Université de Strasbourg

École doctorale : **Mathématiques, Sciences de l'Information et de l'Ingénieur**

Unité de recherche : **ICube – UMR 7357**

THÈSE

Présentée par : **Florian Allender**

Soutenue le : **30 Novembre 2022**

Pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/Spécialité : **Informatique**

Augmentation de données pour l'analyse d'images histopathologiques : approches par génération d'images et déformations spatiales pour la segmentation de glomérules

Thèse dirigée par :

M. WEMMERT Cédric	Professeur des universités, Université de Strasbourg
M. DISCHLER Jean-Michel	Professeur des universités, Université de Strasbourg

Rapporteurs :

M. KURTZ Camille	Maître de conférences, HDR, Université de Paris Cité
M. PASSAT Nicolas	Professeur des universités, Université de Reims Champagne-Ardenne

Examinateurs :

M. NAEGEL Benoît	Professeur des universités, Université de Strasbourg
Mme. VAKALOPOULOU Maria	Maîtresse de conférences, Université de Paris-Saclay

Invité :

M. ALLÈGRE Rémi	Maître de conférences, Université de Strasbourg

Résumé

Dans le cadre de cette thèse, nous nous intéressons à des données histopathologiques rénales. L'histopathologie rénale est cruciale dans l'étude des maladies du rein, et en particulier du rejet de greffes, qui se produit avec un taux d'incidence de 7.9% pendant la première année. Pour étudier ces pathologies, les médecins utilisent les outils issus de l'histopathologie numérique pour analyser les glomérules, structures biologiques chargées du filtrage du sang et de la production de l'urine. Ces structures sont complexes et comportent de multiples sous-structures (membranes, capillaires, cellules mésangiales et endothéliales, podocytes) rendant leur segmentation automatique particulièrement difficile. Notre objectif est d'améliorer la segmentation automatique de glomérules dans des coupes complètes en utilisant un CNN de type U-Net, modèle standard en segmentation d'images médicales. L'entraînement d'un tel modèle nécessite une grande quantité d'images annotées (plusieurs dizaines de milliers). Or, dans notre contexte, le nombre d'images annotées disponibles est de l'ordre de quelques centaines seulement, ce qui pose la question des augmentations de données. Nous proposons dans cette thèse d'étudier l'application et l'impact d'augmentations de deux types. Nous étudions premièrement les variations géométriques, introduites à l'aide de déformations spatiales aléatoires. Bien que l'utilisation de déformations aléatoires soit classique pour l'augmentation de données, aucune évaluation n'a été proposée dans la littérature pour les images histopathologiques, et pour les images de glomérules en particulier. Deuxièmement, nous étudions les variations de texture, introduites à l'aide de méthodes de synthèse de texture et de modèles génératifs, et qui n'ont fait l'objet que de très peu de travaux en imagerie histopathologique rénale.

Nous adaptons des méthodes existantes aux caractéristiques des images de glomérules, et développons des cadres de comparaison et d'évaluation des différentes méthodes dans le contexte d'un nombre restreint d'images annotées réelles disponibles. Nous montrons que le choix et le paramétrage des augmentations est crucial pour tirer le meilleur bénéfice de chaque augmentation. Nous mettons également en évidence qu'entre les images issues des méthodes d'augmentation et les images réelles, seule une proximité visuelle jusqu'à un certain point est nécessaire. Les travaux futurs s'attacheront à préciser cette limite et à étudier l'introduction de variations supplémentaires, notamment d'ordre topologique, en particulier à l'aide de méthodes procédurales.

Abstract

In this thesis, we are interested in renal histopathological data. Renal histopathology is crucial in the study of kidney diseases, and in particular that of transplant rejection, which occurs with an incidence rate of 7.9% in the first year. Pathologists use tools from digital histopathology to analyze glomeruli, i.e. biological structures responsible for blood filtration and urine production. These structures are complex and include multiple substructures (membranes, capillaries, mesangial and endothelial cells, podocytes) making their automatic segmentation particularly difficult. Our objective is to improve the automatic segmentation of glomeruli in whole slide images using a CNN called U-Net, a standard model in medical image segmentation. Training such a model requires a large amount of annotated images (several tens of thousands). However, in our context, the number of available annotated images is of the order of a few hundreds, which raises the question of data augmentation. This thesis investigates the application and the impact of two types of augmentation techniques. We first study geometric variations, introduced using random spatial deformations. Although the use of random deformations is standard for data augmentation, no evaluation has been proposed in the literature for histopathological images, including glomeruli images. Second, we study texture variations, introduced using texture synthesis methods and generative models, on which very little work has been carried out so far in renal histopathological imaging.

We adapt existing methods to the characteristics of glomeruli images, and develop frameworks for comparing and evaluating the different methods in the context of a limited number of real annotated images available. We show that the choice and configuration of augmentations is crucial to get the most benefit from each augmentation. We also highlight that between the images resulting from the methods of augmentation and the real images, only a visual proximity up to a certain point is necessary. Future work will endeavor to specify this limit and to study the introduction of additional variations, in particular of a topological order, using procedural methods.

Remerciements

Je remercie tout d'abord Camille Kurtz, Benoît Naegel, Nicolas Passat et Maria Vakalopoulou pour avoir accepté de constituer mon jury de thèse. J'attends avec impatience leurs retours sur ce manuscrit.

Je remercie également Cédric Wemmert et Jean-Michel Dischler, mes directeurs de thèse, pour la confiance qu'ils ont placé en moi durant ces travaux de recherche. Je remercie bien sûr mon encadrant Rémi Allègre, pour tout le travail et son accompagnement au quotidien, ainsi que son aide lors des moments chauds. Il n'est pas toujours évident de voir où l'on va lorsque l'on a la tête dans le guidon, mais tu m'as bien appris à prendre le recul nécessaire et à rester précis et rigoureux.

Je tiens à remercier les équipes administratives du laboratoire ICube et de l'UFR de Maths-Info, qui font un travail formidable dans la bienveillance et la bonne humeur pour accompagner les chercheurs dans leurs démarches. Ce qui est habituellement une plaie devient agréable lorsque tout se fait avec le sourire. Je remercie aussi tous mes collègues de l'équipe IGG et de l'UFR pour leur accueil et leur accompagnement tout au long de ces quatre dernières années. J'ai rencontré des personnes formidables et c'était un réel plaisir de travailler avec elles.

Plus particulièrement, je remercie mes camarades thésards, les plus anciens comme les petits derniers : Nicolas, Katia, Melinda, Paul, Quentin, David, Geoffrey, Alex, Pascal, Charline, Guillaume. Merci pour les moments de détente et le serrage de coudes. Le Time's Bomb restera à jamais associé avec les pauses du midi au labo ! Malheureusement le Covid a mis à mal une bonne dynamique de groupe, et je regrette de ne pas avoir pu profiter de la même manière avec les arrivants de ces deux dernières années et de ne pas les connaître aussi bien qu'ils le méritent sûrement.

Je remercie évidemment le Skook's Crew : Skook, Ramistar, Hengistt, Barbemoche et Iso, mes chers gauchistes sans étiquette, mes camarades de promo partis aux quatre coins de la France (ou pas). Ensemble nous avons refait le monde, partagé nos passions ainsi que nos frustrations. La vie serait plus dure sans vous les gars.

Et enfin, je remercie Claire, ma moitié, pour son soutien malgré des années difficiles pour elle aussi. Cette thèse, tu l'auras vécue avec moi, les doutes, les prises de tête et la

mauvaise humeur parfois quand j'avais du mal à laisser le travail de côté. Tu donnes de ta personne pour que tout se passe bien pour moi, encore en ce moment, alors que je finalise l'écriture de ce manuscrit. Merci d'être là, et merci pour le merveilleux cadeau que tu m'as fait cette nuit du 23 février 2022.

Table des matières

Table des figures

Table des tableaux

Chapitre 1

Introduction

1.1 Contexte et problématique

1.1.1 Histopathologie numérique

L'histopathologie [His] est l'étude microscopique de tissus, en général prélevés lors d'une biopsie ou d'une chirurgie, dans le but d'identifier d'éventuelles maladies comme le cancer. Les prélèvements, après leur préparation pour conservation, sont découpés en fines tranches, de l'ordre de 2 à 7 micromètres d'épaisseur. Les tranches sont disposées sur des lames en verre et traitées à l'aide de colorants tels que l'hématoxyline ou l'éosine, qui permettent de faire apparaître différentes structures biologiques, comme par exemple les noyaux des cellules. La figure 1.1 montre des exemples de lames histologiques. Les pathologistes étudient les lames au moyen de microscopes électroniques. Ces dernières peuvent également être numérisées en très haute résolution, ce qui permet d'obtenir des images de plusieurs milliers de pixels dans chaque dimension (*Whole Slide Image, WSI*, ou coupe complète). Des outils numériques peuvent ainsi venir assister les pathologistes, leur permettant de naviguer plus facilement dans la tranche, de zoomer ou de poser des annotations sur des régions d'intérêt. Cytomine [MRS+16] est une application web open-source permettant de tels traitements, comme illustré par la figure 1.2.

1.1.2 Étude des pathologies rénales

D'après l'Agence de la Biomédecine [ABM], au 1er janvier 2019, 15 256 patients étaient inscrits en liste d'attente pour une greffe de rein, dont 5 545 nouveaux inscrits l'année précédente. Sur ces patients, 3 643 recevront effectivement une greffe, et 427 décéderont dans l'année. La Haute Autorité de santé [HAS] indique qu'il n'y a qu'un greffon de disponible par malade. Elle indique également que la durée de vie d'un greffon rénal est en

Figure 1.1 : Lames histologiques (source : CHU de Poitiers [CHU]).

moyenne de 12,4 ans, après quoi le patient doit de nouveau subir une transplantation. Face à ces chiffres, il apparaît que pour maximiser les chances de survie du plus grand nombre de patients, il faut pouvoir s'assurer que les greffes tiennent le plus longtemps possible, et notamment dans les premières semaines, durant lesquelles le risque de rejet est le plus important (10% des greffes).

Les médecins se concentrent notamment sur le rejet chronique de la greffe, dû à certaines maladies en particulier : la fibrose interstitielle et l'atrophie tubulaire (*Interstitial Fibrosis and Tubular Atrophy : IF/TA*), et la glomérulosclérose. Pour modéliser et prédire l'évolution du rejet [NS21], les pathologistes disposent d'images histopathologiques, issues de prélèvements rénaux observés au microscope électronique, telle que celle présentée en figure 1.3. Cette image est de taille 95 612 par 72 292 pixels, ce qui représente plusieurs dizaines de giga octets de données. Différentes colorations permettent de faire apparaître différentes structures dans les images et sont généralement étudiées de manière complémentaire. Pour l'ensemble des travaux de cette thèse, nous nous limitons à l'étude d'images en coloration PAS (acide périodique de Schiff, *Periodic Acid Schiff*).

Les structures d'intérêt dans ces images sont les glomérules, des amas de vaisseaux permettant la filtration du plasma sanguin et la production d'urine primitive. Les glomérules sont des structures en forme de boule composées de plusieurs sous-structures : membranes, capillaires, cellules mésangiales et endothéliales, podocytes, etc. La figure 1.4 montre une représentation schématique d'un glomérule et met en évidence les différentes sous-structures. L'apparence des glomérules peut présenter une grande variabilité dans les WSIs, comme illustré par la figure 1.5, due à la section, à la coloration, aux patients, aux procédures dans les laboratoires et/ou appareils de microscope et d'imagerie.

L'apparence des glomérules varie également en raison des changements morphologiques induits par les pathologies, comme par exemple des glomérules sclérosés, ce qui permet

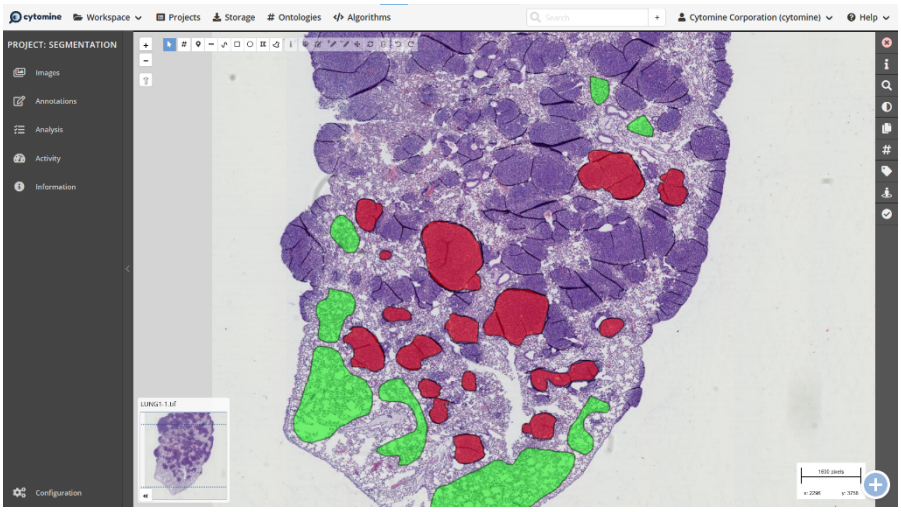


Figure 1.2 : Interface de Cytomine [MRS+16], une application web open-source permettant le traitement d'images histopathologiques (source : Pôle Mecha-Tech [Pol]).

d'étudier les maladies précédemment citées. La détection et la segmentation de glomérules dans des coupes complètes représentent donc un enjeu majeur. Ces opérations sont effectuées manuellement par les pathologistes.

