Analyse de la segmentation des mitochondries dans des images obtenues par microscopie électronique.

Guillaume Charvolin

5 octobre 2024

Abstract

La segmentation des mitochondries dans des images issues de la microscopie électronique est une étape essentielle pour l'analyse cellulaire, particulièrement dans le contexte de certaines pathologies. Ce travail vise à implémenter une segmentation d'instance sur ces images afin de détecter leur position. L'objectif est de développer une solution simple et peu couteuse qui répond au problème de manière satisfaisante. Les résultats obtenus sont pertinents et permettent d'identifier les mitochondries.

Table des matières

Chapitre 1

Introduction

Les mitochondries sont des organelles présentes dans la plupart des cellules eucaryotes et jouent un rôle central dans la production d'énergie, en convertissant les nutriments en adénosine triphosphate (ATP) avec la respiration cellulaire. En plus de leur rôle énergétique, elles sont impliquées dans d'autres processus cellulaires cruciaux comme l'apoptose (mort cellulaire programmée) et la régulation du cycle cellulaire. De nombreuses pathologies, notamment les maladies neurodégénératives et métaboliques, sont associées à des anomalies dans le nombre, la forme ou la fonction des mitochondries. Par conséquent, leur analyse précise à partir d'images obtenues par microscopie électronique est essentielle pour comprendre et diagnostiquer ces affections.

La segmentation automatique des mitochondries dans des images microscopiques est une tâche complexe, en raison de la structure irrégulière de ces organelles et des variations dans les conditions d'imagerie. Bien que des solutions de segmentation fondées sur l'apprentissage profond aient démontré leur efficacité dans l'analyse d'images biomédicales, le choix de l'algorithme reste crucial. Des architectures telles que U-Net ou Attention U-Net sont souvent utilisées pour ce type de tâche, mais présentent des limitations en termes de précision ou de coût de calcul selon la taille des données et la complexité des structures segmentées.

Dans ce travail, nous allons nous concentrer sur la recherche d'un réseau de neurone capable d'identifier les mitochondries dans une série d'images obtenues par microscopie électronique. La performance du modèle sera évaluée à l'aide de métriques telles que le dice similarity coefficient (DSC) et l'indice de Jaccard (IoU), qui sont particulièrement pertinentes pour ce type d'imagerie médicale.

Ce rapport est structuré de la manière suivante : le chapitre 2 présente l'état de l'art sur les techniques de segmentation d'images basées sur l'apprentissage profond, avec un focus sur celles appliquées à l'imagerie par microscopie électronique. Le chapitre 3 détaille la méthodologie employée, notamment la formalisation du problème, l'architecture retenue, et la fonction de coût utilisée. Le chapitre 4 présente les expérimentations et les résultats, en commençant par une présentation des données, les étapes d'optimisation et de réglage des hyperparamètres, suivis des résultats obtenus. Enfin, le chapitre 5 propose une conclusion et discute des perspectives d'amélioration.

Chapitre 2

État de l'art

Segmentation d'image médicale par microscopie électronique

La segmentation automatique des mitochondries dans les images de microscopie électronique constitue un défi majeur en raison de la complexité et de la variabilité de leurs structures. Les mitochondries présentent des formes et des tailles diversifiées, rendant difficile leur identification précise avec des méthodes traditionnelles de traitement d'images.

Pour relever ce défi, Xiao et al. [5] ont proposé une méthode de segmentation 3D des mitochondries basée sur des réseaux neuronaux convolutifs résiduels hautement supervisés. Cette approche permet de capturer les caractéristiques complexes des mitochondries dans des images volumétriques, améliorant ainsi la précision de la segmentation.

Manca et al. [4] ont développé une méthode automatique pour segmenter les compartiments intracellulaires tels que les mitochondries et les endolysosomes. En comparant leur algorithme avec des architectures existantes comme U-Net, V-Net et DeepMedic, ils ont obtenu des scores de Dice de 0,855, 0,898 et 0,867 respectivement, démontrant l'efficacité de leur approche.

Des avancées récentes incluent le travail d'Oztel et al. [1], qui ont rapporté des résultats prometteurs pour la segmentation autonome des mitochondries dans les tissus cérébraux en utilisant des réseaux neuronaux convolutifs profonds. De plus, Chacon et al. [2] ont introduit une méthode d'adaptation de domaine basée sur deux U-Net couplés partageant les poids et utilisant une fonction de perte différentiable approchant l'indice de Jaccard, améliorant la robustesse de la segmentation face à la variabilité des données.

Chapitre 3

Méthode

3.1 Formalisation du Problème

Formellement, soit $\mathcal{I} = \{I_n\}_{n=1}^{N}$ un ensemble d'images en niveaux de gris, où chaque image $I_n \in \mathbb{R}^{H \times W}$ représente une coupe bidimensionnelle de tissu cérébral obtenue par microscopie électronique. Chaque pixel de l'image I_n possède une intensité correspondant à la densité électronique du tissu.

L'objectif est de définir une fonction de segmentation $f : \mathbb{R}^{H \times W} \to \{0,1\}^{H \times W}$ telle que, pour chaque image I_n, la sortie $S_n = f(I_n)$ est un masque binaire où :

$$S_n(i,j) = \begin{cases} 1, & \text{si le pixel } (i,j) \text{ appartient à une mitochondrie,} \\ 0, & \text{sinon.} \end{cases}$$

La tâche est donc un problème de segmentation sémantique binaire au niveau pixel, où l'on cherche à approximer la fonction f en entraînant un modèle paramétrique $\hat{f}_\theta$ (par exemple, un réseau de neurones convolutionnel) sur un ensemble de données annotées $\{(I_n, S_n)\}_{n=1}^{N}$.

Les contraintes spécifiques du problème sont les suivantes :

- **Dimensionnalité des données** : Les images I_n sont de taille 256×256 pixels en niveaux de gris.
- **Complexité computationnelle** : Le temps d'entraînement du modèle doit être limité à un maximum de 2 heures, imposant des restrictions sur la taille du modèle et les ressources utilisées.
- **Performance attendue** : Le modèle doit atteindre au minimum un coefficient de Dice de 0,6 et un indice de Jaccard de 0,4 sur les données de test, garantissant une segmentation de qualité acceptable.
- **Prétraitement des données** : Le prétraitement des images (par exemple, normalisation, augmentation de données, filtrage) est considéré comme un hyperparamètre du modèle. Nous chercherons à identifier les hyperparamètres les plus pertinents, y compris ceux liés au prétraitement, pour optimiser la performance du modèle.

Le défi principal consiste à concevoir un modèle efficace qui puisse apprendre des caractéristiques discriminantes pour distinguer les mitochondries des autres structures cellulaires, malgré la complexité des textures et les variations d'intensité propres aux images de microscopie électronique.

En résumé, le problème consiste à développer un modèle d'apprentissage automatique qui, étant donné une image de microscopie électronique en entrée, produit un masque

de segmentation précis des mitochondries, tout en respectant les contraintes de temps d'entraînement et de performance. L'optimisation des hyperparamètres, y compris ceux liés au prétraitement des données, sera essentielle pour atteindre cet objectif.

3.2 Architecture

Dans cette section, nous discutons les différentes architectures disponibles et justifions les choix effectués pour ce projet. Initialement, nous avions la possibilité d'utiliser un perceptron multicouche (MLP). Toutefois, une architecture MLP n'est pas adaptée à notre tâche de segmentation d'images car elle ne tient pas compte de l'aspect spatial des données. Les MLP traitent chaque pixel de manière indépendante, sans capturer les relations spatiales essentielles dans les images. Cela aurait conduit à une perte d'information cruciale pour la segmentation précise des mitochondries.

