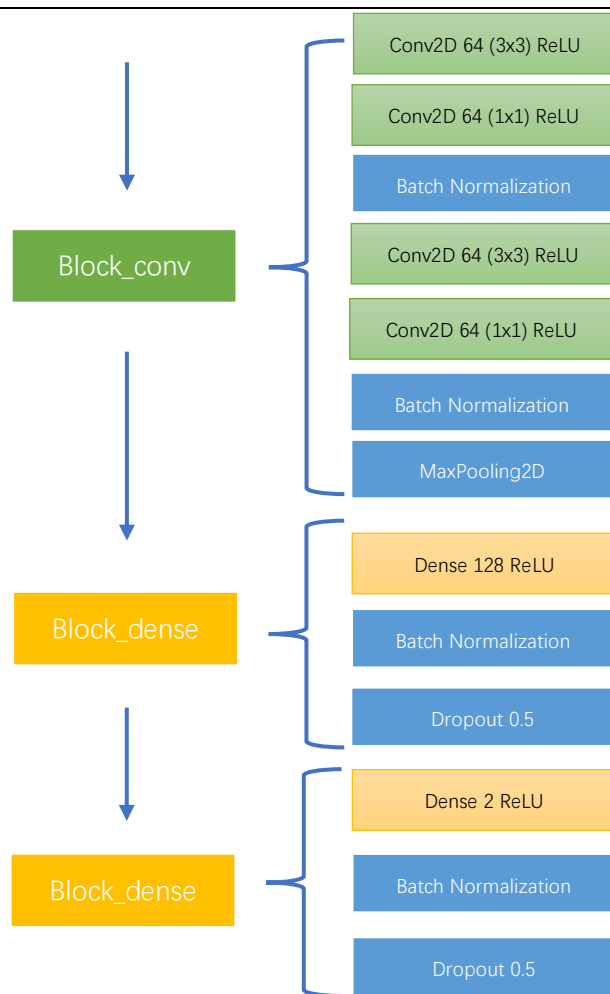


Figure 7 Modèle CNN

Bien entendu, ce modèle a évolué au cours du temps afin d'augmenter la performance non seulement selon nos connaissances mais aussi selon les contraintes auxquelles nous avons fait face. Finalement, nous fixons le modèle et nous utilisons plusieurs techniques qui doivent être soulignées :

- *Batch Normalization* :
Cette technique peut aider à entraîner un modèle plus profond en améliorant la régularisation. Les valeurs des neurones sont normalisées après chaque couche afin de mieux se passer à la suite.
- *Dropout* :
Cette technique peut aussi améliorer la régularisation, mais en amputant une partie de ses neurones pendant la phase d'entraînement et les réactivant pour tester le nouveau modèle.
- *Conv2D avec le noyau de taille 1x1* :
Cette technique peut jouer le rôle de non seulement augmenter la non-linéarité, mais aussi de réajuster les valeurs des neurones en un prix petit, ce

qui permet au modèle d'avoir plus de capacité de passer les informations à la suite.

3. Entraînement

Avant d'entraîner les données, nous réalisons une augmentation des données pour que nous ayons un nombre de données suffisant et que les données sont plus générales. Nous réalisons l'augmentation des données par :

- Coupe aléatoire de taille 42x42 à 36x36
- Renversement horizontal
- Changement de contraste
- Changement de luminance

Donc, on peut obtenir ~50000 patches en entrée au lieu de quelques centaines de patches. De plus, on est capable d'apprendre la robustesse au changement de direction horizontal, de contraste et de luminance.

En ce qui concerne les hyperparamètres de l'entraînement, nous mettons :

- Taux d'apprentissage = 0.001
- Étapes d'entraînement = 4000
- Taille du lot = 64
- Étapes d'affichage = 50

En ce qui concerne la segmentation des données, nous considérons les premières 70% des données comme les données d'entraînement et les dernières 30% des données comme les données de test.

En ce qui concerne l'optimisateur, nous choisissons SGD avec *Moment*. SGD calcule le gradient d'un seul exemple au lieu de calculer le gradient de tous les exemples, ce qui permet d'entraîner le modèle plus vite. Le *Moment* permet de subir une accélération via le gradient, ce qui permet au modèle d'avoir une tendance de converger plus vite. Nous comparons SGD avec Adam et nous concluons que SGD a la capacité de s'entraîner beaucoup plus vite sans perdre de la performance.

4. Résultats

Nous entraînons le modèle et le testons sur les données de test.

Ce qu'il faut souligner, c'est que le modèle et son entraînement ont évolué au cours du temps afin de s'améliorer.

Premier essai : nous établissons un modèle qui:

- est n'est pas profond et n'a pas *Conv2D* avec le noyau de taille 1x1
- n'utilise pas les labels négatifs autour des labels positifs et les labels "négatifs difficiles"

et nous obtenons les résultats comme ci-dessous :

Taux d'exactitude	0.9979
Taux de rappel	1.0000
Taux de précision	0.9959
Matrice de confusion	[[3468 15] [0 3685]]

Tableau 2 Résultat quantitatif du premier essai



Figure 8 Résultat visualisable du **Faux Négatif** du premier essai

Nous constatons que le modèle n'est pas assez capable d'apprendre les caractéristiques des labels positifs.

Donc, (**deuxième essai**) nous rendons le CNN plus profond (Figure 7) avec *Conv2D du noyau de taille 1x1*. Nous obtenons ensuite les résultats comme ci-dessous :

Taux d'exactitude	1.0000
Taux de rappel	1.0000
Taux de précision	1.0000
Matrice de confusion	[[3483 0] [0 3685]]

Tableau 3 Résultat quantitatif du deuxième essai

Nous constatons que ce modèle est assez performant d'apprendre les caractéristiques des labels positifs. Cependant, si nous testons le modèle sur les images complètes, nous visualisons des prédictions fausses parce que les patches dont nous faisons la prédiction ne sont pas dans nos données de tests :