1.1.3 Traitement automatique d'images histopathologiques

Automatiser le traitement des images histopathologiques permet d'assister les praticiens en fournissant des données fiables et moins soumises à l'appréciation humaine subjective, permettant ainsi des diagnostics plus rapides et plus robustes. Pour ce faire, la recherche en analyse d'images fournit des algorithmes capables d'effectuer automatiquement différentes tâches telles que la classification, la détection ou la segmentation. La classification consiste à attribuer une classe à une image, par exemple « sain » ou « tumeur ». La détection a pour objectif de déterminer la présence ou non d'un objet ainsi que sa position, éventuellement sous la forme d'une boîte englobante. La segmentation a pour objectif de détourer précisément un ou plusieurs objets d'intérêt dans une image, ce qui revient à faire une classification pour chaque pixel. La figure 1.6 montre des exemples de résultats de détection et de segmentation sur un patch de coupe complète contenant un glomérule. Les algorithmes de *machine learning*, ou d'apprentissage automatique, offrent les meilleures performances par rapport aux approches à base de descripteurs et aux méthodes de traitement d'images classiques. Ils permettent d'automatiser ces tâches, en travaillant directement sur les données. Ils procèdent en deux étapes : *l'apprentissage*, ou *entraînement*, et l'*application*, ou *test*. L'apprentissage s'effectue directement sur les données disponibles, dont des caractéristiques (*features*) ont été éventuellement extraites par des méthodes de traitement d'images classiques, par essai-erreur : le modèle fait une prédiction sur les données d'entraînement, puis se corrige grâce à une certaine fonction de coût à optimiser. La phase de test permet

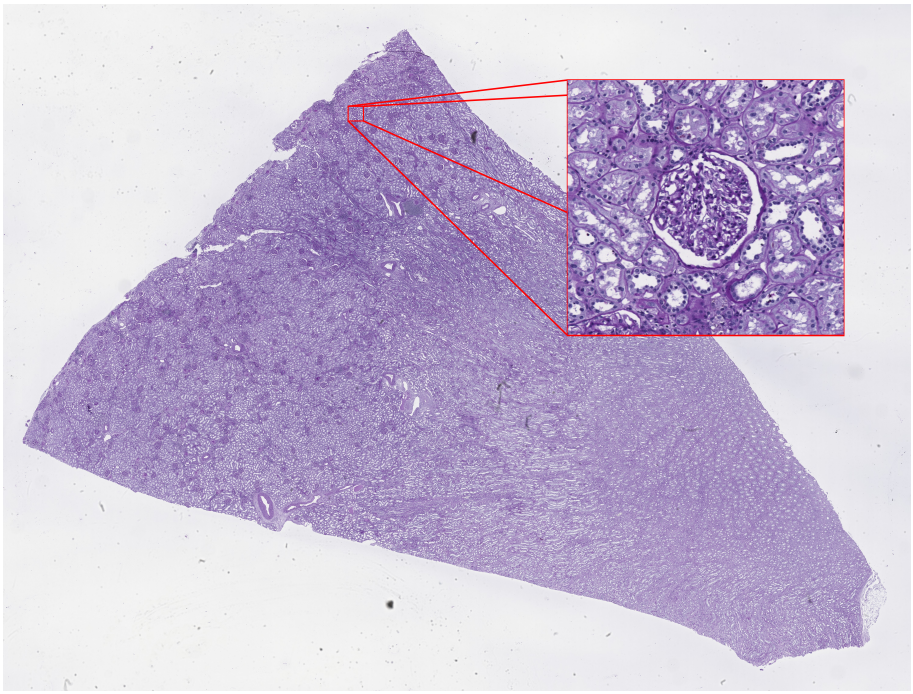


Figure 1.3 : Exemple d'image de coupe complète, issue de l'observation d'un prélèvement rénal.

d'évaluer les performances du modèle sur des nouvelles données, non vues durant l'entraînement. Il existe deux grandes familles d'algorithmes d'apprentissage automatique : les algorithmes supervisés, qui traitent des données *annotées*, c'est-à-dire associées avec une vérité terrain, et les algorithmes non supervisés, traitant des données sans annotation, effectuant du partitionnement (*clustering*). Dans le cas des algorithmes supervisés, le modèle apprend en comparant la réponse proposée et la vérité terrain attendue. Dans notre cas, les données sont les coupes complètes, ou des patches issus de ces images, et la vérité terrain une segmentation manuelle.

Parmi les méthodes d'apprentissage automatique, l'apprentissage profond (*Deep Learning*) est actuellement la plus employée [YTT19]. Les réseaux de neurones artificiels, composés d'unités très simples effectuant des sommes pondérées de leurs entrées, permettent d'après le théorème d'approximation universelle [Cyb89] de modéliser n'importe quelle fonction grâce à leur organisation hiérarchique s'inspirant des neurones humains. L'apprentissage s'effectue en ajustant de manière itérative les poids du réseau. En particulier, l'étape d'extraction de caractéristiques évoquée précédemment est effectuée directement par le réseau, et l'organisation en couches permet d'apprendre des représentations de plus en plus haut niveau. Le Deep Learning offre donc un niveau d'automatisation supplémentaire. Après d'importantes avancées techniques ayant rendu possible leur utilisation, les réseaux de neurones convolutifs (*Convolutional Neural Networks*, CNNs) ont révolutionné le domaine du traitement d'images, en commençant par la tâche de classification [KSH12; SZ15; HZRS16], surpassant tous les modèles précédents, ainsi que l'humain

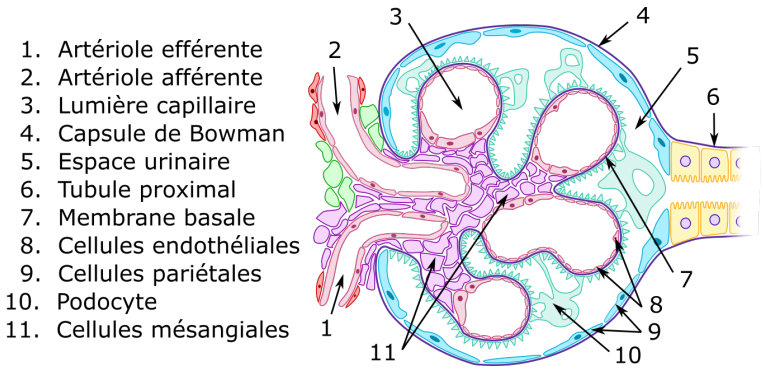


Figure 1.4 : Représentation schématique d'un glomérule (source : Wikipédia [Glo]).

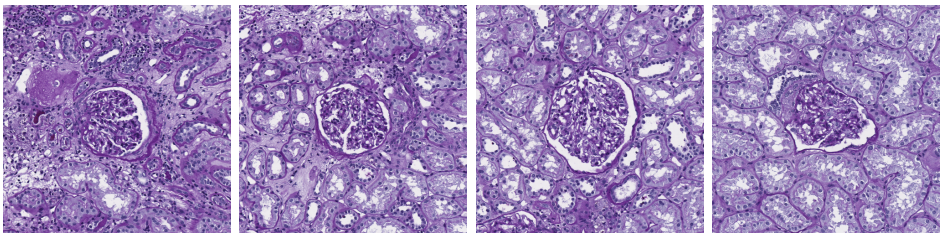


Figure 1.5 : Quatre exemples de coupes de glomérules.

même [MKS+15; HZRS16]. Les CNNs remplacent les neurones précédemment utilisés par des filtres de convolution partagés sur une couche, qui seront appris durant l'entraînement. Le nombre de paramètres sur une couche du réseau est donc beaucoup plus faible que précédemment, ce qui permet en contrepartie d'augmenter le nombre de couches, et donc d'obtenir des modèles capables d'apprendre des représentations complexes. Le Deep Learning s'est imposé comme un standard dans toutes les applications liées aux images médicales, et en particulier les images histopathologiques [LST+16; LKB+17; KI18; DAC19; WWFF21] — voir Srinidhi *et al.* [SCM21] pour une revue complète. En effet, les modèles sont capables d'identifier automatiquement des structures microscopiques, ce qui représente une aide significative pour l'étude des images histopathologiques. Toutefois, ils ont besoin de grandes quantités de données, de plus annotées dans le cadre d'un algorithme supervisé, et de bonne qualité, pour pouvoir *généraliser*, c'est à dire appliquer les connaissances apprises à de nouvelles données. En pratique, les modèles appliqués sur des images générales sont entraînés avec des bases de données comprenant des milliers, voire des millions d'images. Ce pré-requis peut se révéler être un frein en histopathologie numérique, notamment pour les maladies avec peu d'échantillons disponibles. En effet, la collecte et la préparation des données, et notamment leur annotation, est chronophage et coûteuse, car elle nécessite l'intervention d'experts et d'équipements spécifiques. De plus, des considérations de confidentialité empêchent les laboratoires et hôpitaux de mettre en commun certaines ressources, même si des initiatives essayent d'aller en ce sens [Hul20].

Pour répondre à la problématique précédente, toutes les méthodes utilisant du *Deep*

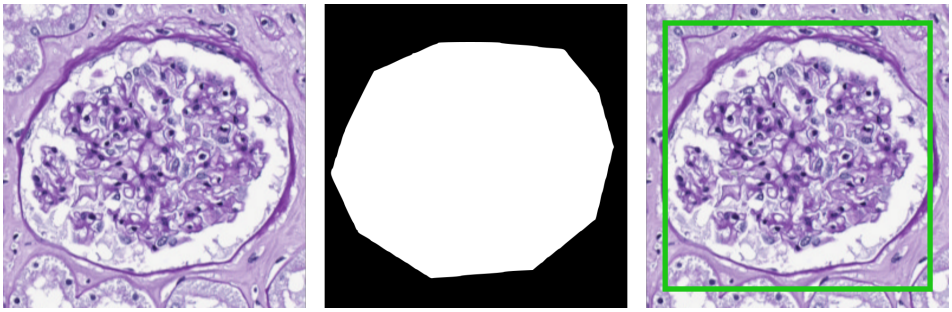


Figure 1.6 : Exemple de segmentation (au milieu) et de détection (à droite) de glomérule, effectuée à la main.

Learning font appel à des techniques d'augmentation de données, qui visent à agrandir artificiellement les jeux de données pendant l'entraînement. Shorten *et al.* [SK19] ont effectué un étude complète de toutes les grandes familles de méthodes existantes, résumée à la figure 1.7. Deux approches se dégagent. La première consiste à effectuer diverses manipulations sur les images, c'est-à-dire les techniques classiques, comprenant les transformations affines (rotations, symétries, translations, mises à l'échelle), les déformations spatiales, ainsi que des modifications de contraste, des ajouts de bruits, des occultations, etc. La deuxième approche réunit des méthodes par Deep Learning : l'entraînement adversaire, qui ne sera pas étudié ici, et les méthodes génératives, comprenant les approches par transfert de style et réseaux génératifs adversaires. Le *méta-apprentissage* permet ensuite de sélectionner et optimiser les paramètres des méthodes d'augmentation parmi les options disponibles.

Pour les images histopathologiques, les augmentations les plus utilisées comprennent les déformations élastiques, et des transformations spécifiques liées aux variations de coloration [VEFD19; XDVF+19; MAdMB+20; FvdLL21]. Une approche en forte croissance ces dernières années consiste à utiliser des données synthétiques, créées spécialement pour élargir les bases d'entraînement. Ces données peuvent être issues de moteurs graphiques [TPA+18], ou de modèles génératifs tels que les GANs [YSH18] (*Generative Adversarial Networks*). Ces derniers ont été largement adoptés dans la communauté médicale, comme l'attestent Yi *et al.* [YWB19], qui reviennent sur toutes les méthodes par GANs utilisées dans le domaine. La synthèse d'images pour l'augmentation de données est en particulier une voie d'exploration pour le traitement d'images histopathologiques [AFF+15; ASN+16]. Les mêmes problématiques de besoin en données vont se poser [BLH+20] pour l'étude des pathologies rénales. La plupart des méthodes de segmentation et de détection de glomérules proposées dans la littérature utilisent des déformations élastiques [dBHS+18; MLS+21; DLX+21], sans justification au sujet des méthodes ni des paramètres utilisés, ainsi que du transfert de coloration [VFWL21a], avec des évaluations portant sur l'impact pour la segmentation.