Une autre option consiste à utiliser des architectures de type U-Net, largement reconnues pour leur efficacité en segmentation d'images. Les U-Net bénéficient d'une capacité à combiner des informations locales et globales via des couches de convolution et des connexions résiduelles (*skip connections*) entre l'encodeur et le décodeur. Cependant, les U-Net classiques ou améliorés sont souvent coûteux en termes de calcul et de mémoire, surtout lorsque le nombre de couches est élevé. L'entraînement de modèles U-Net avec de nombreuses couches risquerait d'augmenter considérablement le temps d'entraînement, dépassant ainsi les contraintes temporelles de notre projet.

Par conséquent, nous avons opté pour une approche basée sur des autoencodeurs convolutionnels. Cette architecture est bien adaptée pour la tâche de segmentation tout en restant plus simple à implémenter et à entraîner dans un délai raisonnable. Nous avons défini des modèles autoencodeurs avec un nombre variable de couches, où l'encodeur augmente progressivement le nombre de canaux pour extraire des caractéristiques de plus en plus complexes, et le décodeur réduit progressivement le nombre de canaux pour reconstruire l'image segmentée.

Formellement, chaque modèle autoencodeur est constitué de couches convolutives avec un padding qui permet de conserver la même taille d'image à chaque étape. Si le modèle contient, par exemple, 6 couches, celles-ci seraient réparties en trois couches d'encodeur et trois couches de décodeur. La partie encodeur commence avec 1 canal (car c'est une image en niveau de gris), qui est ensuite transformé en 8 canaux. La deuxième couche transforme les 8 canaux en 16 canaux, augmentant ainsi la capacité de l'encodeur à extraire des caractéristiques complexes. Ensuite, dans la partie décodeur, les 16 canaux sont reconvertis en 8 canaux, puis en 1 canal final, correspondant à la sortie segmentée de l'image.

Lorsque le nombre de couches est impair, par exemple 5 couches, la couche centrale agit comme un lien entre les deux parties (encodeur et décodeur) sans modifier le nombre de canaux. Par exemple, si l'encodeur se termine avec 16 canaux, la couche centrale aurait également 16 canaux.

Contrairement à la convention courante d'augmenter le nombre de canaux par un facteur de 2 à chaque couche (par exemple 1, 8, 16, 32, etc.), nous avons fait le choix d'une augmentation plus modeste, avec un incrément de 8 canaux à chaque étape. Ce compromis a été nécessaire pour éviter que le modèle ne devienne trop complexe et coûteux en termes de calcul lorsque le nombre de couches augmente. Une augmentation exponentielle du nombre de canaux aurait rapidement fait exploser la taille du modèle, rendant impossible le test avec de nombreuses couches tout en respectant les contraintes temporelles et de ressources.

Ainsi, cette architecture permet de tester différentes configurations avec un nombre variable de couches sans dépasser les limites de complexité du modèle, tout en conservant une capacité suffisante pour capturer les caractéristiques importantes des images de mitochondries.

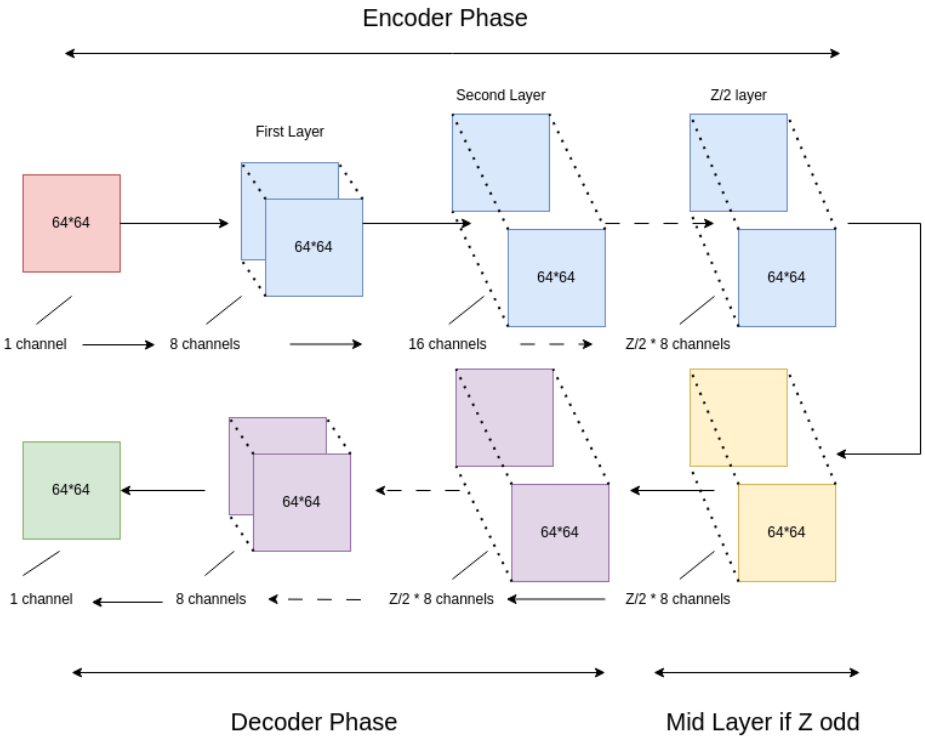

FIGURE 3.1 – Représentation schématique du modèle autoencodeur convolutionnel utilisé dans ce projet. L'encodeur augmente progressivement le nombre de canaux pour extraire des caractéristiques complexes, tandis que le décodeur réduit ce nombre de canaux pour reconstruire l'image segmentée. Le nombre de couches (Z) est variable, permettant une flexibilité dans la profondeur du modèle.

3.3 Fonction de coût

Pour entraîner le modèle de segmentation des mitochondries, nous avons utilisé la **perte d'entropie croisée binaire** (*Binary Cross-Entropy Loss*), également connue sous le nom de fonction de coût logistique. Cette fonction est adaptée aux problèmes de classification binaire, comme c'est le cas pour la segmentation des pixels en deux classes : mitochondrie ou non-mitochondrie.

La perte d'entropie croisée binaire est définie par :

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right],$$

où :

— N est le nombre total de pixels dans l'image,

— $y_i \in \{0, 1\}$ est la valeur réelle (label) du pixel i,

— $p_i \in [0, 1]$ est la probabilité prédite par le modèle que le pixel i appartienne à la classe des mitochondries.

Cette fonction de coût mesure la divergence entre les labels réels et les probabilités prédites par le modèle. Elle pénalise fortement les prédictions erronées, en accordant une importance particulière aux cas où le modèle est certain de sa prédiction mais se trompe.

Le choix de la perte d'entropie croisée binaire se justifie par plusieurs raisons :

— **Adaptation aux problèmes de classification binaire** : Étant spécifiquement conçue pour les tâches de classification binaire, elle est naturellement appropriée pour segmenter les pixels en deux classes distinctes.

— **Simplicité et efficacité computationnelle** : La fonction est simple à implémenter et permet un calcul efficace du gradient, ce qui est essentiel pour optimiser le modèle dans le temps imparti de 2 heures.

— **Gestion des probabilités** : Elle intègre directement les probabilités prédites par le modèle, permettant une interprétation probabiliste des résultats et facilitant l'ajustement du seuil de décision si nécessaire.

Bien que d'autres fonctions de coût, comme la *Dice Loss* ou la *Focal Loss*, puissent être utilisées pour les problèmes de segmentation avec des déséquilibres de classes, la perte d'entropie croisée binaire reste un choix standard et éprouvé. Dans le contexte de notre problématique, où l'objectif est d'obtenir des performances satisfaisantes avec des contraintes temporelles strictes, la simplicité et l'efficacité de la perte d'entropie croisée binaire en font une option appropriée.

Chapitre 4

Expérimentation et résultats

4.1 Présentation des Données

Pour entraîner et évaluer notre modèle de segmentation des mitochondries, nous avons utilisé un jeu de données d'imagerie de microscopie électronique fourni par l'École Polytechnique Fédérale de Lausanne (EPFL). Ce jeu de données représente une section de $5 \times 5 \times 5\,\mu\text{m}^3$ prélevée dans la région CA1 de l'hippocampe du cerveau, correspondant à un volume de $1065 \times 2048 \times 1536$ voxels. Chaque voxel a une résolution d'environ $5 \times 5 \times 5\,\text{nm}^3$, ce qui permet une visualisation détaillée des structures subcellulaires.