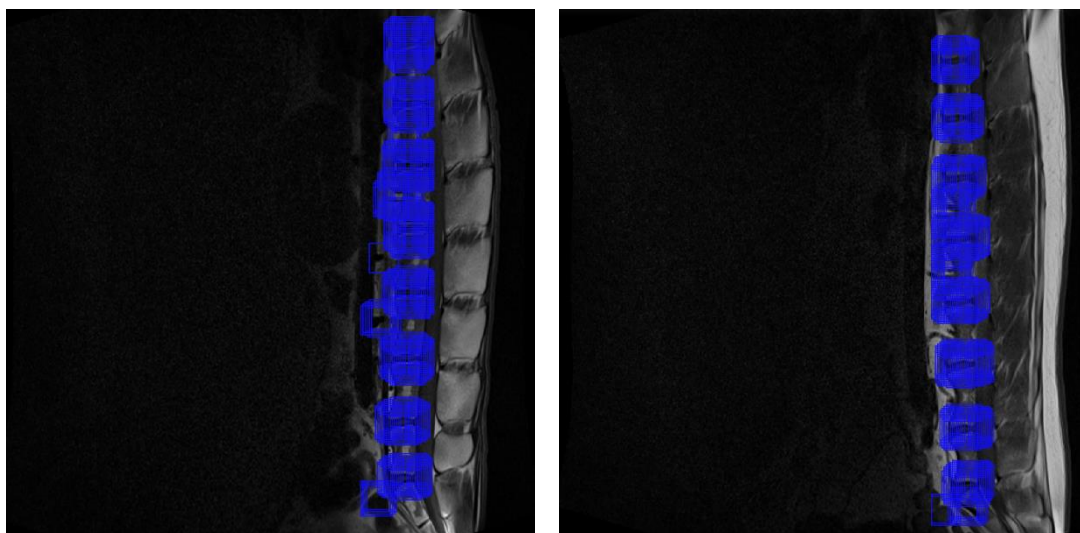


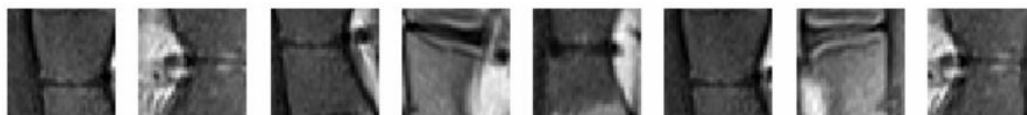
Figure 9 Exemples des prédictions fausses

En visualisant la Figure 9, nous constatons qu'il y a des patchs mal trouvés. Ces patchs sont soit autour des labels positifs soit des labels "négatifs difficiles".

Ensuite, afin de résoudre ce problème, (**troisième essai**) nous ajoutons les labels négatifs autour des labels positifs et les labels "négatifs difficiles" en entrée de l'entraînement pour apprendre les caractéristiques de ces types de patchs. Nous obtenons ensuite les résultats comme ci-dessous :

Taux d'exactitude	0.9994
Taux de rappel	1.0
Taux de précision	0.9989
Matrice de confusion	[[6922 0] [8 6957]]

Tableau 4 Résultat quantitatif du troisième essai

Figure 10 Résultat visualisable du **Faux Positif** du troisième essai

Nous constatons que le modèle est assez capable d'apprendre les caractéristiques des labels positifs et "négatifs difficiles". Même s'il y a des prédictions des négatifs fausses, cela a un nombre beaucoup moins que le nombre total des labels "négatifs difficiles" dans toutes les images, ce qui montre que le modèle est assez performant pour distinguer les labels positifs et les labels négatifs/ "négatifs difficiles". Donc, nous choisissons le modèle du troisième essai.

Afin d'obtenir les patchs intéressants, nous réalisons un regroupement des patchs de prédiction. Éventuellement, Nous visualisons les résultats comme ci-dessous :

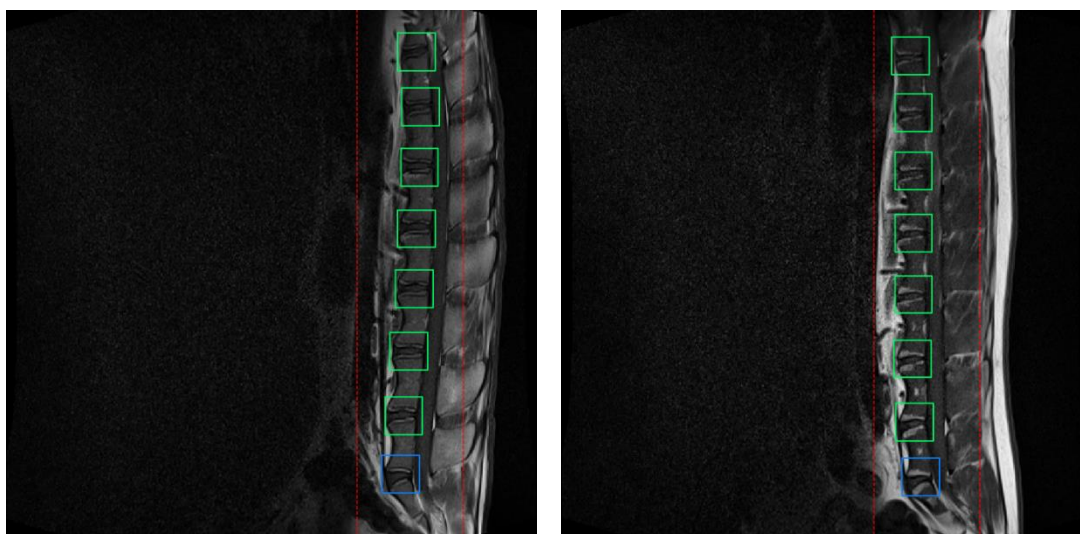


Figure 11 Exemplaires des images complètes

(Les patches bleu signifient que même s'ils sont détectés, ils ne sont pas considérés dans la suite à cause de la forme spécifique. Les deux lignes rouges, détectées par *HOG*, signifient que nous obtenons seulement les patches entre les lignes en entrée du modèle pour réduire le temps de calcul)

Nous constatons que les résultats sont visuellement vrais et robustes. Nous calculons le taux de précision sur toutes les images :

Vrai Positif	Faux Négatif	Taux de précision
513	7	0.9865

Tableau 5 Résultat final présenté en taux de précision

Donc, à partir du dernier essai, nous pouvons détecter automatiquement les patches intéressants, qui sont ensuite les données d'entrée pour la segmentation des régions d'intérêt.

iv. Conclusion

Après avoir comparé la méthode traditionnelle avec la méthode d'apprentissage profond, nous constatons que la méthode traditionnelle est instable, et les résultats sont beaucoup influencés par les données spéciales. Pour la méthode d'apprentissage profond, nous constatons que le modèle est non seulement plus robuste mais aussi plus précis. Le seul inconvénient est le besoin des labels fait manuellement, la précision des labels affecte directement les résultats.