Outre l'augmentation de données, d'autres techniques permettent d'entraîner des modèles de Deep Learning dans un cadre où peu de données sont disponibles. Le *Transfer Learning* [MAKM22] consiste à utiliser les connaissances apprises sur un jeu de données à un autre jeu de données, par exemple en utilisant un CNN pré-entraîné, en ajustant si

Figure 1.7 : Panorama des techniques d'augmentation de données [SK19].

besoin les couches les plus profondes seulement. Cette approche a déjà été utilisée sur des images histopathologiques [ATAO+22], et pour la segmentation de glomérules en particulier [BFCGLD20; SMG+21]. Le *Few Shot Learning* [WYKN20] permet d'entraîner des modèles dans des cas extrêmes où seulement quelques images sont disponibles, de l'ordre de une à vingt. Ce type d'approche est applicable au traitement d'images histopathologiques [MPS+19] mais concerne actuellement uniquement les modèles de classification.

1.2 Objectif et contributions

Nous considérons dans ce manuscrit la segmentation de glomérules dans des patches issus de coupes complètes, à l'aide d'un CNN de type U-Net [RFB15], modèle standard en segmentation d'images médicales. Notre jeu de données est composé de 10 coupes complètes de tissus rénaux provenant de 10 patients différents. Les 10 coupes sont annotées, chaque pixel ayant un label associé parmi « fond », c'est-à-dire le fond de la lame où le prélèvement rénal est totalement absent, « tissu », c'est-à-dire la région correspondant au prélèvement mais n'étant pas un glomérule, et « glomérule ». En pratique, nous fusionnons les labels fond et tissu, pour nous concentrer sur un problème à deux classes, où chaque pixel est associé à « glomérule » ou à « non glomérule ». Nous pouvons extraire de chaque coupe des patches centrés sur un glomérule et des patches sans glomérule, que nous utiliserons pour les entraînements de segmentation.

La variabilité induite par la forme des glomérules, couplée au faible nombre de données annotées disponibles, rend la détection et la segmentation robustes des glomérules difficile. Notre objectif est d'améliorer la segmentation automatique de glomérules dans des patches issus de coupes complètes en utilisant un CNN de type U-Net. L'entraînement d'un tel modèle nécessite une grande quantité d'images annotées. Or, dans notre contexte, le nombre de patches annotés disponibles est de 2 340, ce qui pose la question des augmentations de données. Nous proposons dans cette thèse d'étudier l'application et l'impact d'augmentations de deux types. Nous étudions premièrement les variations géométriques, introduites à l'aide de déformations spatiales aléatoires. Bien que l'utilisation de déformations aléatoires soit classique pour l'augmentation de données, aucune évaluation n'a été proposée dans la littérature pour les images histopathologiques, et pour les images de glomérules en particulier. Deuxièmement, nous étudions les variations de texture, introduites à l'aide de méthodes de synthèse de texture et de modèles génératifs, et qui n'ont fait l'objet que de très peu de travaux en imagerie histopathologique rénale. Nos contributions consistent en l'utilisation et l'adaptation pour nos images de méthodes issues de l'état de l'art en déformation aléatoire, synthèse de texture et synthèse d'images, et de la quantification de l'impact de ces méthodes sur l'entraînement d'un modèle de segmentation de type U-Net.

Les travaux présentés dans ce manuscrit ont fait l'objet de deux communications avec comité de lecture : la première dans le journal *Computer Methods and Programs in Biomedicine* [AAWD22b], et la seconde à la conférence internationale IEEE-BHI-BSN. [AAWD22a]

1.3 Plan

Après ce premier chapitre d'introduction, le deuxième chapitre sera consacré à l'état de l'art en Deep Learning pour les images histopathologiques. Nous présenterons les méthodes de détection et de segmentation employées et introduirons formellement les réseaux génératifs adversaires (GANs). Nous reviendrons également plus en détail sur les notions d'apprentissage et d'augmentation de données. Le troisième chapitre traitera des déformations spatiales aléatoires appliquées aux glomérules et à leur impact sur les performances de segmentation. Le quatrième chapitre s'intéressera à la synthèse de texture par l'exemple telle qu'étudiée en informatique graphique. Nous tenterons de synthétiser des images de glomérules et d'incorporer ces images dans notre base d'entraînement. Dans le cinquième chapitre, nous procéderons de la même manière (synthèse de glomérule et entraînement d'un modèle de segmentation avec ces images), mais en nous intéressant aux méthodes par GANs.

Chapitre 2

État de l'art

Ce chapitre est consacré à une revue de l'état de l'art autour du traitement d'images histopathologiques. Nous nous intéresserons en particulier aux modèles de détection et de segmentation de glomérules, avant d'introduire plus en détail les notions liées à l'augmentation de données. Nous introduirons également les GANs.

2.1 Le *Deep Learning* pour les images histopathologiques

Nous présentons dans cette section quelques modèles parmi les plus utilisés pour faire de la détection et de la segmentation sur des images histopathologiques, ainsi que des exemples de traitements appliqués aux glomérules.

2.1.1 Modèles de détection

Les R-CNNs [GDDM14], pour *Regions with CNN features*, représentent la grande famille de méthodes pensées pour la détection d'objets sur des images. Les auteurs proposent un algorithme en deux étapes : d'abord la proposition de 2 000 régions d'intérêt, qui sont ensuite classifiées par un CNN. Les régions en question sont déterminées par un algorithme de recherche sélective (*Selective Search*) en 3 étapes :

- Sous-segmentation initiale,
- Combinaison de régions similaires en une seule plus grande via un algorithme glouton,
- Proposition des régions candidates.

Le CNN est pré-entraîné pour de la classification d'images générales, et produit un vecteur de caractéristiques. Ce vecteur est donné en entrée d'un algorithme de type *Support*

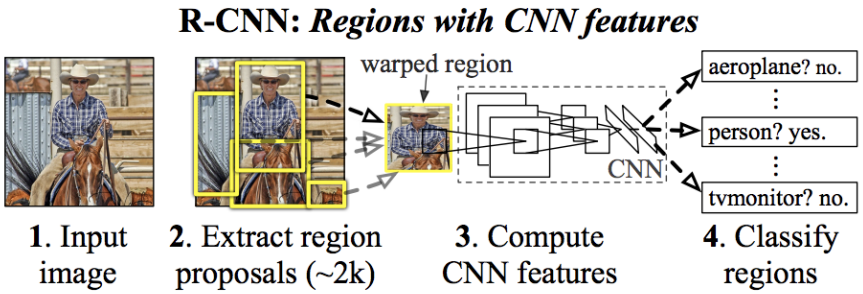


Figure 2.1 : Résumé de la méthode R-CNN [GDDM14].

Vector Machines (SVM), qui fournira à la fois une classification de la région et des valeurs de compensation qui permettent de placer les coins de la boîte englobante. La figure 2.1 résume la méthode.

Les R-CNNs ont le défaut d'être très lents : 47 secondes pour inférer sur une image, ce qui empêche toute utilisation en temps réel. Notons également que l'algorithme de recherche n'est pas appris, ce qui peut éventuellement poser problème pour les performances de la méthode. Ce sont ces défauts que cherchent à résoudre les évolutions des R-CNNs, appelées Fast R-CNN [Gir15] et Faster R-CNN [RHGS15].

En particulier, les auteurs de Faster R-CNN proposent un seul et unique réseau pour encapsuler la méthode complète (voir la figure 2.2). Ainsi, un CNN produit une carte de caractéristiques, qui sera donnée en entrée d'un module, un sous réseau de neurones dédié à une tâche spécifique, qui proposera les régions d'intérêt en lieu et place de l'algorithme de recherche sélective. Une couche dite de *Region of Interest pooling* permet de combiner les informations à disposition, c'est-à-dire la carte de caractéristiques et les régions d'intérêt, afin de les fournir à un algorithme qui effectuera la classification et fournira les coordonnées de la boîte englobante. Cette nouvelle méthode permet d'obtenir un temps d'inférence de seulement 0,2 secondes, ce qui permet désormais des applications en temps réel.

You Only Look Once est une autre famille de méthodes de détection. La troisième version en particulier, YOLOv3 [RF18], permet d'obtenir des résultats encore plus rapidement, avec des temps d'inférence allant de 20 à 50 millisecondes, tout en maintenant une qualité d'un niveau égal ou proche de l'état de l'art. La méthode parvient à de telles performances grâce à une architecture, présentée sur la figure 2.3, qui évalue l'image en entrée en une seule passe, sans avoir besoin de recourir à la sélection de régions d'intérêt. L'utilisation de connexions saute-couche (*skip connections*) héritées de ResNet [HZRS16] et de trois branches de prédiction héritées de *Feature-Pyramid Network* [LDG+17], permettent à YOLOv3 d'étudier les images à différentes échelles, et donc de repérer des objets quelle que soit leur taille relative dans l'image. Enfin, une fonction de coût pertinente permet à la méthode d'obtenir d'excellents résultats de détection. Cette fonction de coût est composée

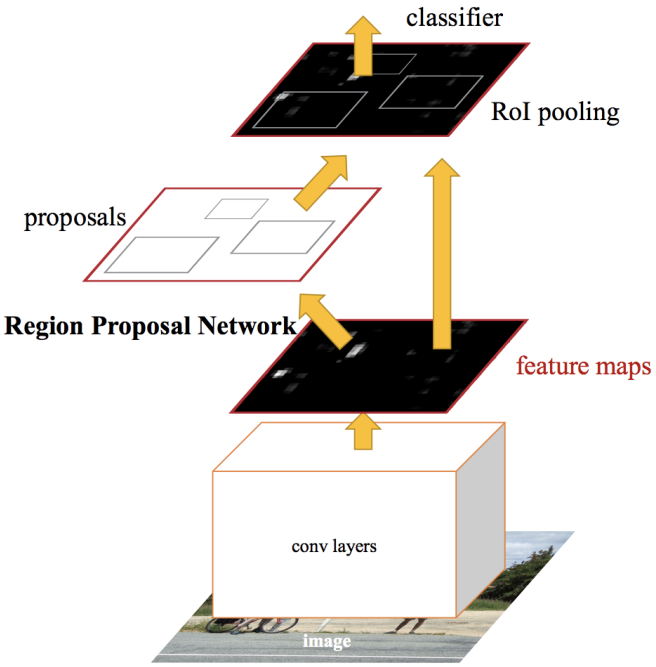


Figure 2.2 : Résumé de la méthode Faster R-CNN [RHGS15].

de différents termes qui permettent de corriger notamment les positions et les classifications des meilleures prédictions, en associant des poids plus importants aux meilleurs boîtes englobantes.

Les travaux récents cherchant à détecter les glomérules utilisent Faster RCNN ou YOLOv3 [TOFS+17; GPL+18; LWHC19; KML+19]. C'est le cas notamment de Heckenauer *et al.* [HWW+20] qui visent la détection en temps réel de glomérules. Pour ce faire, les auteurs utilisent la méthode YOLOv3, pré-entraînée sur une base de données d'images générales, dont ils affinent ensuite l'apprentissage sur les images de glomérules (*fine tuning*). Un de leurs objectifs est de reproduire la manière dont les pathologistes étudient les coupes complètes : par patches de la taille d'un écran standard d'ordinateur, avec différents niveaux de détail, d'où une approche multi-échelles.

2.1.2 Modèles de segmentation

Les modèles de segmentation partagent une architecture commune : un encodeur, qui extrait et condense spatialement de l'information depuis les images d'entrée, un décodeur, qui traite cette information et la remet à la taille de l'entrée par sur-échantillonnage, et enfin un classifieur qui utilise les informations décodées pour produire une carte de segmentation. Taghanakai *et al.* [ATAC+21] proposent une revue des méthodes utilisées pour les images naturelles et les images médicales.

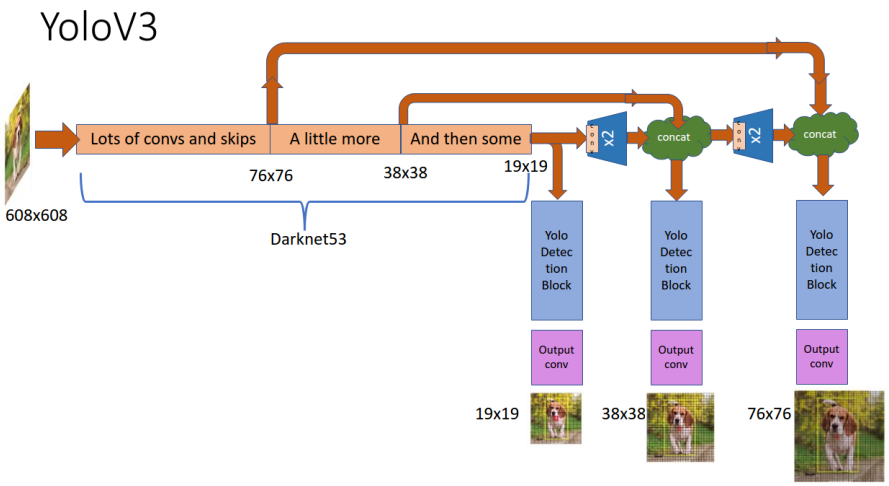


Figure 2.3 : Architecture de YOLOv3 (source : Towards Data Science [RF18]).