Les données sont fournies sous forme de fichiers TIFF multipages, pouvant être chargés et visualisés à l'aide de logiciels tels que *Fiji*. Pour les besoins de notre étude, le volume complet a été divisé en deux sous-volumes annotés :

- **Sous-volume d'entraînement** : Composé des 165 premières coupes de la pile d'images (partie supérieure du volume). Ce sous-volume a été utilisé pour l'apprentissage du modèle.
- **Sous-volume de test** : Composé des 165 coupes suivantes (partie inférieure du volume), utilisé pour évaluer les performances du modèle.

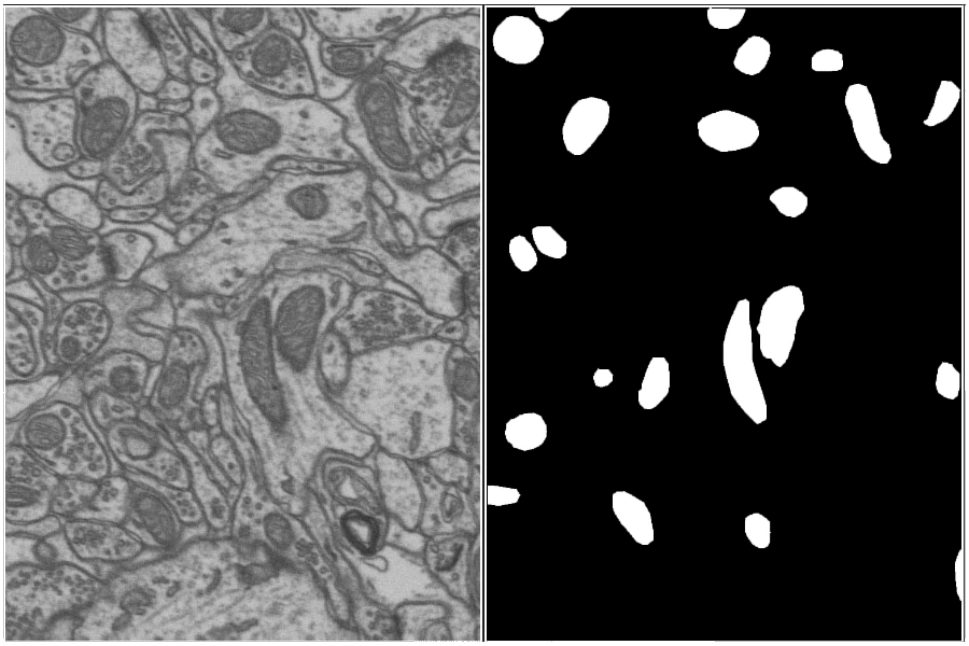

FIGURE 4.1 – À gauche : Image capturée par microscopie électronique du sous-volume. À droite : Masque associé montrant la segmentation manuelle des mitochondries.

Les mitochondries ont été manuellement annotées dans ces deux sous-volumes, fournissant des masques de segmentation précis pour chaque image. Les images sont en niveaux de gris.

Ce jeu de données présente plusieurs caractéristiques importantes pour notre problématique :

- **Annotations de qualité** : Les annotations manuelles fournissent des labels fiables pour l'apprentissage supervisé, améliorant ainsi la qualité de l'entraînement du modèle.
- **Représentativité biologique** : Provenant de la région CA1 de l'hippocampe, le jeu de données est pertinent pour des études liées aux fonctions cognitives et à la mémoire.

Dans notre approche, nous avons utilisé 55 % des données pour l'entraînement et 45 % pour le test, sans constituer de jeu de validation séparé. Cette répartition vise à maximiser la quantité de données disponibles pour l'apprentissage tout en conservant un ensemble de test suffisant pour une évaluation robuste des performances.

Le prétraitement des images a été considéré comme un hyperparamètre clé dans notre méthode. Des techniques telles que la normalisation ont été appliquées pour maximisez la vitesse de convergence des modèles. L'optimisation de ces étapes de prétraitement a contribué de manière significative à l'amélioration des performances du modèle de segmentation.

4.2 Optimisation

4.2.1 Optimiseur

Pour optimiser notre modèle de segmentation, nous avons utilisé l'optimiseur **Adam** [3], reconnu pour sa polyvalence et son efficacité dans l'entraînement de réseaux de neurones

profonds. Adam (*Adaptive Moment Estimation*) combine les avantages de deux méthodes de descente de gradient adaptatives, *AdaGrad* et *RMSProp*, en adaptant dynamiquement les taux d'apprentissage pour chaque paramètre du modèle.

Le choix d'Adam s'explique par sa capacité à converger rapidement et de manière stable, ce qui est particulièrement utile dans le contexte de notre étude où le temps d'entraînement est limité à 2 heures. De plus, Adam est robuste aux paramètres de configuration, fonctionnant efficacement avec les valeurs par défaut, ce qui simplifie le processus d'optimisation des hyperparamètres. Cette efficacité et cette simplicité en font un choix pertinent pour notre tâche de segmentation des mitochondries dans des images de microscopie électronique complexes.

4.2.2 Redimensionnement

En raison des contraintes de temps, nous avons réduit la taille des images d'entrée de `256x256` à `64x64` pixels. En effet, les temps d'entraînement avec des images en haute résolution étaient prohibitifs, comme le montrent les résultats obtenus :

— **Sans redimensionnement** (`256x256`) :

```
Starting training for without_scale...
Epoch [1/3], Loss: 0.2519, Epoch Time: 65.93s, Total Time Elapsed: 65.93s
Epoch [2/3], Loss: 0.1602, Epoch Time: 64.98s, Total Time Elapsed: 130.91s
Epoch [3/3], Loss: 0.1503, Epoch Time: 79.06s, Total Time Elapsed: 209.96s
Test Loss: 0.1405
dice_coefficient: 0.0000
jaccard_index: 0.0000
```

— **Avec redimensionnement** (`64x64`) :

```
Starting training for with_scale...
Epoch [1/3], Loss: 0.2092, Epoch Time: 2.12s, Total Time Elapsed: 2.12s
Epoch [2/3], Loss: 0.1315, Epoch Time: 2.21s, Total Time Elapsed: 4.33s
Epoch [3/3], Loss: 0.1305, Epoch Time: 2.16s, Total Time Elapsed: 6.50s
Test Loss: 0.1252
dice_coefficient: 0.0611
jaccard_index: 0.0322
```

Nous constatons une différence significative en termes de temps d'entraînement entre les deux configurations. Pour les images en haute résolution (256×256), chaque époque prend environ 65 à 79 secondes, entraînant un temps total de près de 210 secondes pour seulement 3 époques. À l'inverse, avec le redimensionnement à 64×64, chaque époque prend environ 2 secondes, pour un total de 6,5 secondes sur 3 époques. Cette différence nous permet de conclure que le redimensionnement était indispensable pour respecter la contrainte de 2 heures imposée par cette étude.

Cependant, la réduction de la résolution des images a également des conséquences importantes sur la performance maximale que notre modèle peut atteindre. En réduisant la taille des images, les structures internes complexes des mitochondries, telles que les canaux mitochondriaux, sont fortement lissées voire totalement perdues. Ces structures fines sont pourtant cruciales pour différencier les mitochondries d'autres éléments présents dans les images. En conséquence, l'absence de ces détails entraîne une limitation inhérente dans la capacité du modèle à effectuer une segmentation précise, ce qui signifie qu'il est peu probable que le modèle atteigne des coefficients de Dice ou des indices de Jaccard très haut.

Néanmoins, dans le cadre de notre étude, ce compromis entre rapidité d'entraînement et précision est jugé acceptable, compte tenu de la contrainte temporelle. Le temps imparti

pour l'entraînement étant limité à 2 heures, il n'était pas possible de traiter efficacement les images en haute résolution (256×256), qui auraient nécessité un temps de calcul beaucoup plus long.