Ensuite, nous arrivons à la deuxième grande partie : Segmentation des Régions d'Intérêt, qui prend en entrée les patches détectés par la première grande partie.

b) Segmentation des Régions d'Intérêt

Dans cette deuxième grande partie, nous avons envie de faire la segmentation des trois régions d'intérêts (voir la Figure 12) à partir des patches détectés. Les valeurs de ces régions réfléchissent le niveau dégénérative de la disque des brebis.

Les trois régions sont dessinées sur les séquences qualitatives :

- ROI A = Anneau fibreux ventral
- ROI B = Noyau Pulpeux
- ROI C = Anneau fibreux dorsal

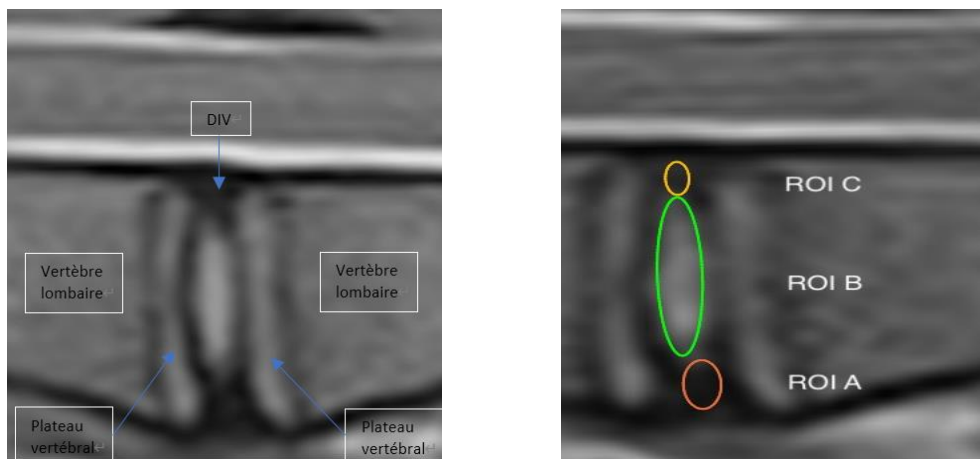


Figure 12 Explication des trois régions d'intérêt

i. Préparation des Données

Dans cette partie, nous avons les patches détectés par la première grande partie. Il y a 520 patches au total qui sont détectées par toutes les images T1 SAG. Tous les patches sont en taille 36x36. Nous visualisons quelques exemplaires des patches détectés:

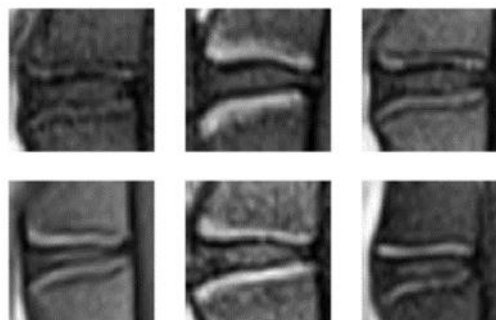


Figure 13 Exemplaires des patches détectés

Nous allons ensuite segmenter les régions dans les patches détectés.

ii. Méthode Traditionnelle

1. Modèle

Après avoir trouvé les patches, nous allons essayer de trouver les régions d'intérêt. Donc, nous proposons 3 étapes pour trouver les trois régions d'intérêts:



Figure 14 Étapes pour la détection des patch

- *Gaussian Blur:*
 - *Entrée: patches détectés*
 - *Sortie : patches moyennés par le noyau gaussien*
 - Cette technique peut aider à éliminer les bruits.
- *HOG (Histogram of Oriented Gradients):*
 - *Entrée: patches moyennés par le noyau gaussien*
 - *Sortie : 4 lignes (2 horizontaux et 2 verticaux) qui indique la zone cible*
 - Cette technique peut aider à trouver le contour de la zone cible afin d'affiner davantage le champ de la classification.
- *GMM clusters:*
 - *Entrée: zone cible délimitée par les 4 lignes*
 - *Sortie : clusters d'où deux sont les régions d'intérêts (centre et côtés)*
 - Cette technique peut approximer en douceur la distribution de densité de n'importe quelle forme. Ainsi, nous pouvons distinguer les différentes parties.

2. Entrainement

Nous avons sélectionné au hasard 6 patches pour les expériences :

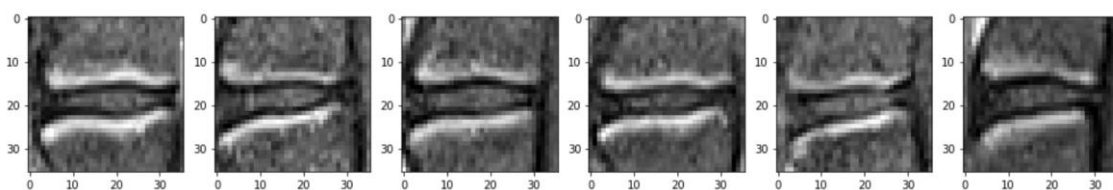


Figure 15 Exemple des 6 patches détectés à partir d'une seule image

- *Gaussian Blur*
- *HOG (Histogram of Oriented Gradients):*

Nous choisissons deux types de cellules pour trouver le contour des ROI. La taille d'une cellule est 36×2 , l'autre est 2×36 , puis les patches de taille 36×36 est divisées en 18 colonnes et 18 rangées. Ainsi, nous définissons 2×2 cellules dans chaque bloc. Les histogrammes de gradient de toutes les cellules à l'intérieur de chaque bloc sont combinés et normalisés. Les caractéristiques de tous les blocs sont combinées pour former la fonction HOG finale. Autrement dit, nous extrayons les caractéristiques horizontales et verticales.