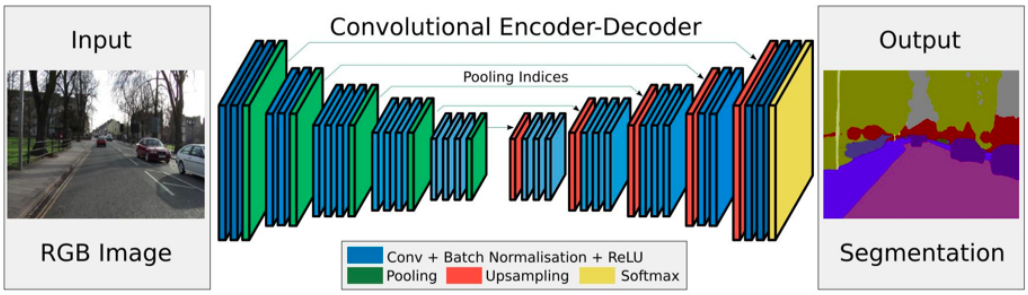


Figure 2.4 : Architecture de SegNet [BKC17].

Dans le cas de SegNet [BKC17], présenté à la figure 2.4, l'encodeur est un réseau de classification pré-entraîné sur des images générales, VGG-16 [SZ15], ce qui permet de profiter de la puissance de représentation de ce réseau tout en réduisant significativement le nombre de paramètres à apprendre. En parallèle, les paramètres des couches de *max-pooling*, qui réduisent la taille des images en conservant la valeur maximale dans une certaine fenêtre d'analyse, en général de taille 2×2, sont stockés en mémoire afin d'être réutilisés par le décodeur lors des étapes de sur-échantillonnage.

Le modèle le plus utilisé en imagerie médicale est le U-Net [RFB15], qui tire son nom de la manière dont le présentent les auteurs : en forme de U, tel que sur la figure 2.5. L'encodeur et le décodeur sont entraînés ensemble, et des connexions saute-couche (*skip connections*) permettent de transférer de l'information d'une partie à l'autre en concaténant les cartes de caractéristiques (*features maps*). Cette architecture est conçue pour pouvoir être entraînée même avec des jeux de données de petite taille, ce qui est souvent le cas en imagerie histopathologique.

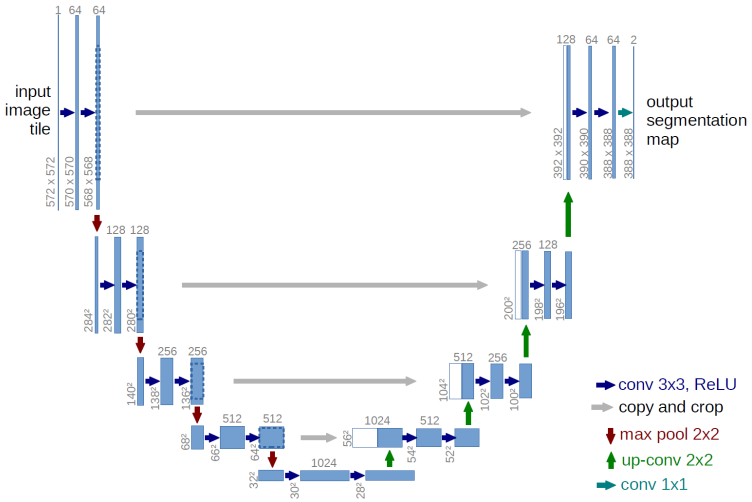


Figure 2.5 : Architecture de U-Net.

Une évolution de U-Net, U-Net++ [ZRSTL18; ZSTL20], cherche à améliorer les performances de segmentation du modèle orignal en remplaçant les *skip connections* par un maillage complet de couches de convolution, comme montré sur la figure 2.6. L'objectif de ces nouvelles connexions est de concaténer des *features maps* portant une information sémantique cohérente au niveau du décodeur, ce qui n'était pas forcément le cas dans la précédente version. Une supervision profonde intermédiaire permet de calculer la fonction de coût finale à partir d'un maillage plus ou moins complet, ce qui permet de créer une balance dans les performances recherchées entre rapidité et qualité de la segmentation.

Les auteurs de Attention U-Net [OSF+18] s'intéressent également aux *skip connections*, en les agrémentant d'une porte d'attention (*Attention Gate*) supplémentaire. En effet, les *skip connections* ont tendance à transporter de l'information redondante, et les *Attention Gates* permettent de donner un poids plus important aux régions réellement intéressantes des *features maps*. Comme montré sur la figure 2.7, la porte d'attention prend en entrée la couche à concaténer et la couche précédente du réseau, de plus petite taille et donc avec des caractéristiques possédant une meilleure force de représentation. Les deux couches sont ramenées à la même taille et sommées pour aligner les poids, et une fonction sigmoïde permet d'obtenir des sorties entre 0 et 1. Les régions proches de 1 sont les plus intéressantes.

Le *Recurrent Residual Convolutional* UNet (ou R2UNet) [AYH+19] tire parti de méthodes issues de la littérature autour des réseaux de neurones, les Recurrent CNN [LH15] et les Residual CNN [HZRS16]. Les réseaux résiduels permettent une meilleure stabilité de l'entraînement des réseaux profonds en créant des raccourcis dans le modèle, et les CNNs récurrents permettent d'obtenir des caractéristiques avec un meilleur pouvoir de représentation en appliquant plusieurs couches de convolution à une carte de caractéristiques et en sommant les sorties. Il existe de nombreuses autres variantes de UNet, pouvant cumuler les approches précédentes telles que l'Attention Residual UNet [LLY+21], Attention UNet++ [LTC+20] ou tentant d'appliquer d'autres techniques connues en Deep

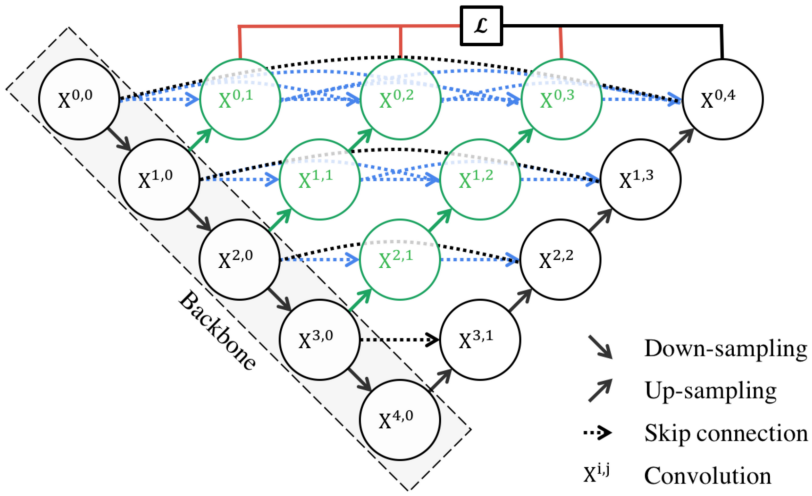


Figure 2.6 : Architecture de U-Net++ [ZRSTL18; ZSTL20].

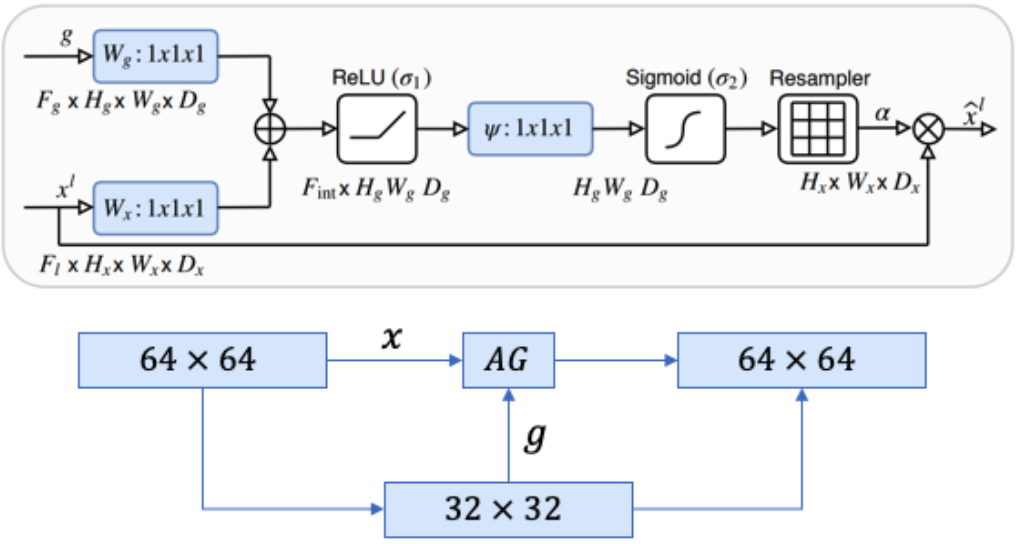
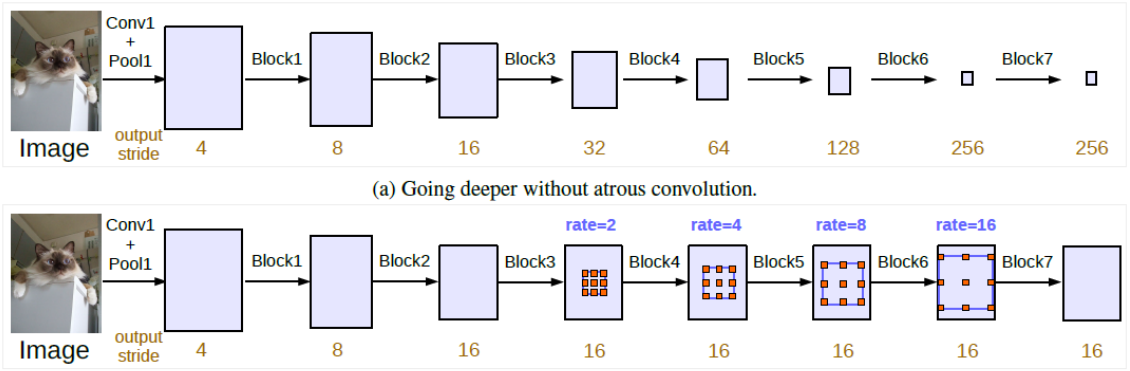


Figure 2.7 : Attention Gate de AttUNet. [OSF+18] Première ligne : représentation schématique de la porte d'attention. Deuxième ligne : schéma des entrées/sorties.



(a) Going deeper without atrous convolution.

(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Figure 2.8 : Illustration des convolutions « à trou » utilisées par DeepLabv3 [CPSA17]. Elles permettent de ne pas réduire la résolution des cartes de caractéristiques tout en gardant un bon champ récepteur pour les couches profondes.

Learning telles que Inception UNet [PA20], Attention UNet avec des *skip connections* dilatées [HGZ21] ou encore *Self and Cross Attention* UNet [PTR+21]. Nous ne reviendrons pas sur chaque modèle et ne pourrons tester toutes les variantes dans le manuscrit, aussi nous nous concentrerons sur la version originale, déjà très performante.

La famille de méthodes DeepLab, dont DeepLabv3 est la troisième itération [CPSA17], propose une architecture différente grâce à des convolutions dites « à trou » (*atrous* dans le texte). Les pixels composant un filtre de convolution sont espacés d'une distance r variable, ce qui permet d'augmenter le champ récepteur des couches (*field of view*) sans avoir à recourir à des couches de *pooling*, qui réduisent la résolution des cartes de caractéristiques tout au long de la progression dans le réseau. La carte de segmentation est alors directement obtenue en sortie du réseau encodeur, sans devoir passer par une phase de décodage. La figure 2.8 illustre l'opération. En complément, les auteurs utilisent une pyramide de convolutions à trous appliquées en parallèle avec différents écarts entre les éléments des filtres, afin d'extraire des caractéristiques à différent niveaux d'échelle. C'est la phase de *Atrous Spatial Pyramid Pooling* (ASPP). Les caractéristiques résultantes sont concaténées et convoluées de nouveau pour revenir à la forme des caractéristiques initiales.