4.3 Réglage des hyperparamètres

Afin de calibrer les hyperparamètres du modèle, nous avons effectué une série de tests succincts avec différentes valeurs pour chacun d'eux. L'objectif était d'explorer l'impact de chaque hyperparamètre en testant des ordres de grandeur différents et en ajustant progressivement les réglages afin de trouver les combinaisons optimales pour le modèle.

4.3.1 Réglage du Learning Rate

En particulier, nous avons testé plusieurs valeurs du taux d'apprentissage (`learning rate`) : 0.5, 1e-2, et 1e-5, afin d'observer leur influence sur la convergence du modèle. Le graphique ci-dessous, généré avec TensorBoard, illustre les pertes (loss) au cours de l'entraînement pour ces trois valeurs de taux d'apprentissage.

Les résultats les plus pertinents à considérer pour chaque test incluent la **Test Loss**, le **Dice Coefficient** et l'**Indice de Jaccard** après 10 époques d'entraînement avec des batchs de 30 exemples :

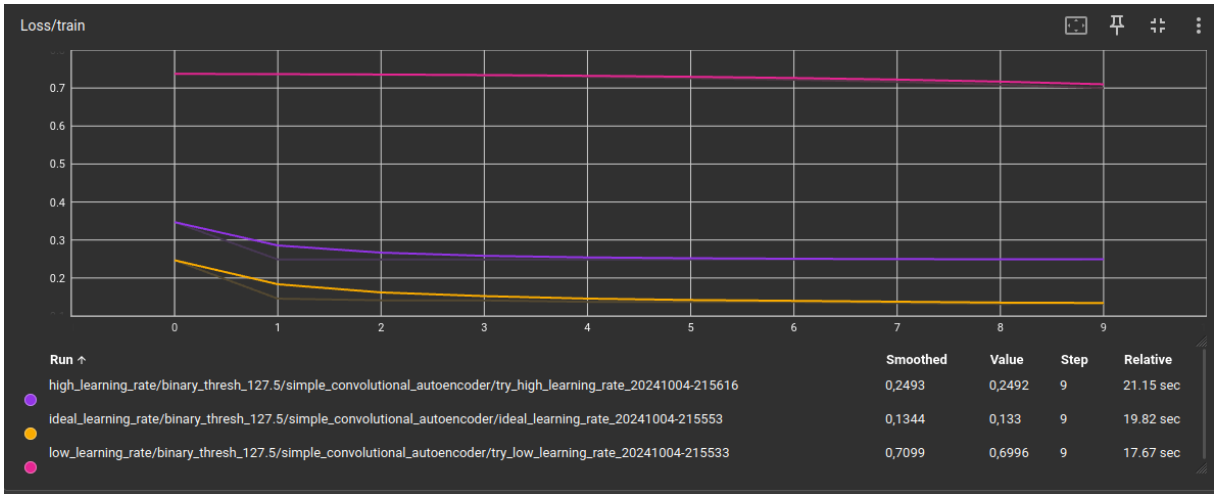

FIGURE 4.2 – Visualisation de l'évolution de la perte au cours de l'entraînement pour différentes valeurs de taux d'apprentissage (high = `0.5`, ideal = `1e-2`, low = `1e-5`) avec TensorBoard.

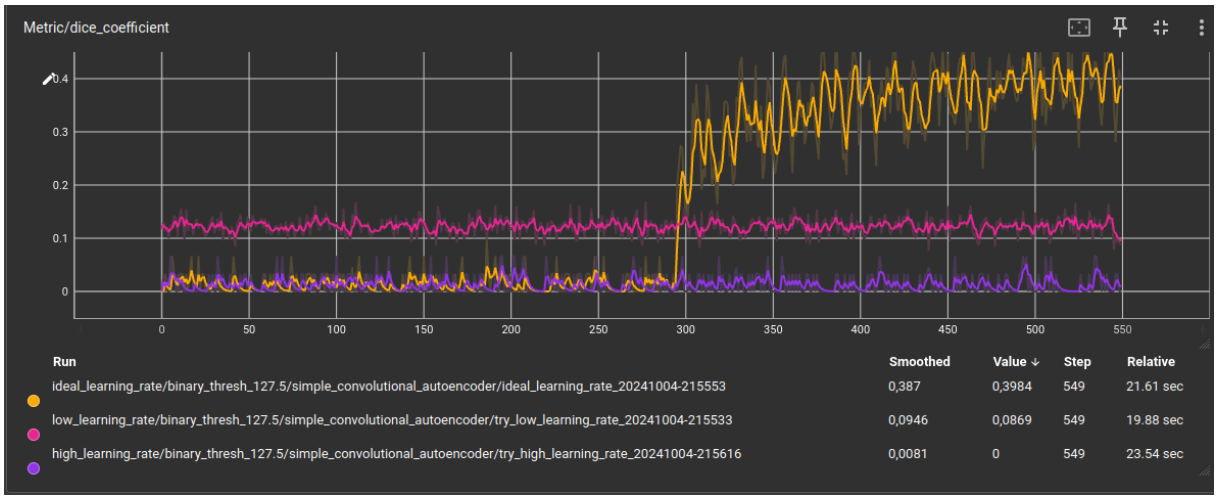

FIGURE 4.3 – Visualisation de l'évolution de dice au cours de l'entraînement pour différentes valeurs de taux d'apprentissage (high = `0.5`, ideal = `1e-2`, low = `1e-5`) avec TensorBoard.

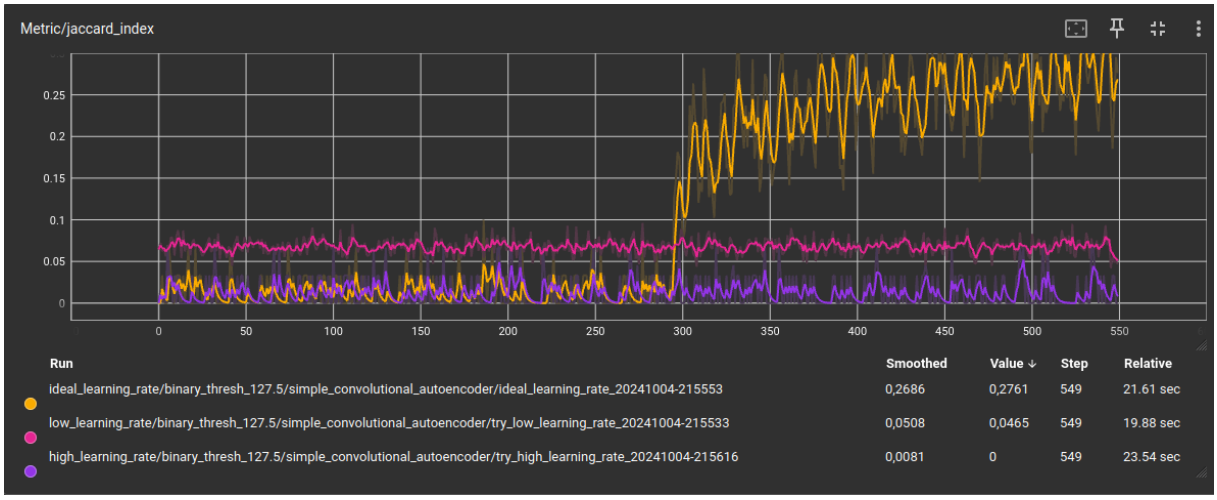

FIGURE 4.4 – Visualisation de l'évolution de jaccard au cours de l'entraînement pour différentes valeurs de taux d'apprentissage (high = `0.5`, ideal = `1e-2`, low = `1e-5`) avec TensorBoard.

Learning Rate	**Test Loss**	**Dice Coefficient**	**Jaccard Index**
Low (1e-5)	0.6935	0.0813	0.0442
Ideal (1e-2)	0.1312	0.5025	0.3658
High (0.5)	0.2306	0.0000	0.0000

TABLE 4.1 – Comparaison des performances avec différents taux d'apprentissage (Learning Rate).