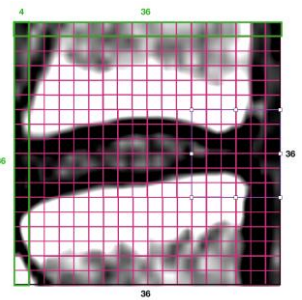


Figure 16 HOG

Nous avons trouvé quatre lignes, et la zone centrale délimitée par les quatre lignes était notre zone cible rétrécie.

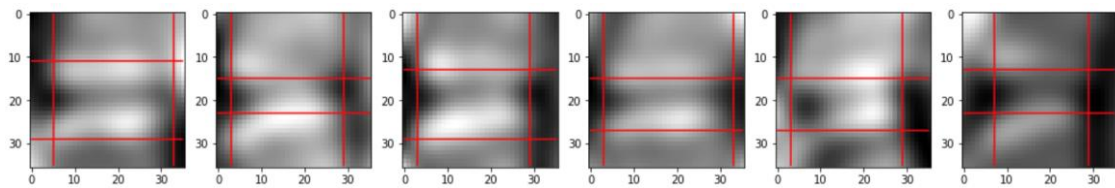


Figure 17 zones cibles trouvées par HOG

➤ *GMM clusters:*

Nous prenons la région réduite en entrée et effectuons gmm clustering. En fait, nous recherchons trois types de ROI, mais ici, nous définissons les catégories à cinq pour obtenir de meilleurs résultats de classement.

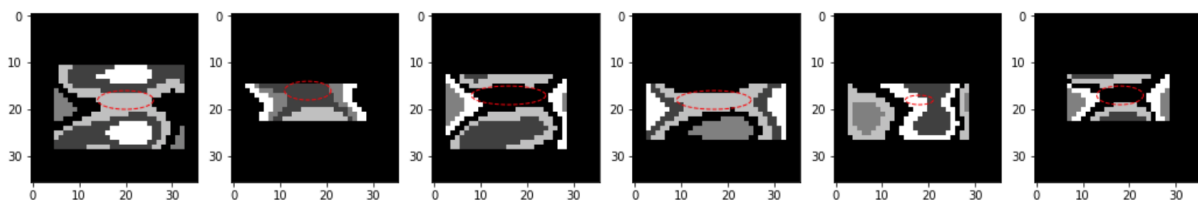


Figure 18 Les régions d'intérêts trouvées par GMM clusters

3. Résultats

Nous obtenons les résultats qui sont affichés dans la Figure 18. Nous décidons que le cluster central est la région d'intérêt au centre, et le cluster dans les deux côtés sont les régions d'intérêt aux côtés. Nous essayons de trouver une ellipse qui peut représenter la région d'intérêt au centre par un algorithme de remplissage:

No. 1		Final centers: (20, 18)		Final rayons: (6, 2)
No. 2		Final centers: (16, 16)		Final rayons: (5, 2)
No. 3		Final centers: (16, 17)		Final rayons: (8, 2)
No. 4		Final centers: (17, 18)		Final rayons: (8, 2)
No. 5		Final centers: (18, 18)		Final rayons: (3, 1)
No. 6		Final centers: (18, 17)		Final rayons: (5, 2)

Figure 19 Résultats de la recherche des ellipses

En voyant la Figure 18 et la Figure 19, nous constatons que la classification n'était pas très efficace.

En résumé, nous pouvons voir que pour certains patches avec de meilleurs résultats de classification, nous pouvons trouver les régions d'intérêt, mais pour les patches avec de mauvais résultats de classification, il est difficile de déterminer les régions d'intérêt. Par conséquent, les méthodes traditionnelles ne sont pas réalisables pour trouver les régions d'intérêt. Nous devons appliquer une méthode plus efficace pour distinguer ces petites zones. Donc, nous pensons à la segmentation dans l'apprentissage profond.

iii. Méthode U-net

1. Pré-traitement des données

Pour séparer les régions d'intérêt à partir de ces patches, nous devons faire des labels binaires pour marquer les régions d'intérêt, qui sont des entrées de notre modèle suivant.

Les labels binaires sont obtenus de manière manuelle. Cependant, afin de rendre les labels plus précis et le réseau plus profond, nous augmentons la taille des patches à 128x128. Les labels binaires se divisent en deux groupes : un groupe est le masque au centre des régions d'intérêt **(a)**, l'autre groupe est le masque des deux côtés des régions d'intérêt **(b)**. Nous visualisons les labels du groupe **(a)**, les labels du groupe **(b)** et les deux groupes ensemble **(c)** respectivement :

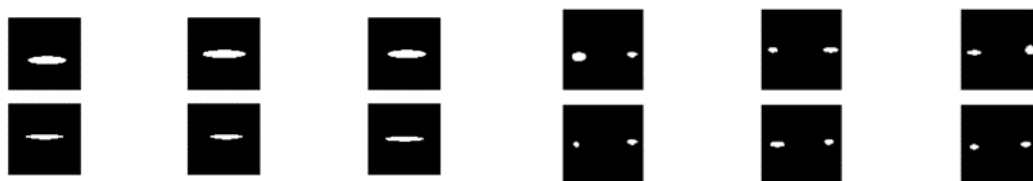


Figure 20 **(a)** à gauche et **(b)** à droite

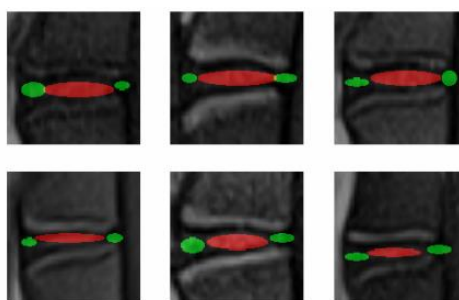


Figure 21 **(c)** des labels sont marqués avec les couleurs rouges (centre) et verts (côtés)

2. Modèle

En constatant que la méthode traditionnelle ne permet pas d'avoir un résultat précis, nous nous intéressons ensuite aux méthodes d'apprentissage profond. Nous proposons U-net [3] qui a pour objectif de faire la segmentation des régions d'intérêt.