Le modèle U-Net est l'architecture la plus utilisée en imagerie biomédicale [LKB+17], et en particulier pour la segmentation de glomérules sur des patches de WSIs [dBHS+18; MLS+21; DLX+21]. Les variantes telles que U-Net++ ont apporté de légères améliorations [LWHC19], mais seulement sur quelques exemples. Altini *et al.* [ACB+20], en plus de proposer un logiciel d'aide au diagnostic assisté par ordinateur, cherchent à segmenter et classer les coupes de tissus en 3 régions : fond, glomérule et glomérule sclérosé. Pour ce faire, les auteurs utilisent et comparent SegNet et DeepLabv3. Ces réseaux sont entraînés sur des patches issus de coupes complètes. Durant l'inférence, les segmentations sont projetées sur les coupes complètes, puis des opérateurs de morphologie mathématique et un algorithme des K-moyennes sont appliqués en post-traitement. Merveille *et al.* [MLS+21] ont pour objectif la mise en place d'une chaîne de travail pour l'extraction d'information de

WSI consécutives d'un volume 3D et traitées avec différents colorants. La première étape consiste à effectuer un recalage non-rigide entre deux coupes consécutives. La seconde consiste à segmenter les glomérules de chaque coupe à l'aide d'un U-Net, puis à rechercher des appareillements entre glomérules d'une coupe à l'autre, dans les deux sens. Seuls les appareillements valides dans les deux directions sont conservés, et enfin des caractéristiques sont extraites des patches résultants, dans les deux versions. D'autres travaux utilisent un Seg-Net [BFCGLD20] ou un U-Net avec un encoder ResNet [SMG+21], pré-entraînés sur le jeu de données ImageNet [DDS+09]. Les deux obtiennent de meilleurs résultats qu'avec un U-Net, Salvi *et al.* [SMG+21] ajoutant une étape de post-traitement basée sur la détection de noyaux de cellules pour raffiner la segmentation. Il existe également des travaux utilisant des versions modifiées de U-Net pour le traitement de WSI, qui doivent alors gérer des images d'entrée en très haute résolution [HdBdB+19; GDK+19; ACB+20; KFN+21].

Pour l'ensemble de la thèse, nous considérons la tâche de segmentation des glomérules, c'est-à-dire la séparation entre l'objet glomérule du tissu (ou fond), dans des patches extraits de WSIs. Cela revient à résoudre un problème à deux classes pour chaque pixel de chaque patch. Nous supposons que chaque patch contient un ou plusieurs glomérules, complets ou partiels, entourés de tissu, ou contient seulement du tissu. Nous utiliserons pour toutes les expérimentations qui seront conduites dans les chapitres suivants la version originale de U-Net [RFB15], avec une seule couche de convolution par étage, six étages et 64 filtres pour la première couche de convolution. Les filtres sont de taille 4×4. Nous utilisons des convolutions transposées pour le sur-échantillonnage. L'architecture est résumée au tableau 2.1. Nous utilisons enfin pour fonction de coût une entropie croisée (*cross-entropy loss*). Nous avons implémenté U-Net en python3 à l'aide de Tensorflow 2 [AAB+15]. Tous les entraînements ont été réalisés sur les serveurs du centre de calculs (*High Performance Computing Center*) ROMEO, hébergé par l'Université de Reims Champagne-Ardenne.

2.2 Introduction aux GANs

2.2.1 Généralités

Les réseaux génératifs adversaires (*Generative Adversarial Networks*, GANs) [GPAM+14] ont été introduits en 2014. Leur but est d'estimer la distribution d'un jeu de données et de produire de nouvelles données issues de cette distribution. Un GAN est composé de deux réseaux de neurones distincts : un réseau générateur G, qui doit estimer la distribution des données d'entraînement, et un réseau discriminateur D, qui doit estimer la probabilité qu'un échantillon reçu en entrée provienne des données réelles ou soit une donnée générée par G. En pratique, G reçoit en entrée un vecteur de bruit issu d'une certaine distribution et produit une donnée, et D reçoit en entrée une donnée, réelle ou générée, et donne en sortie une valeur de probabilité. Les deux réseaux sont entraînés conjointement et de manière compétitive : l'objectif de G est de tromper D, et D doit différencier données réelles et données générées.

couche	nombre de filtres	taille des cartes de caractéristiques
conv1	64	$128 \times 128 \times 64$
conv2	128	$64 \times 64 \times 128$
conv3	256	$32 \times 32 \times 256$
conv4	512	$16 \times 16 \times 512$
conv5	512	$8 \times 8 \times 512$
conv6	512	$4 \times 4 \times 512$
up-conv1	512	$8 \times 8 \times 512$
up-conv2	512	$16 \times 16 \times 512$
up-conv3	256	$32 \times 32 \times 256$
up-conv4	128	$64 \times 64 \times 128$
up-conv5	64	$128 \times 128 \times 64$
up-conv6	1	$256 \times 256 \times 1$ (carte de segmentation)

Table 2.1 : Architecture de notre modèle U-Net. Les couches de convolution (conv1 à conv6) ont un pas de 2, réduisant donc la taille des cartes de caractéristiques par un facteur 2 en largeur et en hauteur. Les couches de sur-échantillonnage (up-conv1 à up-conv6) augmentent la taille des cartes de caractéristiques par un facteur 2, et reçoivent en entrée la concaténation de la carte précédente et de la carte de la première branche du réseau de même taille. Par exemple, up-conv3 reçoit la concaténation des sorties de up-conv2 et conv4.

Notons p_{data} la distribution des données réelles. Cette distribution est inconnue et doit être estimée par le générateur. Nous notons p_g la distribution apprise par le générateur. Nous définissons également la distribution p_z du bruit dont sont issues les entrées du générateur. Nous modélisons le générateur comme une fonction différentiable, qui à un vecteur de bruit $\mathbf{z}$ associe une donnée $G(\mathbf{z})$. Nous notons enfin $D(\mathbf{x})$ la probabilité qu'une donnée $\mathbf{x}$ provienne des données réelles (et donc de p_{data}).

Les deux réseaux jouent à un jeu à somme nulle de type min-max modélisé par la formule suivante :

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.1}$$

Les auteurs ont démontré que le jeu possède un optimum global tel que $p_g = p_{data}$. En théorie, G peut donc apprendre n'importe quelle distribution. De plus, pour un certain G fixé, le discriminateur D optimal est :

$$D^*_G(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \tag{2.2}$$

Ainsi, si les deux réseaux ont convergé vers l'optimum global, D n'est plus capable de différencier les données réelles des données générées, et nous aurons toujours $D(\mathbf{x}) \simeq \frac{1}{2}$.

Si les GANs peuvent en théorie apprendre n'importe quelle distribution, ils ont le défaut de ne pas rendre explicite $p_g(\mathbf{x})$.

2.2.2 GANs convolutifs

Radford *et al.* [RMC16] ont étendu le principe des GANs aux réseaux de neurones convolutifs avec les DCGANs (*Deep Convolutionnal Generative Adversarial Networks*). Ces derniers sont capables de générer des images telles que des chambres ou des visages. Des modifications par rapport à l'architecture classique des CNNs permettent d'améliorer leur stabilité :

- Remplacement des couches de *pooling* par des convolutions à pas supérieur à 1 (*strided convolutions*)
- Utilisation de la normalisation par lots (*batchnorm)*
- Abandon des couches entièrement connectées (*fully connected layers*)
- Utilisation de la fonction d'activation *tanh* pour la dernière couche du générateur
- Utilisation des LeakyReLU [1] pour toutes les couches du discriminateur

Afin d'évaluer le pouvoir de représentation des DCGANs, les auteurs ont entraîné un DCGAN sur la base de données ImageNet, puis ont entraîné un algorithme de classification supervisé de type SVM (*Support Vector Machines*) sur la base CIFAR-10, en utilisant les cartes d'activation du discriminateur comme extracteur de caractéristiques. Les résultats obtenus (précision de 82.8%), proches de méthodes similaires de type k-moyennes (80.6%), montrent la robustesse et la généralité des caractéristiques apprises, alors même que le discriminateur n'a jamais vu d'image provenant de la base ImageNet.

2.2.3 GANs conditionnels

Les GANs conditionnels [MO14] permettent, comme leur nom l'indique, de générer des données selon des conditions, qui sont passées en entrée du générateur, par exemple sous la forme d'un entier codant une classe c. Le générateur et le discriminateur reçoivent tout deux cette condition en entrée. La fonction de coût s'écrit facilement dans ce nouveau cas [MO14] :

$$L_{cGAN}(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}|\mathbf{c}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c})))] \tag{2.3}$$

Nous étudierons plus en détails une forme particulière de GANs conditionnels au chapitre 5.

[1] reLU : $f(x) = max(0,x)$, Leaky reLU : $f(x) = max(0,x)$ si $x > 0$, $f(x) = 0.01x$ sinon.

2.3 Augmentation de données

2.3.1 Le sur-apprentissage

Lors de l'entraînement d'un modèle de *Deep Learning* supervisé, les données d'entraînement sont séparées en deux sous-ensembles, les ensembles d'entraînement et de validation. Le modèle apprend sur les images du jeu d'entraînement : pour chaque image, l'erreur entre la réponse du réseau et la vérité terrain permet de calculer la valeur d'une fonction de coût à optimiser. Les poids du réseau sont ensuite mis à jour par rétro-propagation à partir des gradients de la fonction de coût. Comme pour tout modèle d'apprentissage automatique, le réseau peut faire coller ses résultats au jeu d'entraînement, il n'est alors plus capable d'appliquer les connaissances apprises à de nouvelles données : le modèle est en sur-apprentissage (*overfitting*). La figure 2.9 illustre ce phénomène. La courbe verte est une bonne modélisation des données d'entraînement : l'erreur entre le modèle et les données est non nulle mais l'erreur sur des données de test sera faible. La courbe rouge montre un sur-apprentissage : l'erreur entre le modèle et les données est nulle, mais la capacité de généralisation est mauvaise puisque l'erreur entre le modèle et des données de test sera élevée. Le jeu de validation sert à se prémunir contre le sur-entraînement : après chaque époque (*epoch*), c'est-à-dire chaque cycle d'entraînement, le coût associé aux images de validation est calculé, sans mise à jour du réseau. Ce coût de validation diminue durant l'apprentissage, conjointement au coût d'entraînement, jusqu'à un certain point, comme illustré par la figure 2.10. Lorsque le coût de validation augmente, le modèle est en sur-apprentissage. Plus les fonctions de coût auront diminué avant le début du sur-apprentissage, plus les performances du réseau seront bonnes. Il est donc souhaitable que le réseau puisse apprendre pendant le plus d'époques possible, en repoussant le moment où commence le sur-apprentissage. Pour ce faire, le moyen le plus simple est d'augmenter la taille du jeu de données d'entraînement. Cela peut se faire par la collecte de nouvelles données ou, comme évoqué dans l'introduction, de manière artificielle en appliquant des techniques d'augmentation de données.

2.3.2 Les augmentations en histopathologie numérique

Nous avons vu au chapitre d'introduction qu'il existe de nombreuses méthodes d'augmentation de données, pouvant être utilisées conjointement. Le tableau 2.2 présente une vue d'ensemble des grandes familles de méthodes utilisées dans les articles traitant d'images histopathologiques rénales. Les rotations et symétries, simples à implémenter et très efficaces, sont systématiquement utilisées. Parmi les augmentations les plus utilisées, nous trouvons les déformations élastiques et les transformations colorimétriques.

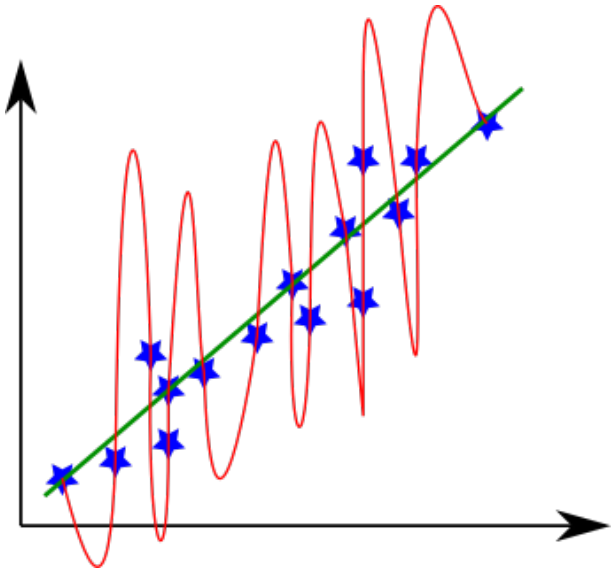


Figure 2.9 : Illustration du sur-apprentissage. La courbe verte montre un apprentissage correct, la courbe rouge montre un sur-apprentissage.