Il est important de noter que ces résultats doivent être interprétés avec prudence, car les modèles ont été entraînés sur 10 époques seulement avec des batchs de taille

30. Cependant, ces essais permettent de donner une bonne indication pour calibrer le modèle et sélectionner des valeurs d'hyperparamètres pertinentes. Le test avec un taux d'apprentissage de 1e-2 a montré les meilleurs résultats en termes de segmentation, avec un coefficient de Dice de 0.5025 et un indice de Jaccard de 0.3658.

4.3.2 Réglage du Nombre de Couches du Modèle

Pour explorer l'impact du nombre de couches (*layers*) sur la performance de notre modèle de segmentation, nous avons testé quatre configurations de modèles avec un nombre de couches variant entre 3 et 11. Ces configurations sont décrites ci-dessous :

- **low_layers_number** : 3 couches
- **middle_low_layers_number** : 5 couches
- **middle_high_layers_number** : 8 couches
- **high_layers_number** : 11 couches

Les résultats des tests sur 10 époques d'entraînement avec des batchs de taille 30 pour chaque configuration montrent que, globalement, l'augmentation du nombre de couches améliore les performances du modèle, mais au prix d'un temps d'entraînement plus long. Les métriques clés pour comparer les modèles incluent la **Test Loss**, le **Dice Coefficient**, et l'**Jaccard Index**.

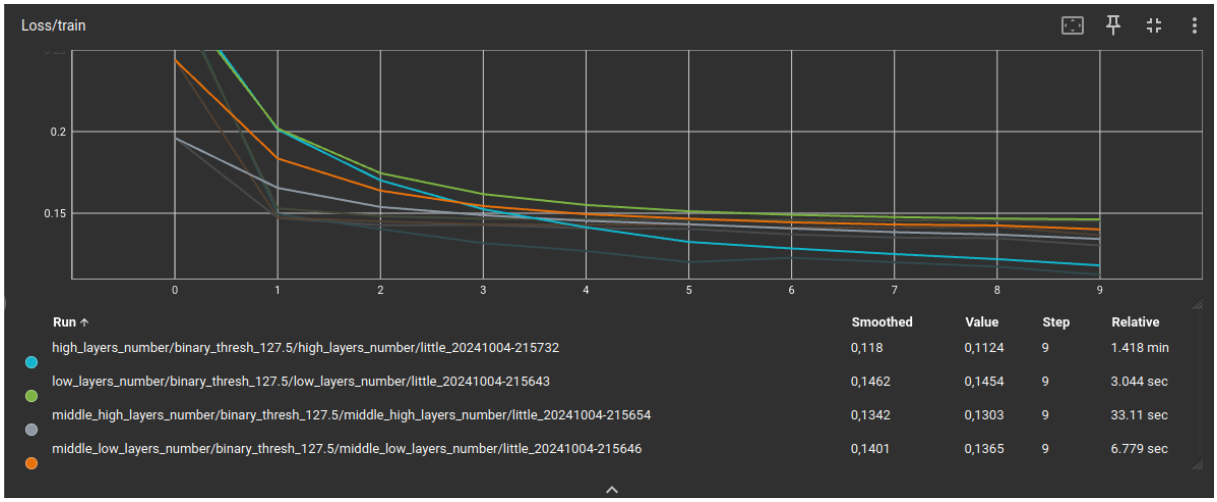

FIGURE 4.5 – Évolution de la perte (*Loss*) au cours de l'entraînement pour différents nombres de couches.

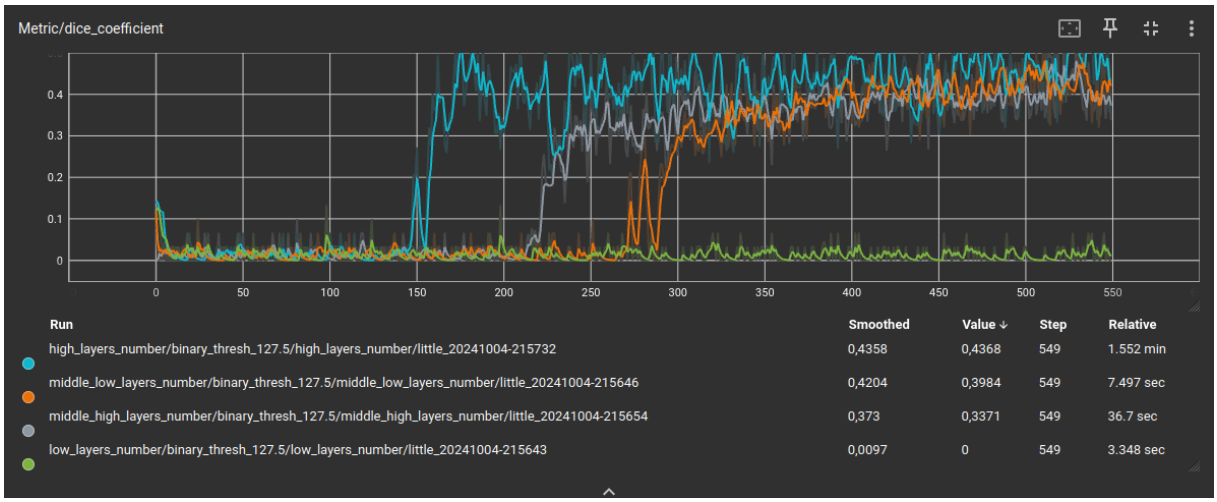

FIGURE 4.6 – Évolution du Dice Coefficient au cours de l'entraînement pour différents nombres de couches.

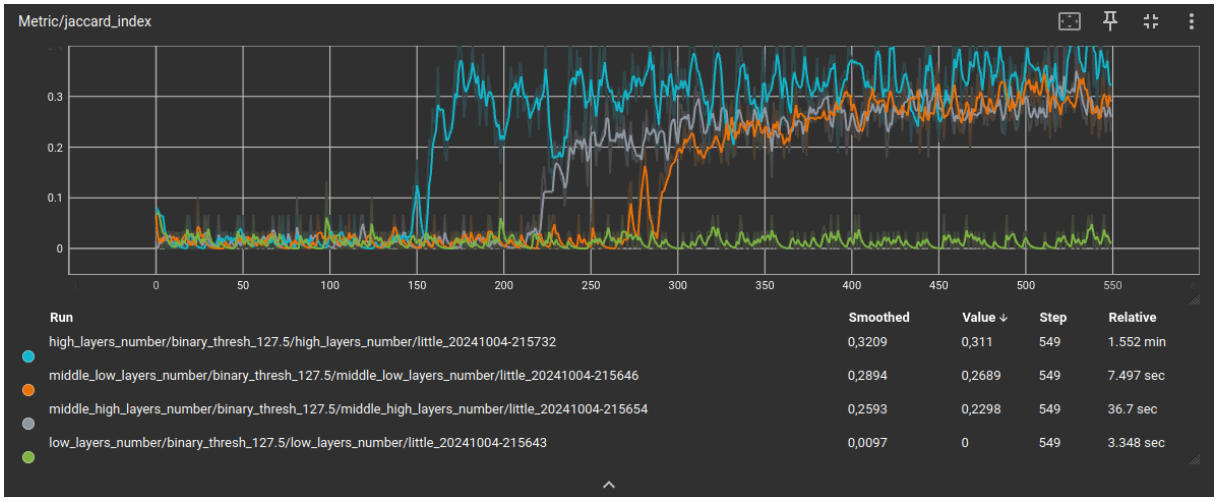

FIGURE 4.7 – Évolution de l'indice de Jaccard au cours de l'entraînement pour différents nombres de couches.