L'architecture du U-net se compose de deux parties. L'une est l'encodeur, l'autre est le décodeur et chaque partie comprend cinq blocks. Chaque block est composé des couches

convolutionnelles, Maxpooling, Concaténation, et la fonction d'activation LeakyReLU. Nous visualisons l'architecture ci-dessous :

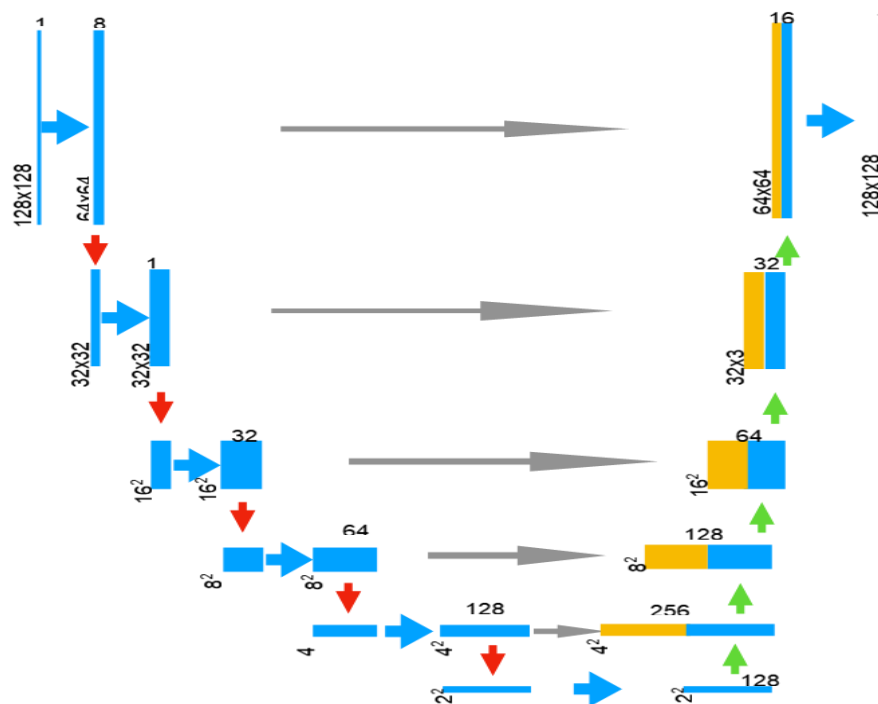


Figure 22 Architecture du U-net

Bien entendu, ce modèle a évolué au cours du temps afin d'augmenter la performance non seulement selon nos connaissances mais aussi selon les contraintes auxquelles nous avons fait face. Finalement, nous fixons le modèle et nous utilisons plusieurs techniques qui doivent être soulignées :

- LeakyReLU :
Il y a une fonction d'activation non linéaire derrière chaque couche de convolutionnelle. La superposition de ces fonctions d'activation non linéaires rend le réseau capable d'apprendre des modèles complexes. LeakyReLU résout le problème de la disparition des gradients et accélère la convergence.
- Concaténation:
Les couches de concaténation, qui est utilisé pour faire le Skip-Connection, sont les couches très importantes. Il ajoute les détails omis de l'encodeur à la partie décodeur correspondante, ce qui est très utile pour la restauration de l'image.

3. Entraînement

Nous choisissons par hasard 50 patches parmi 520 pour faire du label binaire. Par conséquent, avant d'entraîner les données, nous réalisons une augmentation des données pour que nous ayons un nombre de données suffisant et que les données sont plus générales. Nous réalisons l'augmentation des données par :

- Faire rotation des images entre 0 à 10 degrés

- Changement de hauteur et de largeur d'une plage de 0.1
- Renversement horizontale
- Cisaillement des images d'une plage de 0.2
- Zoom d'une plage de 0.2

Donc, nous sommes capables d'apprendre la robustesse au changement de direction horizontal, de position horizontale et verticale, de degrés de rotation et de zoom.

En ce qui concerne les hyperparamètres de l'entraînement, nous mettons :

- Taux d'apprentissage initial = 0.001 qui s'évalue au cours de l'époque
- Facteur de moment = 0.9
- Taille du lot = 2
- Époque = 240

En ce qui concerne la segmentation des données, nous considérons par hasard 70% des données comme les données d'entraînement et 30% des données comme les données de test pour chaque époque.

En ce qui concerne l'optimisateur, nous choisissons SGD avec Moment. SGD calcule le gradient d'un seul exemple au lieu de calculer le gradient de tous les exemples, ce qui permet d'entraîner le modèle plus vite. Le Moment permet de subir une accélération via le gradient, ce qui permet au modèle d'avoir la tendance de converger plus vite. Nous comparons SGD avec Adam et nous concluons que SGD a la capacité de s'entraîner beaucoup plus vite sans perdre de la performance.

En ce qui concerne le taux d'apprentissage, à l'époque 30, il est augmenté 10 fois que ceux original. Ensuite, après chaque 80 époques, il est réduit au dixième de ceux précédent. Le taux d'apprentissage au début ne doit pas être grand pour éviter l'explosion de gradient. Ensuite, le taux d'apprentissage diminue pour mieux converger. Nous visualisons l'évolution au cours de l'époque :

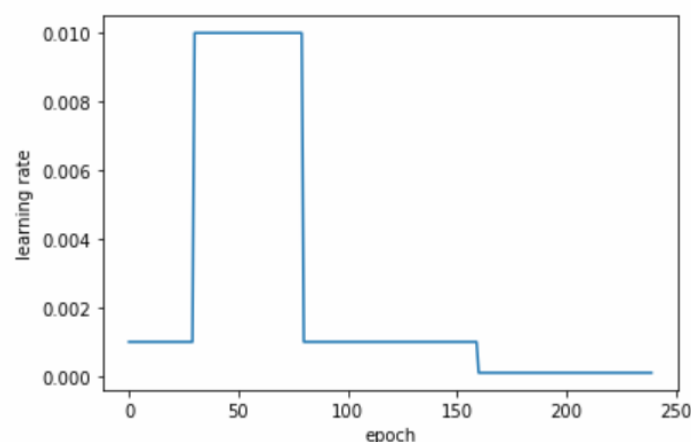


Figure 23 Évolution du taux d'apprentissage au cours de l'époque

En ce qui concerne la métrique, nous choisissons l'intersection moyenne sur l'union (*mean-*

IOU). Mean-IOU est une métrique d'évaluation utilisée pour mesurer la précision d'un détecteur d'objet sur un ensemble de données particulier. Il est très adapté à la segmentation sémantique.