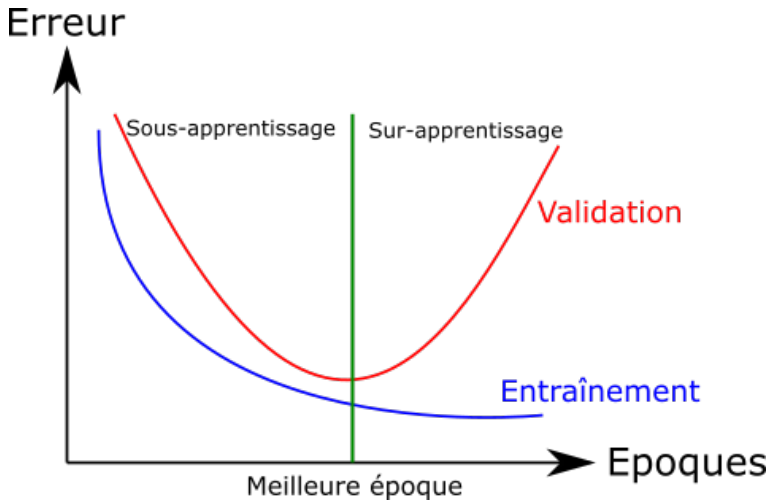


Figure 2.10 : Évolution des erreurs d'entraînement et de validation. Lorsque l'erreur de validation diminue, le modèle est en sous-apprentissage. Lorsqu'elle augmente, le modèle est en sur-apprentissage.

Augmentations	[dBHS+18]	[MLS+21]	[DLX+21]	[SMG+21]	[HdBdB+19]	[ACB+20]	[KFN+21]	[GPL+18]	[KJC16]	[HWW+20]	[FvdLL21]
Rotations	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Symétries	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Décalage/Recadrage		✓	✓						✓		✓
Mise à l'échelle		✓	✓	✓	✓	✓			✓		
Cisaillement									✓		✓
Déformations élastiques	✓	✓	✓		✓	✓	✓				✓
Floutage	✓	✓	✓		✓	✓					✓
Ajout de bruit		✓			✓	✓					
Coupes			✓								
Transformations colorimétriques	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

Table 2.2 : Augmentations utilisées dans une sélection d'articles traitant d'images histopathologiques rénales.

2.3.3 Les GANs en histopathologie numérique

Les GANs ont été utilisés avec succès en histopathologie numérique pour du transfert de coloration [VFWL21b] ou pour la génération de tissus [TOG20]. De précédents travaux s'intéressent à la synthèse de patches de tissus pour améliorer l'entraînement de classifieurs dans le cadre du diagnostic de cancers [WSV+20; LPF+20; QMSY20], ou l'amélioration de la segmentation de noyaux de cellules [HAS+17; MBC+20]. Hou *et al.* [HAS+17] mettent en place une chaîne de traitement résumée à la figure 2.11. Les images sont considérées comme étant sur deux plans, le premier plan étant composé des noyaux et l'arrière plan du fond. À partir d'une sur-segmentation, les noyaux sont effacés de l'image de départ grâce à une méthode d'*inpainting* [Tel04], consistant à compléter une texture manquante dans une zone d'une image à partir de la texture environnante. Ensuite, à partir d'une sous-segmentation et d'une autre image réelle dans une autre coloration, les auteurs créent de la texture pouvant correspondra à des noyaux. Cette texture est placée sur la nouvelle image de fond à l'aide d'un masque de segmentation généré grâce à des polygones déformés. Enfin, un GAN vient raffiner l'image ainsi créée, à partir notamment des caractéristiques de texture d'une image de référence.

Senaras *et al.* [SNS+18] s'intéressent au cancer du sein. Les auteurs utilisent une forme particulière de GAN conditionnel, admettant en entrée un masque de segmentation [IZZE17]. Il s'agit d'une méthode de translation d'image-à-image (*image-to-image translation*), que nous étudierons en détail au chapitre 5. Mahmood *et al.* [MBC+20] utilisent un CycleGAN [ZPIE17], méthode entraînant conjointement deux modèles de translation d'image-à-image à l'aide d'une fonction de cohérence entre ces deux modèles. Cette méthode permet d'avoir un modèle qui transforme les images du domaine A en images du domaine B et un autre procédant dans l'autre sens, comme le montre la figure 2.12. Dans le cas de l'article, les auteurs peuvent ainsi synthétiser soit de nouvelles cartes de segmentation, soit de nouvelles images texturées.

Vasiljevic *et al.* [VFWL21a] s'intéressent au transfert de coloration à l'aide d'un CycleGAN également. Les auteurs ont remarqué que ce type de transformation induit un bruit caché sur les images, et que ce bruit est porteur d'information relative aux colorations considérées. Ainsi, en perturbant ce bruit, il est possible d'augmenter la variabilité des images sans toucher à leur contenu. Partant de ce fait, les auteurs mettent au point une

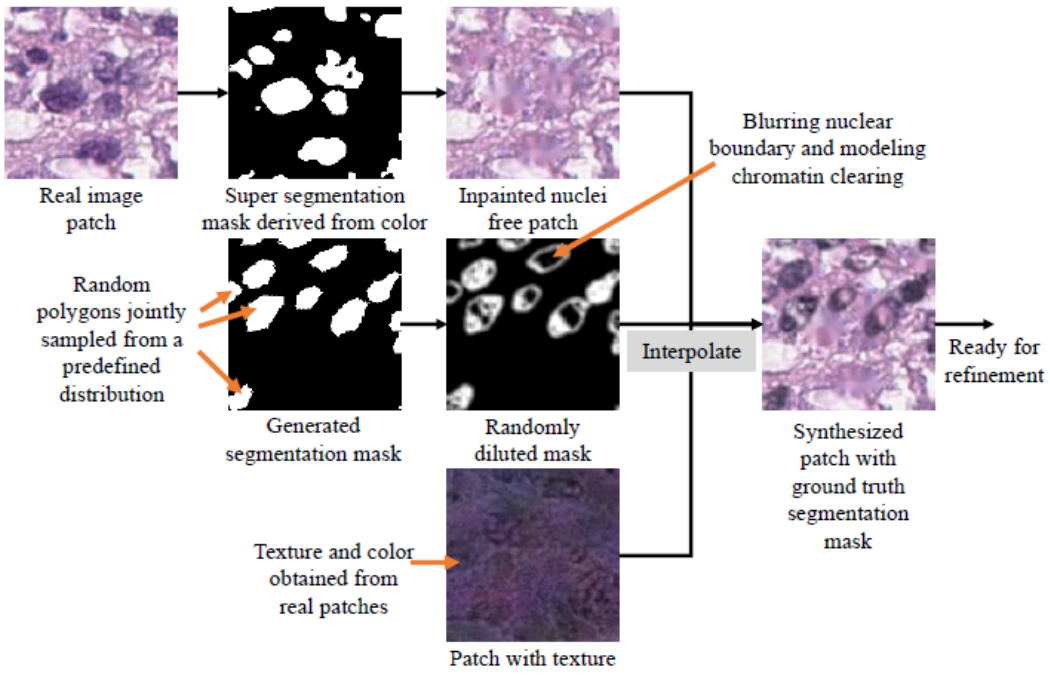


Figure 2.11 : Méthode de synthèse d'images histopathologiques par *inpainting*. Un GAN permet de raffiner les images générées [HAS+17].

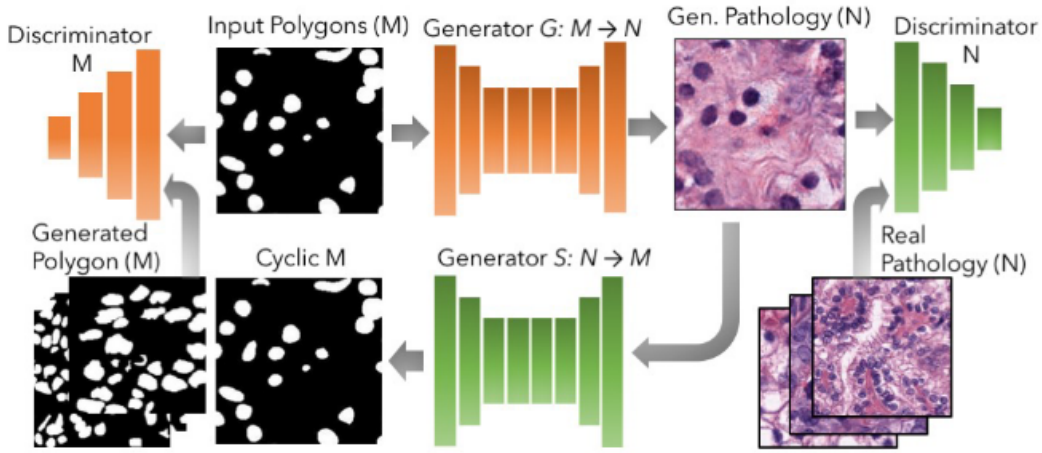


Figure 2.12 : Méthode de synthèse d'images histopathologiques [MBC+20]. Cette méthode utilise des CycleGAN [ZPIE17] pour générer des cartes de segmentations ou des images texturées de manière cohérente.

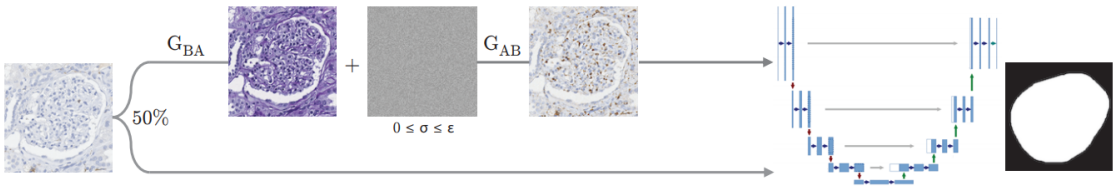


Figure 2.13 : Résumé de la méthode d'augmentation par transfert de coloration [VFWL21a].

nouvelle technique d'augmentation de données. En résumé, ils entraînent un CycleGAN à transformer une image d'une coloration A à une coloration B et inversement, selon le principe de la méthode. Pour procéder à l'augmentation, ils utilisent la méthode pour transformer l'image A vers l'image B. L'image B est perturbée par un bruit gaussien (*adversarial attack*), puis l'image perturbée est de nouveau transformée en l'image A. La procédure est résumée à la figure 2.13.

Murali *et al.* [MLG+20] explorent l'utilisation de *Deep Convolutional* GANs [RMC16] combinés à des *Enhanced Super-Resolution* GANs (ESRGAN) [WYW+18] pour générer des images de tissus rénaux de haute résolution (1024×1024), qui seront évaluées visuellement par des experts. Cependant, aucune évaluation n'est faite dans le cadre de l'augmentation de données pour l'entraînement d'un modèle de segmentation.

Dans tous les cas, la synthèse d'images soulève la question de la chaîne de traitement, c'est-à-dire les modèles à utiliser, ainsi que de la taille des jeux de données disponibles, les GANs en particulier nécessitant beaucoup de données pour produire de bons résultats [KAH+20; SZG+21; TJL+21].

2.3.4 Le méta-apprentissage pour optimiser les augmentations

Des travaux récents visent à trouver les politiques d'augmentation optimales pour une tâche donnée, en utilisant des techniques de méta-apprentissage [CZM+19; LKK+19; CZSL20b] ou de l'apprentissage adversaire [ZWZZ19]. Il s'agit en général de sélectionner des modèles de transformation et d'optimiser leurs paramètres : fréquences des transformations, ordre dans lequel elles sont appliquées, amplitude des déformations, etc. Nous reviendrons sur ces méthodes au chapitre 3.

2.3.5 Quelles perspectives pour notre jeu de données ?

Nous disposons de 10 coupes complètes capturées à plusieurs niveaux de résolution d'échantillonnage. Nous utilisons le niveau correspondant à une résolution de 0.506 µm/pixel. Les dix coupes sont réparties entre les jeux d'entraînement (4 coupes), de validation (2 coupes) et de test (4 coupes). Depuis ces coupes et grâce à la vérité terrain, nous extrayons des

patches dont la taille varie selon les chapitres, ne contenant que du tissu, ou contenant un glomérule centré (et potentiellement des morceaux de un à deux autres glomérules). Nous obtenons la répartition initiale suivante pour les images avec glomérules :

- entraînement : 660 patches
- validation : 589 patches
- test : 1091 patches

Cette répartition sera amenée à changer selon les chapitres du manuscrit.

Les bénéfices apportés par les transformations de coloration [TLB+19; LMS+19; VFWL21a; LDX+21; FvdLL21] et les transformations affines [FvdLL21] sont bien quantifiés dans la littérature. L'impact de l'utilisation de différentes augmentations sur des WSIs d'organes de rat est évalué dans [KFN+21], y compris les transformations affines et les déformations élastiques, mais seulement avec une approche binaire (avec/sans). Nous nous intéresserons donc dans cette thèse à l'évaluation des méthodes d'augmentation qui font défaut dans la littérature autour de la segmentation de glomérules. Nous nous concentrerons ainsi sur trois méthodes, une par transformation et deux par génération. En particulier, nous étudierons l'impact sur l'apprentissage des augmentations par déformations aléatoires et par synthèse d'images, impact qui n'est pas évalué dans la littérature.