Les résultats obtenus sont les suivants :

Configuration des Couches	Test Loss	Dice Coefficient	Jaccard Index	Temps Total (s)
low_layers_number (3 couches)	0.1371	0.0000	0.0000	3.36
middle_low_layers_number (5 couches)	0.1267	0.4345	0.2986	7.51
middle_high_layers_number (8 couches)	0.1227	0.4675	0.3309	36.76
high_layers_number (11 couches)	0.1224	0.5646	0.4359	93.26

TABLE 4.2 – Comparaison des performances pour différentes configurations de couches.

L'augmentation du nombre de couches améliore les performances du modèle, comme le montrent les gains en termes de *Dice Coefficient* et d'*Indice de Jaccard*, particulièrement marqués pour la configuration `high_layers_number` avec 11 couches. Cependant, cette amélioration s'accompagne d'une augmentation significative du temps d'entraînement, qui

passe de 3.36 secondes pour le modèle à 3 couches à 93.26 secondes pour le modèle à 11 couches.

Ainsi, bien que les modèles plus profonds produisent des résultats meilleurs, leur coût en temps de calcul devient prohibitif dans le contexte de contraintes de temps limitées.

Pour les tests finaux, nous avons décidé de retenir un modèle ayant un nombre de couches compris entre 5 et 9. Ces configurations représentent un compromis acceptable entre performance et temps d'entraînement. Le modèle avec 5 couches, par exemple, obtient un *Dice Coefficient* de 0.4345 en seulement 7.51 secondes, ce qui en fait une option intéressante lorsque les ressources de calcul sont limitées.

4.3.3 Choix de la Taille du Noyau

Un *noyau* (*kernel*) en convolution est une matrice de petite taille utilisée pour balayer l'image d'entrée et effectuer des opérations de filtrage afin d'extraire des caractéristiques importantes. La taille du noyau (*kernel size*) est un paramètre déterminant qui influe sur la capacité du modèle à capturer des détails locaux dans l'image. Par exemple, un noyau de petite taille se concentre sur les détails fins, tandis qu'un noyau plus grand capture des structures plus globales.

Nous avons effectué trois séries de tests en variant la taille du noyau (*kernel size*) afin d'évaluer son impact sur les performances de notre modèle. Les tailles de noyau testées incluent 1, 3, 5, 7 et 9, et les résultats obtenus sont résumés dans le tableau suivant :

Taille du Noyau	Test Loss	Dice Coefficient	Jaccard Index	Temps Total (s)
Exécution 1				
1 (low_kernel_size)	0.1938	0.0000	0.0000	5.55
3 (middle_kernel_size)	0.1187	0.4749	0.3367	7.02
5 (high_kernel_size)	0.1274	0.5638	0.4271	13.42
7 (very_high_kernel_size)	0.1207	0.4898	0.3613	20.35
9 (highest_kernel_size)	0.1248	0.5454	0.4198	35.05
Exécution 2				
1 (low_kernel_size)	0.2309	0.0000	0.0000	4.68
3 (middle_kernel_size)	0.1288	0.4725	0.3302	7.41
5 (high_kernel_size)	0.1112	0.4818	0.3522	12.07
7 (very_high_kernel_size)	0.1191	0.5198	0.3883	19.10
9 (highest_kernel_size)	0.1241	0.4020	0.2897	30.57
Exécution 3				
1 (low_kernel_size)	0.1954	0.0000	0.0000	5.23
3 (middle_kernel_size)	0.1254	0.4255	0.2953	7.11
5 (high_kernel_size)	0.1132	0.5389	0.4106	12.32
7 (very_high_kernel_size)	0.1113	0.4524	0.3305	18.83
9 (highest_kernel_size)	0.2312	0.0000	0.0000	34.62

TABLE 4.3 – Résultats des tests pour différentes tailles de noyau sur trois exécutions indépendantes.

En observant les résultats, nous remarquons que l'influence de la taille du noyau sur les performances n'est pas systématiquement positive. Par exemple, lors de l'exécution 2, la taille de noyau *high_kernel_size* (5) atteint une **Test Loss** de 0.1112, avec un **Dice Coefficient** de 0.4818 et un **Jaccard Index** de 0.3522. En revanche, lors de l'exécution 3 avec la même configuration, bien que la **Test Loss** soit proche (0.1132), le **Dice Coefficient** est plus élevé (0.5389) et le **Jaccard Index** également (0.4106). Cela démontre une certaine variabilité dans les résultats, même pour des tailles de noyau similaires.

De plus, des tailles de noyau plus larges n'ont pas nécessairement conduit à de meilleures performances. Par exemple, la configuration avec la taille de noyau *highest_kernel_size* (9) a montré une dégradation importante des performances lors de l'exécution 3, avec une

Test Loss de 0.2312, un **Dice Coefficient** de 0.0000, et un **Jaccard Index** de 0.0000, tout en nécessitant un temps d'entraînement beaucoup plus long (34.62 secondes).

Cette variabilité et l'absence de preuve claire qu'une taille de noyau plus large améliore systématiquement les performances nous ont conduit à choisir un noyau de taille **3** et **5**. Cette configuration est rapide à entraîner (entre 7 et 7.5 secondes en moyenne) pour une taille 3 et de 12,75 seconde en moyenne pour une taille 5, et dans l'ensemble, elle produit des résultats compétitifs avec un **Dice Coefficient** d'environ 0.4255 à 0.4749 selon les exécutions pour une taille 3 et d'environ 0.4818 à 0.5638 en **Dice Coefficient** pour une taille de kernel 5.

Ainsi, le compromis entre rapidité et performance nous a poussé à retenir une taille de noyau de 3 à 5 pour les expériences finales.

4.3.4 Impact de la normalisation

Lors des premiers tests sans normalisation des données, nous avons observé une divergence très rapide du modèle, conduisant à des pertes absurdes et des résultats incohérents. Comme le montre le tableau ci-dessous, les valeurs de la perte (*Loss*) augmentent de manière exponentielle vers des valeurs négatives extrêmes, tandis que les métriques de performance telles que le *Dice Coefficient* et l'*Indice de Jaccard* donnent des résultats totalement irréalistes, atteignant même des valeurs très élevées (par exemple, un *Dice Coefficient* de 1.8206).

Méthode	**Test Loss**	**Dice Coefficient**	**Jaccard Index**
Sans normalisation	-5.8731e+16	1.8206	15.5419
Avec normalisation	0.1159	0.4926	0.3570

TABLE 4.4 – Comparaison des résultats avec et sans normalisation.

Sans normalisation, le modèle produisait des gradients incontrôlables, conduisant à une divergence rapide pendant l'entraînement, comme en témoigne l'évolution des pertes qui devenaient rapidement négatives et extrêmement grandes. Cela a conduit à des résultats incohérents et non représentatifs des performances réelles du modèle.

Pour stabiliser l'entraînement et éviter la divergence, nous avons donc introduit une étape de normalisation des données. Avec la normalisation, le modèle a produit des résultats beaucoup plus cohérents, avec des valeurs de *Loss*, *Dice Coefficient*, et *Jaccard Index* raisonnables et significatives. Comme le montre le tableau, la normalisation permet au modèle de converger correctement et d'obtenir des résultats valides.

En conclusion, il s'est avéré indispensable de normaliser les données pour empêcher le modèle de diverger et pour garantir un entraînement stable et des résultats fiables. Sans cette étape, le modèle devient instable et donne des résultats aberrants.

4.4 Experience

Dans cette section, nous décrivons les différentes expériences que nous avons effectuées en combinant plusieurs configurations de prétraitement des données, d'architecture de modèle, et de paramètres d'entraînement. Le tableau 4.5 présente les configurations de prétraitement appliquées aux images, le tableau 4.6 détaille les architectures des modèles utilisés, et le tableau 4.7 montre les configurations d'entraînement.

4.4.1 Configurations de prétraitement

Le tableau 4.5 décrit les différentes étapes de prétraitement appliquées aux données avant l'entraînement du modèle.