En ce qui concerne la fonction de perte, nous combinons deux méthodes de calcul des pertes. La première est le coefficient de Dice qui s'agit de calculer la similarité entre deux régions de contour, ce qui est très courant en segmentation sémantique. La deuxième, Nous ajoutons une fonction de perte d'entropie croisée adaptée à la classification binaire. Nous combinons ces deux méthodes de calcul de la fonction de perte pour rendre les résultats plus précis.

4. Résultats

Nous entraînons le modèle et le testons sur les données de test.

Les résultats se composent de deux parties, la première partie est le région d'intérêt au centre et l'autre partie est le région d'intérêt à deux côtés.

Première partie : cette partie travail sur les régions d'intérêt du centre

En ce qui concernant la perte, nous traçons l'évolution de la perte des données de validation. Nous la visualisons ci-dessus :

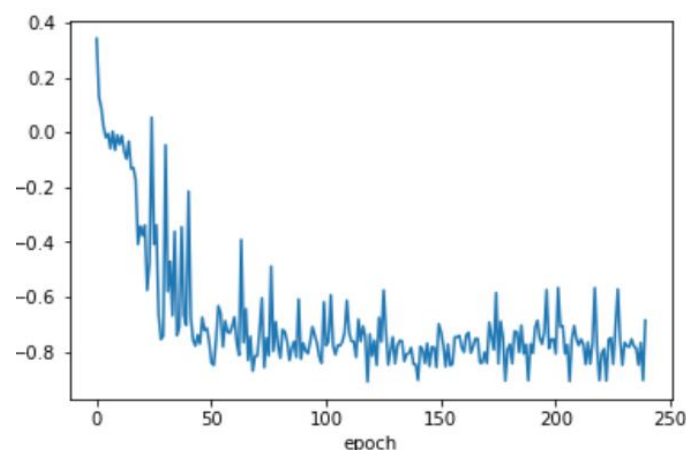


Figure 24 Évolution de la perte des données de validation de la région d'intérêt du centre

Nous pouvons voir qu'il y a une tendance à la baisse dans l'ensemble, mais après environ 50 époques, il y aura de grandes fluctuations. La moyenne de ces fluctuations est d'environ -0,8, l'amplitude est de plus ou moins 0,1. Donc, l'entraînement reste à améliorer.

En ce qui concernant le Mean IOU, nous traçons l'évolution du Mean IOU des données de validation et la visualisons ci-dessus :

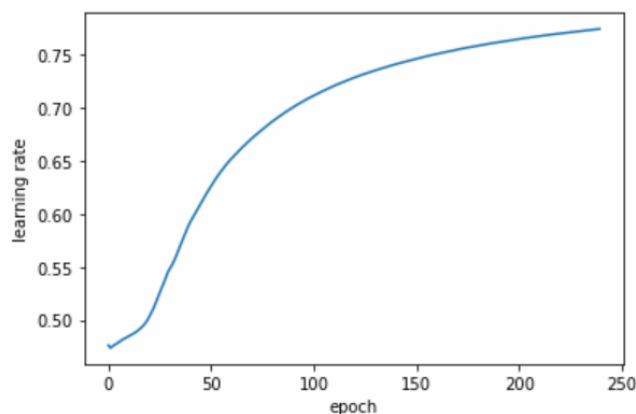


Figure 25 Évolution du Mean IOU des données de validation de la région d'intérêt du centre

Nous pouvons voir qu'il y a une tendance d'augmentation douce sans aucune fluctuation.

Après l'entraînement de 240 époques, nous obtenons les résultats comme ci-dessous :

Perte d'entraînement	-0.7748
Mean IOU d'entraînement	0.7746
Perte de validation	-0.6860
Mean IOU de validation	0.7747

Tableau 6 Résultat quantitatif de la région d'intérêt du centre

Nous visualisons les patches et leur résultat de segmentations ci-dessous :

Patches					
Résultats de segmentation					

Figure 26 Résultats qualitatifs de la région d'intérêt du centre

Nous constatons que même si la valeur du Mean IOU n'est pas assez grand, les résultats sont satisfaisants.

Deuxième partie: cette partie travail sur les régions d'intérêt de deux côtés

En ce qui concernant la perte, nous traçons l'évolution de la perte des données de validation. Nous la visualisons ci-dessus :

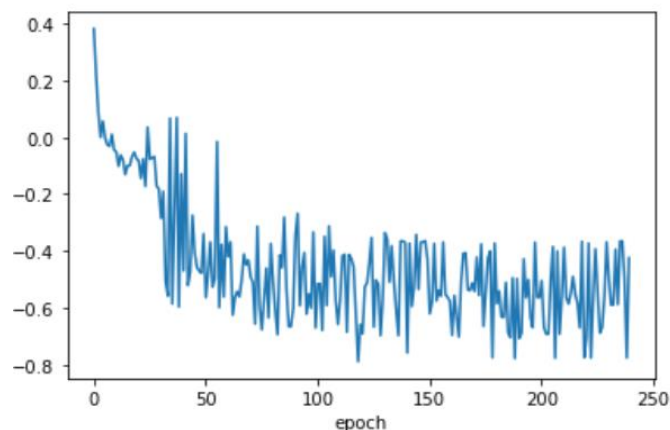


Figure 27 Évolution de la perte des données de validation de la région d'intérêt des côtes

Nous pouvons voir qu'il y a une tendance à la baisse dans l'ensemble, mais après environ 50 époques, il y aura de grandes fluctuations. La moyenne de ces fluctuations est d'environ -0,5, l'amplitude est de plus ou moins 0,2.