Un de nos objectifs est donc de synthétiser des glomérules pour de l'augmentation de données. Comme vu précédemment, les méthodes pour ce faire sont nombreuses, mais toutes ne sont pas adaptées à nos images. Nous devrons faire face à différentes contraintes : la nécessité d'asservir la méthode de synthèse à une vérité terrain ainsi que le haut niveau de détail présent dans nos images. Notons que la plupart des articles cités précédemment proposent une méthode en deux temps : d'abord la génération d'une vérité terrain, puis la synthèse de l'image à partir de cette vérité terrain. Logiquement, nous reprendrons cette chaîne de traitement, représentée à la figure 2.14, pour la suite de nos travaux. Nous nous concentrerons sur la deuxième partie de la chaîne. Par ailleurs, dans tout le manuscrit, nous évaluerons en première approche la qualité des images générées par un examen visuel, qui nous permettra d'indiquer le niveau de réalisme d'une synthèse. Ainsi, une image générée sera jugée réaliste si la synthèse a permis de reproduire fidèlement à la fois les structures et les détails de texture à tous les niveaux d'échelle. Une image floue ou présentant d'importantes distorsions sera jugée moins réaliste. Cet examen n'est ici pas effectué par des experts, qui pourraient trouver des détails non réalistes même dans des images qui sont pour nous de bonne qualité.

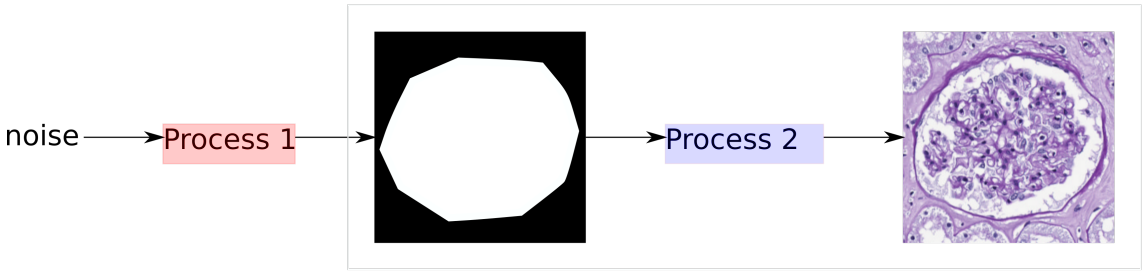


Figure 2.14 : La chaîne de traitement que nous souhaitons mettre en place. Le premier processus consiste à générer une vérité terrain, le second à générer une image de glomérule à partir de cette vérité terrain.

Chapitre 3

Déformations spatiales aléatoires

Comme nous l'avons vu au chapitre 2, la robustesse des méthodes de segmentation face aux variations de coloration ont été étudiées [LMS+19; LWHC19; VFWL21a]. Nous nous concentrons ici sur les variations de formes, qui n'ont pas été pleinement étudiées dans la littérature. Des méthodes de déformation sont bien utilisées dans le papier originel de U-Net [RFB15] ou dans des travaux récents sur les glomérules [MLS+21], mais l'impact sur l'entraînement des réseaux de neurones n'est pas quantifié. Nous nous proposons dans ce chapitre d'évaluer l'impact de l'augmentation des données d'entraînement pour la segmentation, avec différentes méthodes de déformations spatiales aléatoires.

Nous considérons une sélection de modèles de déformation, comprenant :

- des Champs de Déplacement Aléatoires (*Random Displacement Fields*, RDF) ;
- des modèles de déformation basés sur des points de contrôle (*Grid Based Deformations*, GBD) ;
- des déformations basées sur du mouvement brownien implémenté avec du bruit de Perlin (*Fractional Brownian Motion*, FBM) ;
- des déformations basées sur des moindres carrés mobiles (*Moving Least Squares as-rigid-as-possible deformations*, MLS-arap) ;
- des déformations basées sur l'intégration de champs de vitesse continus et affines par morceaux (*Continuous Piecewise-Affine Based*, CPAB).

Ces modèles proviennent de différents milieux : certains sont plus utilisés au sein de la communauté du traitement d'images médicales, d'autres au sein de la communauté d'informatique graphique. Par ailleurs, nous laissons de côté le recalage tel qu'envisagé par exemple par Napela *et al.* [NMP+19], nos images se prêtant peu à ce type de méthode, et les déformations aléatoires étant de toute manière plus générales. Les travaux présentés dans ce chapitre ont été publiés dans le journal *Computer Methods and Programs in Biomedicine* [AAWD22b].

3.1 Contexte et notations

Nous définissons une déformation spatiale aléatoire comme un champ de vecteur résultant du filtrage d'un bruit blanc, c'est-à-dire la réalisation d'un processus stochastique ayant une distribution gaussienne. Une grande variété de modèles a été proposée dans la littérature, avec différentes manières de générer le champ de vecteurs. Une déformation spatiale en 2D est une fonction continue $\mathcal{D} : \mathbb{R}^2 \to \mathbb{R}^2$, potentiellement non bijective, qui associe aux coordonnées d'entrée $\mathbf{x} = [x,y]$ les coordonnées transformées $[\mathcal{D}_x(\mathbf{x}), \mathcal{D}_y(\mathbf{x})]$. Appliquer des déformations à une image implique de résoudre un problème d'échantillonnage : les coordonnées transformées peuvent ne pas correspondre à des emplacements de pixels, et peuvent ne pas couvrir l'ensemble du domaine de l'image. C'est pourquoi les transformations $\mathcal{D}$ sont en général définies de manière inversée (*reverse warping*) et calculées à partir de la grille de pixels de l'image de sortie vers le domaine spatial de l'image d'entrée. Pour la suite, nous considérons une image carrée de taille N^2. La grille de pixels de l'image de sortie (l'image déformée) I sera notée $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{N^2}$, où les $\mathbf{p}_i = [x_i, y_i]$ sont les coordonnées des pixels. La grille de pixels de l'image d'entrée associée I' sera notée $\mathbf{P'} = \{\mathbf{p}'_i\}_{i=1}^{N^2}$. Nous pouvons ainsi écrire :

$$\mathcal{D}(\mathbf{P}) = \mathbf{P} + \overrightarrow{\mathbf{D}} \tag{3.1}$$

où $\overrightarrow{\mathbf{D}} = \{\vec{\mathbf{d}}_i\}_{i=1}^{N^2}$ est un ensemble de vecteurs de déplacement, ou un champ de déplacement (*flow field*). Suivant cette approche, les valeurs des pixels de I sont obtenues en interpolant les valeurs des pixels de I', en considérant les pixels les plus proches des coordonnées déformées $\mathcal{D}(\mathbf{P})$ dans le domaine spatial, c'est-à-dire :

$$I = \mathcal{S}(\mathcal{D}(\mathbf{P}), I') = \mathcal{S}(\mathbf{P} + \overrightarrow{\mathbf{D}}, I') \tag{3.2}$$

où $\mathcal{S}$ est une fonction d'interpolation continue. Des choix classiques pour $\mathcal{S}$ sont les interpolations bilinéaires, bicubique ou par spline. Nous considérons que ce choix n'est pas crucial pour notre étude et utilisons une interpolation bilinéaire ou bicubique selon la méthode et son implémentation. Nous notons également que certains emplacements de $\mathcal{D}(\mathbf{P})$ peuvent se trouver en dehors des frontières de la grille de pixels d'entrée. Une solution couramment utilisée est d'étendre l'image d'entrée par *padding*. Nous utilisons un padding symétrique le long des bords de l'image d'entrée.

3.2 Caractérisation des déformations aléatoires

3.2.1 Propriétés statistiques

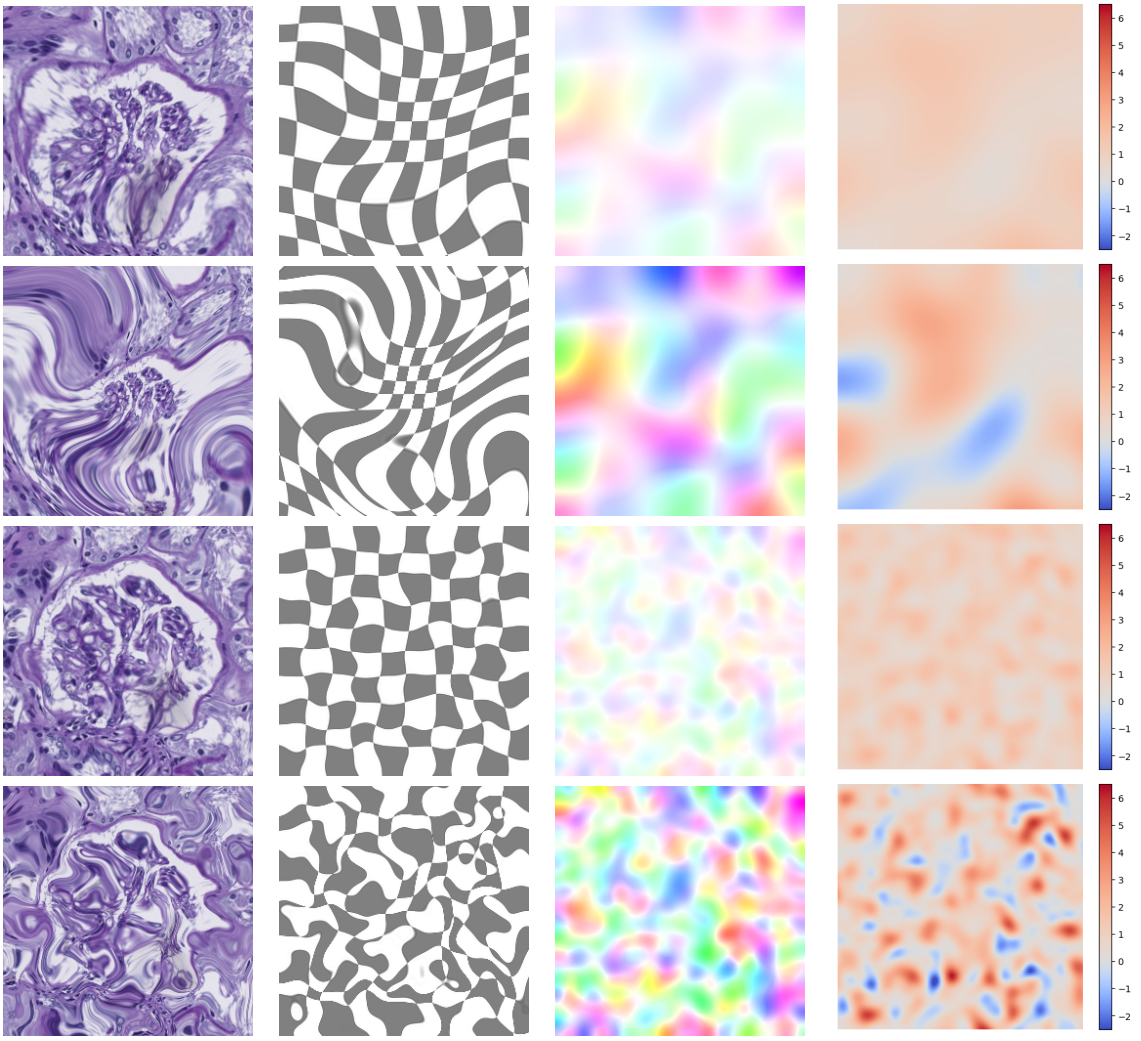
Chacun des modèles de déformation considérés possède ses propres paramètres. Pour un modèle donné et un ensemble de valeurs de paramètres, une déformation $\mathcal{D}$ est aussi définie par un ensemble de valeurs aléatoires, telles que $\mathcal{D}$ soit le résultat de la réalisation d'un processus stochastique avec une distribution gaussienne ou uniforme, qui peut être

assimilée à du bruit blanc. Les modèles de déformation étudiés ici diffèrent dans la manière dont les bruits sont générés et contrôlés en termes d'amplitude et de fréquence. Le champ de déplacement résultant $\overrightarrow{\mathbf{D}}$ pour une déformation donnée est un champ aléatoire gaussien multi-varié, où les vecteurs de déplacement aux emplacements d'échantillonnage sont corrélés spatialement. Ces corrélations spatiales peuvent provenir par exemple d'un filtrage, de la distance aux points de contrôle ou de la somme pondérée de bruits blancs filtrés passe-bas, selon la méthode. Chaque modèle a sa propre manière de contrôler ces corrélations spatiales. La quantité de corrélation entre les vecteurs de déplacement à des emplacements voisins influence les fréquences présentes dans le champ de déplacement : une haute corrélation implique de basses fréquences et inversement. L'amplitude des déformations est contrôlée au moyen d'un facteur d'échelle sur la taille des vecteurs de déplacement. La figure 3.1 montre des exemples de déformations avec des basses fréquences (première et deuxième ligne), des hautes fréquences (troisième et quatrième ligne), ainsi qu'avec des amplitudes basses (première et troisième ligne) et des amplitudes hautes (deuxième et quatrième ligne).