Nom du Prétraitement	Normalisation	Seuil	Égalisation Histogramme	Flou Gaussien
`binary_thresh_only`	Oui	127.5	Non	Non
`hist_eq_and_thresh`	Oui	127.5	Oui	Non
`full_combo`	Oui	100	Oui	Oui

TABLE 4.5 – Configurations de prétraitement des données utilisées dans les expériences.

4.4.2 Configurations des Modèles

Le tableau 4.6 présente les différentes architectures de modèles que nous avons testées, en variant le nombre de couches et la taille du noyau.

Nom du Modèle	Nombre de Couches	Taille du Noyau
`default_3x3_5_layers`	5	3
`mid_3x3_7_layers`	7	3
`high_3x3_9_layers`	9	3
`default_5x5_5_layers`	5	5
`mid_5x5_7_layers`	7	5
`high_5x5_9_layers`	9	5

TABLE 4.6 – Configurations des modèles utilisés dans les expériences.

4.4.3 Configurations d'Entraînement

Le tableau 4.7 décrit la configuration d'entraînement, incluant le nombre d'époques, la taille des batchs, le taux d'apprentissage et l'optimiseur utilisé.

Nom	Optimiseur	Taux d'Apprentissage	Fonction de Perte	Époques	Taille des Batch
`real`	Adam	1e-2	BCEWithLogitsLoss	300	50

TABLE 4.7 – Configurations d'entraînement utilisées dans les expériences.

4.4.4 Pipeline d'entrainement

Dans ces expériences, nous avons combiné chaque modèle avec chaque méthode de prétraitement, et chaque configuration a été entraînée en utilisant les paramètres d'entraînement `real` (300 époques, taille de batch de 50, taux d'apprentissage de 1e-2, optimiseur Adam, et fonction de perte BCEWithLogitsLoss).

Chaque modèle a été entraîné avec chaque configuration de prétraitement, soit un total de 18 combinaisons (`6 modèles` × `3 prétraitements`). Ces tests nous ont permis d'évaluer l'impact de différentes architectures et techniques de prétraitement sur les performances globales du modèle, mesurées à travers les métriques *Test Loss*, *Dice Coefficient*, et *Jaccard Index*.

4.5 Résultats

Dans cette section, nous présentons les résultats des trois meilleurs modèles identifiés au cours des expériences. Ces modèles ont été évalués sur la base des coefficients de Dice et de l'indice de Jaccard, après avoir appliqué un lissage de 0,6 via TensorBoard pour stabiliser les courbes d'entraînement. Chaque modèle a été testé avec un prétraitement particulier et les résultats sont indiqués dans le tableau 4.8. Les meilleurs résultats ont été obtenus avec les configurations de prétraitement et de modèle suivantes :

Modèle	Prétraitement	Dice (lissé)	Jaccard (lissé)	Étape	Temps d'Entraînement
`5x5_7_layers`	full_combo	0.7779	0.6846	9 899	18.19 min
`5x5_7_layers`	hist_eq_and_thresh	0.7716	0.6751	9 899	17.55 min
`3x3_9_layers`	binary_thresh_only	0.7550	0.6389	9 899	28.91 min

TABLE 4.8 – Les trois meilleurs modèles testés avec différents prétraitements. Les résultats sont lissés avec un facteur de 0,6 sur TensorBoard.

Le modèle ayant obtenu les meilleurs résultats globaux est `full_combo_mid_5x5_7_layers`, avec un coefficient de Dice lissé de 0,7779 et un indice de Jaccard lissé de 0,6846. Ce modèle a été entraîné avec le prétraitement `full_combo`, qui inclut la normalisation, l'égalisation de l'histogramme et le flou gaussien, et a montré des performances stables avec un temps d'entraînement raisonnable de 18,19 minutes.

Le deuxième meilleur modèle est `hist_eq_and_thresh_mid_5x5_7_layers`, qui a obtenu un coefficient de Dice lissé de 0,7716 et un indice de Jaccard lissé de 0,6751. Ce modèle a été testé avec le prétraitement `hist_eq_and_thresh`, où seule l'égalisation de l'histogramme et le seuillage binaire ont été appliqués.

Le troisième modèle est `binary_thresh_only_high_3x3_9_layers`, qui a obtenu des résultats légèrement inférieurs avec un coefficient de Dice lissé de 0,7550 et un indice de Jaccard lissé de 0,6389. Ce modèle a été testé avec le prétraitement `binary_thresh_only`, qui consiste en une normalisation basique suivie d'un seuillage binaire.

En conclusion, le meilleur modèle testé est `full_combo_mid_5x5_7_layers`, qui utilise un prétraitement complet incluant l'égalisation d'histogramme et le flou gaussien. Ce modèle a montré des résultats supérieurs en termes de précision et de temps de convergence.

4.5.1 Visualisation des courbes d'entraînement

Les courbes d'entraînement des trois meilleurs modèles sont présentées dans les graphiques ci-dessous. Ces courbes montrent l'évolution de la *loss*, du *Dice coefficient* et de l'*indice de Jaccard* au cours des étapes d'entraînement.

Les courbes montrent que le modèle `full_combo_mid_5x5_7_layers` a une convergence plus rapide et plus stable, avec de meilleures performances globales sur les métriques de Dice et de Jaccard, par rapport aux deux autres modèles.

4.5.2 Visualisation des résultats des tests

En plus des courbes d'entraînement, nous présentons également les résultats visuels des tests pour les trois meilleurs modèles. Ces graphiques montrent la comparaison entre les prédictions du modèle et les véritables masques de segmentation. Les résultats sont visualisés pour des exemples sélectionnés dans les tests finaux.

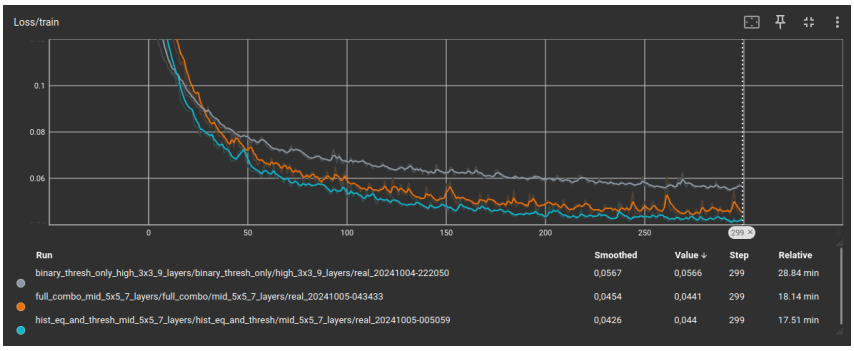

FIGURE 4.8 – Courbe de la perte (Loss) pour les trois meilleurs modèles.

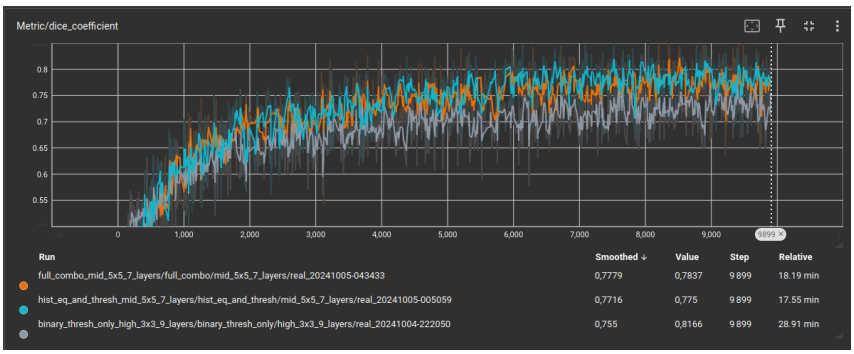

FIGURE 4.9 – Courbe du coefficient de Dice pour les trois meilleurs modèles.