En ce qui concernant le Mean IOU, nous traçons l'évolution du Mean IOU des données de validation et la visualisons ci-dessus :

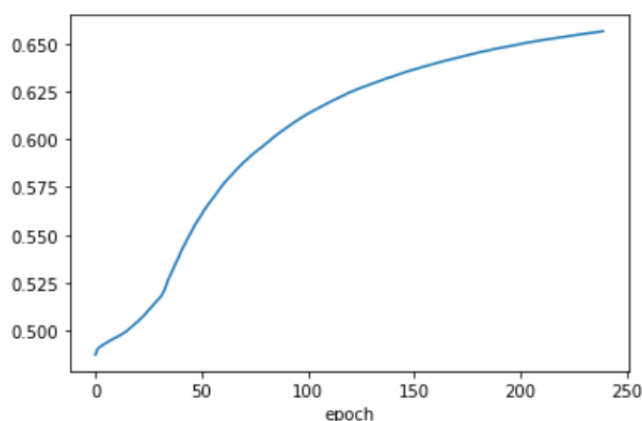


Figure 28 Évolution du Mean IOU des données de validation de la région d'intérêt des côtes

Nous pouvons voir qu'il y a une tendance d'augmentation douce sans aucune fluctuation.

Après l'entraînement de 240 époques, nous obtenons les résultats comme ci-dessous :

Perte d'entraînement	-0.5150
Mean IOU d'entraînement	0.6565
Perte de validation	-0.4242
Mean IOU de validation	0.6566

Tableau 7 Résulta quantitatif de la région d'intérêt aux côtés

Nous avons trouvé que ni la valeur du Mean IOU ni la valeur de la perte ne sont très bonnes. Nous pensons qu'il y a plusieurs raisons possibles :

- Il est difficile de voir clairement la position spécifique de la région d'intérêt par l'observation oculaire, donc des erreurs peuvent survenir lors des labels binaires qui se fait manuellement.
- La région d'intérêt est de taille petite, donc la segmentation est inexacte.

Nous visualisons les patches et leur résultat de segmentations ci-dessous :

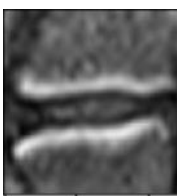
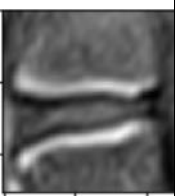








Patches					
Résultats de segmentation					

Figure 29 Résultats qualitatifs de la région d'intérêt des côtes

iv. Conclusion

Après avoir comparé la méthode traditionnelle avec la méthode d'apprentissage profond, nous constatons que la méthode traditionnelle est instable, et les résultats sont beaucoup influencés par les données spéciales. Pour la méthode d'apprentissage profond, nous constatons que le modèle est non seulement plus robuste mais aussi plus précis. Le seul inconvénient est le besoin des labels fait manuellement, la précision des labels affecte directement les résultats.

c) Recalage

D'après les deux grandes parties précédentes, nous avons réalisé la location des trois régions d'intérêt dans les images T1 SAG et T2 SAG. Cette partie envisage de faire du recalage afin de localiser les régions d'intérêt dans les images T1, T2 et T2*, d'où les valeurs numériques servent à définir le niveau dégénératif des disques. Nous avons réalisé la superposition des différents types des images pour voir la nécessité du recalage :

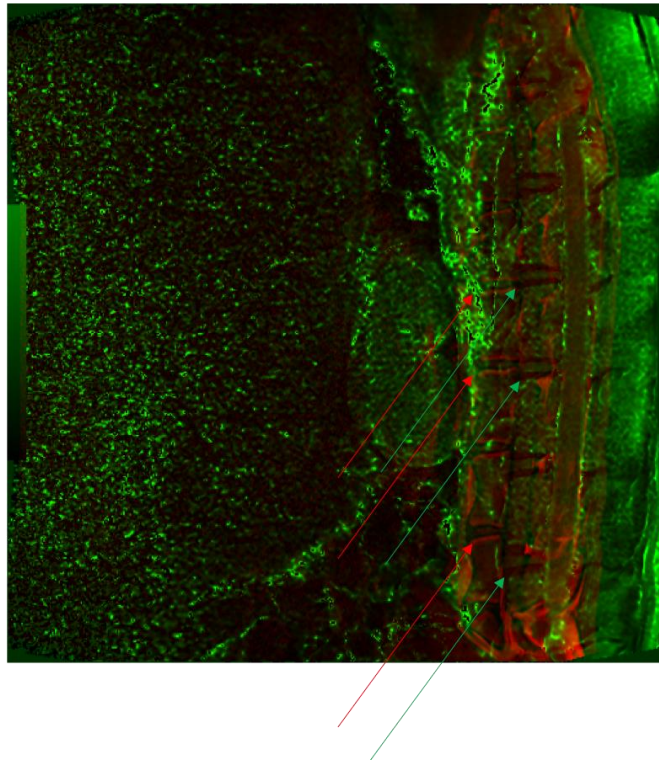


Figure 30 Exemple de la superposition de T1 SAG (rouge) et T1 (vert)

Nous voyons que les localisations ne sont pas correspondantes. Il faut donc faire un recalage.

Nous proposons une méthode semi-automatique qui demande les connaissances à priori sur les localisations des régions d'intérêt.

Après la localisation des régions d'intérêt dans les images objectives, la segmentation est appliquée afin de trouver les valeurs numériques qui servent à définir le niveau dégénératif des disques. Nous les trouvons bien et pensons que cette méthode est faisable ^[a].

Pour aller plus loin, nous allons réaliser une méthode automatique d'où le modèle sera Spatial Transformer Network [4]. Spatial Transformer Network a la capacité d'apprendre la transformation affine. Dans le plan 2D, le filtre de la transformation affine peut être représenté par une matrice de dimension (3x2) qui peut être appris par la propagation du gradient. Dans ce cas, notre pipeline sera complètement automatique.

Cependant, à cause de la limitation du temps, nous choisissons de réaliser la méthode semi-automatique qui est proposée dans l'intention de faire un compromis entre les besoins techniques et l'économisation du temps.

[a] Les résultats numériques sont disponibles sur <https://github.com/chengqianben/Automatic-MRI-Degeneration-Detection>

IV. BILAN DU PROJET

a) Analyse des Résultats

Après la comparaison des valeurs numériques avec ce qui sont obtenus par le côté de l'École Nationale Vétérinaire de Nantes, nous constatons qu'il existe une erreur qui est parfois acceptable, mais parfois un peu grande à cause de nombreuses raisons :

- Les prises manuelles des labels ne sont pas techniquement parfaites.
- La location de la première grande partie (détection) est un peu décalée lors de la détection et du clustering :



Figure 31 Exemple d'un résultat de la détection qui est un peu décalé vers le haut

- L'entraînement du modèle U-net avec les données augmentées reste à améliorer, ce qui vient du fait que les prises manuelles des labels ne sont pas nombreuses, et aussi du fait que la taille des régions d'intérêt n'est pas grande, d'où les valeurs, qui sont pixel par pixel, ne peuvent pas être considérées continues.

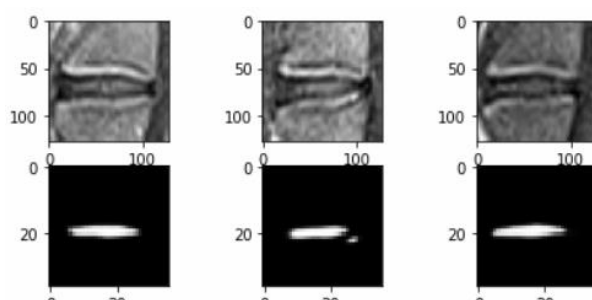


Figure 32 Entraînement du modèle U-net reste à améliorer

- Les positions des images sources et objectives ne sont pas complètement correspondantes, ce qui pose une erreur pour le recalage:

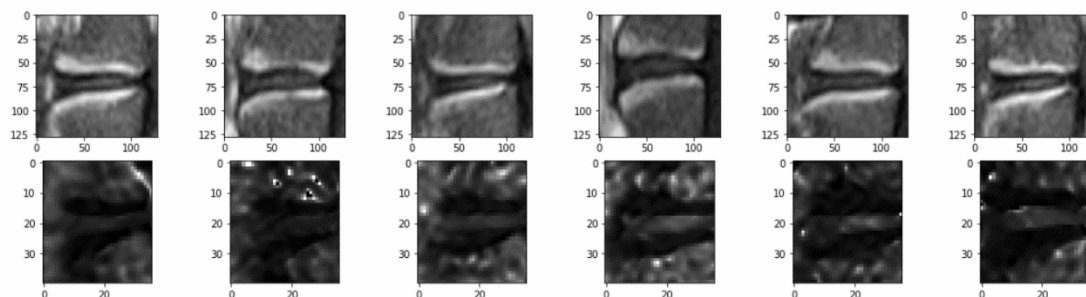


Figure 33 Superposition des images sources (ci-dessus) et objectives (ci-dessous) n'est pas parfaite

Donc, les erreurs sont additionnées étape par étape et déduit à une valeur que nous ne pouvons pas ignorer.

b) Conclusion

En général, nous avons obtenu de bons résultats et pratiquement atteint nos objectifs initiaux. Après avoir communiqué activement avec les enseignants, avoir recherché les articles et avoir essayé différentes méthodes, nous avons déterminé notre plan de base: utiliser l'apprentissage profond pour déterminer les régions d'intérêt. Nous utilisons d'abord la méthode CNN pour détecter les patches, ensuite la méthode U-net pour réaliser la segmentation des régions d'intérêt et finalement le recalage pour trouver les valeurs numériques des disques.

Nous avons beaucoup appris au cours du projet. Premièrement, nos connaissances professionnelles ont été renforcées. Contrairement aux projets dans le cours habituel, ce projet professionnel est plus systématique, pratique et créatif. De l'interprétation du contenu du projet à la recherche d'une méthode adaptée pour y parvenir, nous avons besoin d'expérimentations sans cesse et d'innovations. Dans ce processus, nous avons revu et approfondi nos connaissances, y compris les connaissances simples en traitement d'images, les méthodes traditionnelles de détection d'objets et les modèles complexes d'apprentissage profond. Dans le même temps, nous avons amélioré nos capacités de traitement des données. Comment extraire les données dont nous avons besoin à partir d'une grande quantité de données et les organiser dans un dossier, comment implémenter les résultats de différents logiciels (Image J et Horos) en python et comment visualiser les données sont autant de problèmes que nous avons mis beaucoup d'énergie à réfléchir. Enfin, notre capacité à coopérer et à communiquer a été améliorée. En plus de la coopération entre nous trois, ce projet nécessite également une communication constante avec nos collaborateurs pour clarifier leurs besoins, et doit également communiquer avec notre tuteur pour des conseils et des inspirations.

Cependant, notre projet n'a pas encore donné de résultats parfaits. Nous sommes arrivés à trouver les régions d'intérêt dans les images T1 SAG et T2 SAG mais les informations dont nous avons besoin pour évoluer la dégénérescence discale lombaire chez la brebis sont stockées dans les images T1, T2 et T2*. Une méthode de recalage est indispensable, mais pour l'instant une méthode semi-automatique est proposée. Pour aller plus loin, nous proposons Spatial Transformer Networks qui est automatique et nous allons continuer d'affiner notre projet.

Enfin, les temps impartis à la réalisation de chaque partie furent bref et il a fallu faire preuve de flexibilité et de persévérance, parfois pour respecter les délais, parfois pour respecter les contraintes technologiques imposées par le projet. Somme toute, nous avons retrouvé lors de ces cinq mois, les compétences, les contraintes mais aussi l'excitation d'un projet d'école et d'entreprise.

c) Lien Externe

Dans l'intention de nous échanger des idées et nous communiquer, nous établissons un lien externe et y mettons tous les codes : <https://github.com/chengqianben/Automatic-MRI-Degeneration-Detection>

V. BIBLIOGRAPHIE

1. Lootus M., Kadir T., Zisserman A. (2014) Vertebrae Detection and Labelling in Lumbar MR Images.
2. Jamaludin, Amir & Kadir, Timor & Zisserman, Andrew. (2017). SpineNet: Automated Classification and Evidence Visualization in Spinal MRIs. Medical Image Analysis.
3. Lu, Jen-Tang & Pedemonte, Stefano & Bizzo, Bernardo & Doyle, Sean & Andriole, Katherine & Michalski, Mark & Gonzalez, R. & Pomerantz, Stuart. (2018). DeepSPINE: Automated Lumbar Vertebral Segmentation, Disc-level Designation, and Spinal Stenosis Grading Using Deep Learning.
4. M Jaderberg, K Simonyan, A Zisserman - Advances in neural information processing systems. (2015) Spatial Transformer Networks.