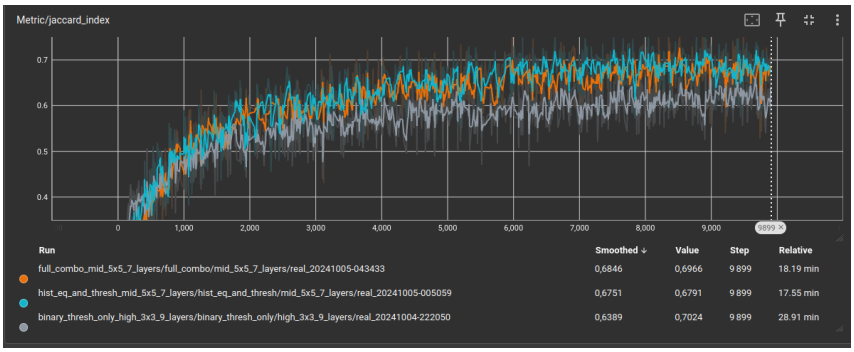

FIGURE 4.10 – Courbe de l'indice de Jaccard pour les trois meilleurs modèles.

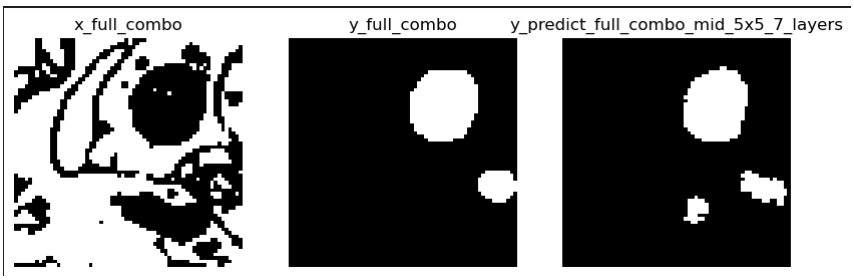

FIGURE 4.11 – Résultats des tests pour le modèle `full_combo_mid_5x5_7_layers`. La première image montre l'image d'entrée, la seconde la prédiction du modèle et la troisième le masque réel.

Ces visualisations montrent la qualité des prédictions faites par les modèles. Le modèle

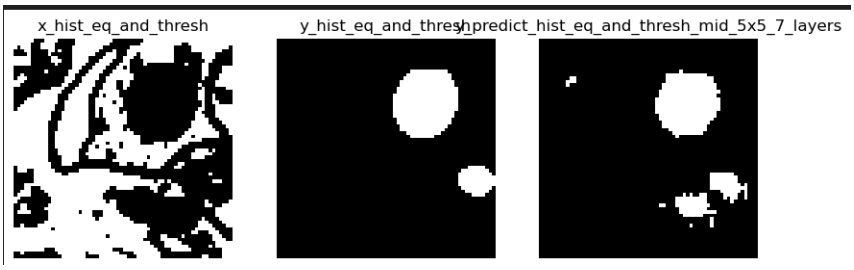

FIGURE 4.12 – Résultats des tests pour le modèle `hist_eq_and_thresh_mid_5x5_7_layers`. La première image montre l'image d'entrée, la seconde la prédiction du modèle et la troisième le masque réel.

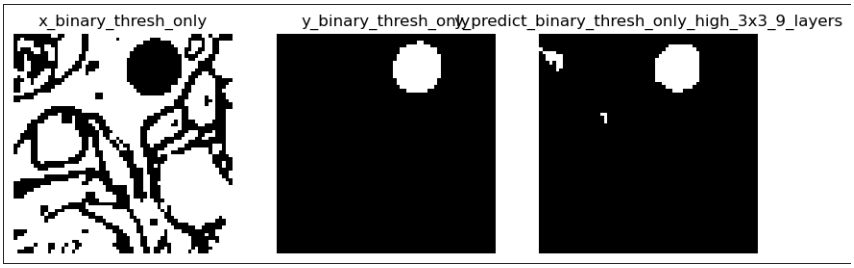

FIGURE 4.13 – Résultat d'un test pour le modèle `binary_thresh_only_high_3x3_9_layers`. La première image montre l'image d'entrée, la seconde la prédiction du modèle et la troisième le masque réel.

`full_combo_mid_5x5_7_layers` produit des masques de segmentation qui correspondent étroitement aux masques réels, confirmant ainsi ses bonnes performances observées.

Chapitre 5

Conclusions/Perspectives

Les résultats de nos expériences montrent clairement que le prétraitement des données a un impact significatif sur les performances du modèle. Le fait d'avoir utilisé le prétraitement comme hyperparamètre s'est avéré pertinent, permettant de trouver des configurations qui optimisent les résultats. En particulier, l'utilisation de l'autoencoder convolutionnel a fourni des résultats prometteurs, grâce à sa structure en *encoder-decoder*, qui est parfaitement adaptée aux tâches de segmentation. Cette architecture a permis de capturer les caractéristiques importantes des images et de reconstruire efficacement les masques de segmentation.

Nous avons réussi à atteindre nos objectifs initiaux en identifiant un modèle qui répond à nos attentes, avec un score supérieur à 0,4 pour l'indice de Jaccard et 0,6 pour le coefficient de Dice, tout en respectant la contrainte de temps d'entraînement (moins de 2 heures). Le modèle `full_combo_mid_5x5_7_layers` s'est révélé être le plus performant avec ces critères.

Cependant, certaines limites de cette étude méritent d'être soulignées. D'abord, une exploration plus approfondie des différentes techniques de prétraitement, en particulier des méthodes adaptées à la nature spécifique des mitochondries, aurait pu améliorer encore davantage les résultats. Par exemple, l'utilisation de techniques de normalisation entre 0 et 1 n'a pas été testée, et nous avons opté systématiquement pour une binarisation afin de faciliter la convergence. Cela peut être une piste d'amélioration pour les futurs travaux.

La contrainte de temps d'entraînement a également représenté un défi, notamment parce qu'elle nous a poussé à réduire la résolution des images par downsampling. Cela a entraîné une perte de détails cruciaux, en particulier les canaux mitochondriaux visibles dans les images d'origine, qui caractérisent la structure complexe des mitochondries. En réduisant la résolution, ces canaux disparaissent, et cela a probablement affecté la qualité des prédictions du modèle. Les mitochondries ne sont pas simplement des boules grises, mais possèdent des structures internes fines qui ont été partiellement perdues lors du processus de downsampling.

De plus, l'utilisation d'un jeu de test à chaque fois pour évaluer les performances du modèle constitue une autre limite. Un jeu de calibration dédié aurait permis de mieux ajuster les hyperparamètres et de garantir une évaluation plus juste des performances du modèle. Cette approche aurait évité le risque de surévaluation des résultats sur le jeu de test.

En somme, cette étude a démontré l'efficacité des autoencoders convolutionnels pour la tâche de segmentation des mitochondries et a mis en lumière l'importance du choix du prétraitement. Les résultats obtenus montrent que nous avons atteint nos objectifs de

performance et de temps d'entraînement, bien que des pistes d'amélioration subsistent, notamment concernant la résolution des images et l'utilisation d'une calibration dédiée.

Bibliographie

[1] Lenka Backová. Segmentation of multi-dimensional multi-parametric microscopic data of biological samples using convolutional neural networks. Master's thesis, Charles University, Prague, Czech Republic, August 2021. Supervisor : Mgr. Aleš Benda, PhD.

[2] Róger Bermúdez-Chacón, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. A domain-adaptive two-stream u-net for electron microscopy image segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 620–623, Washington, DC, USA, April 2018. IEEE.

[3] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[4] Manca Žerovnik Mekuč, Ciril Bohak, Samo Hudoklin, Byeong Hak Kim, Rok Romih, Min Young Kim, and Matija Marolt. Automatic segmentation of mitochondria and endolysosomes in volumetric electron microscopy data. *Computers in Biology and Medicine*, 119 :103693, April 2020.

[5] Chi Xiao, Xi Chen, Weifu Li, Linlin Li, Lu Wang, Qiwei Xie, and Hua Han. Automatic mitochondria segmentation for EM data using a 3d supervised convolutional network. *Frontiers in Neuroanatomy*, 12 :92, November 2018.