3.2.2 Propriétés géométriques

Les déformations aléatoires peuvent préserver les formes et les textures contenues dans une image jusqu'à un certain point. Si la déformation est « trop forte », les images résultats semblent « distordues », « moins réalistes ». Alors que les travaux précédents cherchent à augmenter leurs données avec des images déformées visuellement réalistes, notre objectif est de comparer les modèles de déformation et de déterminer le montant optimal de distorsions pour l'entraînement d'un modèle de segmentation. Nous essayons dans cette section de quantifier ces notions et de caractériser les déformations aléatoires d'un point de vue géométrique.

La capacité des déformations aléatoires à préserver les formes et textures contenues dans une image est directement en lien avec la préservation des distances et des angles dans une image. Un cas particulier de distorsion est le repli (*foldover*), qui a lieu quand les orientations entre des paires d'emplacements de pixels sont inversées. Le repli peut être vu comme une limite haute de distorsion, puisque les incohérences induites dans la topologie peuvent provoquer de profonds changements dans les apparences des images déformées, aussi bien localement que globalement. La quantité de distorsion locale induite par un champ de déplacement peut être caractérisée par la matrice jacobienne de la fonction de déformation $\mathcal{D}$ à chaque point $\mathbf{p}_i \in \mathbf{P}$, que nous notons $\nabla\mathcal{D}(\mathbf{p}_i)$. Le déterminant de la matrice jacobienne, c'est-à-dire $\mathrm{J}(\mathbf{p}_i) = \det(\nabla\mathcal{D}(\mathbf{p}_i))$, encode le facteur d'échelle local, qui indique si les distances sont raccourcies ($|\mathrm{J}(\mathbf{p})| \in [0,1)$) ou étendues ($|\mathrm{J}(\mathbf{p})| > 1$), et son signe indique si les déformations produisent du repli ($\mathrm{J}(\mathbf{p}) \leq 0$) ou non ($\mathrm{J}(\mathbf{p}) > 0$). Les distances sont préservées quand $|\mathrm{J}(\mathbf{p})| = 1$. Ainsi, moins les vecteurs de déplacement pour des pixels voisins sont corrélés, plus le repli est probable localement. La figure 3.1 montre l'affichage de J pour deux exemples de déformations obtenues avec le modèle *Random Displacement Fields*. Le cas de la seconde ligne montre du repli : J est négative pour environ



(a) (b) (c) (d)

Figure 3.1 : Quatre exemples de déformations appliquées à des patchs contenant un unique glomérule, avec l'affichage des valeurs correspondantes du déterminant de la matrice jacobienne J. Première ligne : basses fréquences et basse amplitude. Deuxième ligne : basses fréquences et amplitude élevée. Troisième ligne : hautes fréquences et basse amplitude. Quatrième ligne : hautes fréquences et amplitude élevée. Les déformations des deuxièmes et quatrièmes lignes sont importantes, provoquant du repli, qui se traduit par des valeurs négatives de J. (a) Image déformée. (b) Même déformation appliquée à un échiquier. (c) Champ de déplacement codé en HSV. (d) Affichage de J.

20% des positions du champ de déplacement. Une autre caractéristique des champs de déplacement impliquée dans la préservation des formes et des textures est la mise à l'échelle locale des angles. Une déformation préservant les angles est dite conforme, si sa matrice jacobienne est une matrice de rotation multipliée par un scalaire. Pour savoir si une déformation est proche d'une déformation conforme, il faut calculer le produit scalaire des

composantes normalisées de la matrice jacobienne, c'est-à-dire $\mathrm{S}(\mathbf{p}) = \frac{\mathcal{D}_x(\mathbf{p})}{\|\mathcal{D}_x(\mathbf{p})\|} \cdot \frac{\mathcal{D}_y(\mathbf{p})}{\|\mathcal{D}_y(\mathbf{p})\|}$. La valeur de $\mathrm{S}(\mathbf{p})$ est comprise entre -1 (agrandissement maximal de l'angle) et 1 (rétrécissement maximal de l'angle). Plus la valeur est proche de 0, plus les angles sont préservés. Les difféomorphismes forment une catégorie spécifique de fonctions de déformation. Ils sont différentiables, inversibles et leur inverse est également différentiable. Le champ de déplacement produit par un difféomorphisme ne contient aucun repli et est conforme. Parmi notre sélection de modèles de déformation, seules les déformations basées sur l'intégration de champs de vitesse CPA [FHBF15; FHBF17] appartiennent à cette catégorie.

Pour chacun des modèles de déformation sélectionnés et pour différentes valeurs de paramètre, nous examinons la distribution des métriques précédentes, calculées sur 1 000 champs de déplacement générés aléatoirement, de la manière suivante :

- Taux de repli, calculé comme : $|\{\mathbf{p}_i \in \mathbf{P} \,\mathrm{tq}\, \mathrm{J}(\mathbf{p}_i) \leq 0\}_{i=1}^{N^2}|/N^2$;
- Préservation des distances, calculée comme l'histogramme de : $\{\mathrm{J}(\mathbf{p}_i),\ \mathbf{p}_i \in \mathbf{P}\}$;
- Préservation des angles, calculée comme l'histogramme de : $\{\mathrm{S}(\mathbf{p}_i), \mathbf{p}_i \in \mathbf{P}\}$.

La figure 3.2 montre le tracé de ces distributions pour le modèle des *Random Displacement Fields*. Le tracé pour les autres méthodes est disponible en annexe.

3.3 Les modèles de déformation

3.3.1 *Random Displacement Fields*

Les *Random Displacement Fields* (RDF) [SSP03] ont été proposés par Simard *et al.*. pour simuler des déformations lisses. Tout d'abord, un champ de déplacement $\overrightarrow{\mathbf{V}} = \{\vec{\mathbf{v}}_i\}_{i=1}^{N^2}$ est généré, où $\mathbf{v}_i = [v_x, v_y]$ est tel que $v_x = \mathrm{rand}(-1, +1)$ et $v_y = \mathrm{rand}(-1, +1)$. La fonction $\mathrm{rand}(-1, +1)$ renvoie des nombres pseudo-aléatoires uniformément distribués et compris entre -1 et $+1$. Ensuite, $\overrightarrow{\mathbf{V}}$ est convolué avec un filtre Gaussien $\mathcal{G}_\sigma$ d'écart-type σ, jouant le rôle de « coefficient d'élasticité » : de petites valeurs donnent un champ complètement aléatoire tandis que de grande valeurs produisent des déformations rigides. Le champ est finalement multiplié par un facteur d'échelle α qui contrôle l'amplitude des déformations. Ainsi, le champ de déplacement s'écrit :

$$\overrightarrow{\mathbf{D}} = \alpha\, \mathcal{G}_\sigma(\overrightarrow{\mathbf{V}}) \tag{3.3}$$

Le seul travail relatif à la segmentation de glomérules qui mentionne l'usage des RDFs pour l'augmentation de données est celui de Merveille *et al.* [MLS+21], où $\sigma = 10$ et $\alpha = 100$, pour des patchs de taille $N = 508$. Ce choix n'est pas discuté par les auteurs. La résolution d'échantillonnage par rapport à la taille des tissus étant la même que dans nos expériences (0,506 µm/pixel), il est possible de déformer nos patchs de taille $N = 256$ avec les mêmes

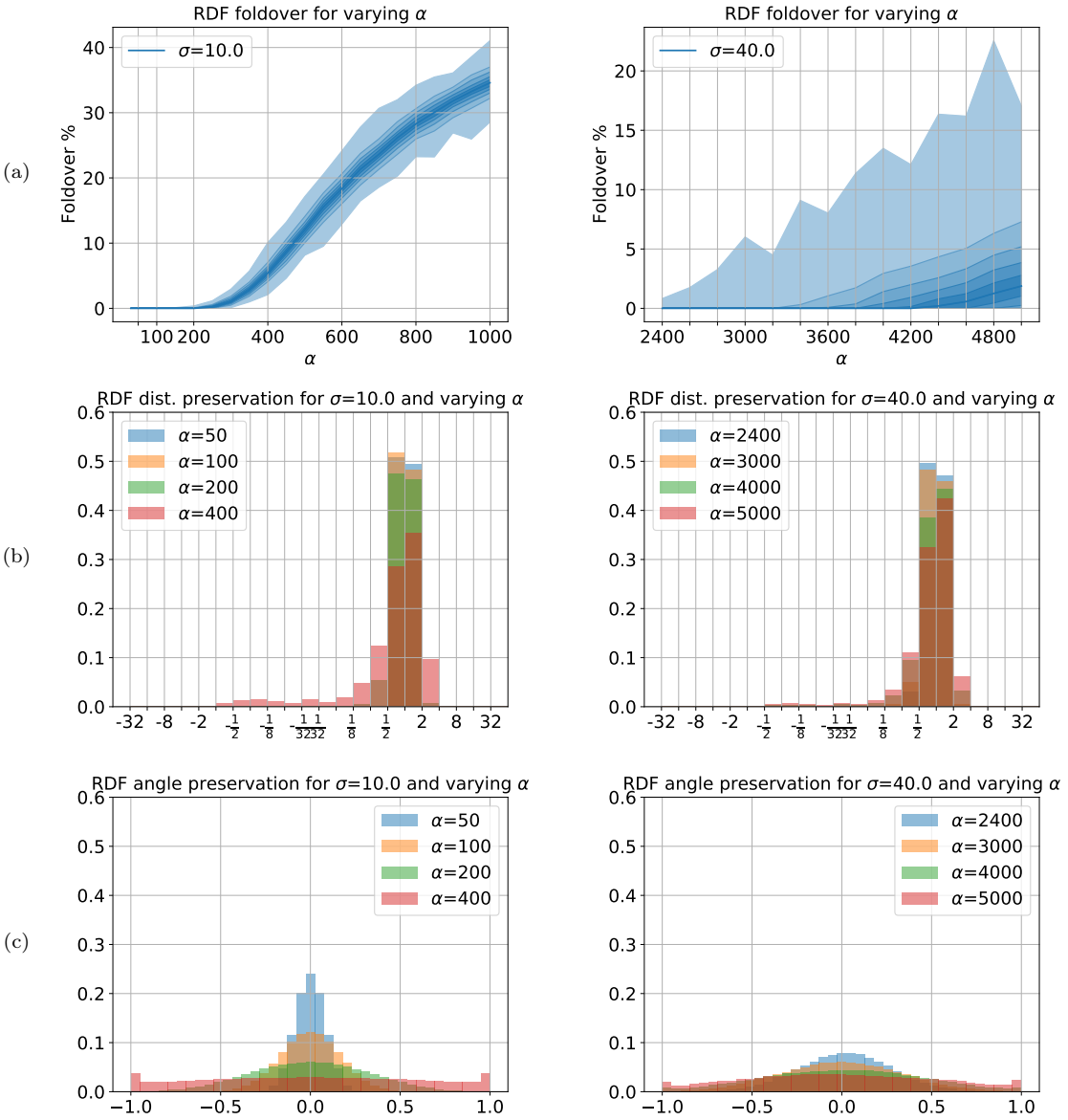


Figure 3.2 : Caractérisation des champs de déplacement générés avec la méthode des *Random Displacement Fields*, basée sur 1 000 champs générés aléatoirement, pour $\sigma = 10$ (première colonne) et $\sigma = 40$ (deuxième colonne) et des valeurs croissantes de α. (a) Taux de repli (la courbe centrale représente la médiane de la distribution, tandis que les autres courbes correspondent aux percentiles). (b) Préservation des distances (échelle $\log_2$ symétrique). (c) Préservation des angles.

paramètres pour le même impact visuel. Ces valeurs de paramètres produisent des résultats qui sont visuellement proches des images originales, comme illustré par la figure 3.3.

La courbe de la figure 3.2(a) illustre l'émergence du repli pour le cas $\sigma = 10$, qui arrive

lorsque α dépasse 150 (voir la section précédente pour l'interprétation). Le taux de repli reste bas (moins de 1%) pour des valeurs telles que $150 < \alpha < 200$, et touche seulement 10% des champs de déformation. Les distances restent bien préservées, tandis que le facteur d'échelle pour les angles varie approximativement entre $-0{,}5$ et 0,5. Ces éléments posent la question du bénéfice que pourraient apporter des valeurs plus grandes d'α par rapport à σ pour l'augmentation de données. La figure 3.3 montre des résultats de déformation pour $\sigma = 10$ et $\sigma = 40$, avec différents α. Cette figure, avec la figure 3.2, seconde colonne, montre que des déformations avec peu de distorsions peuvent être produites avec de grandes valeurs de σ et des valeurs appropriées de α, comme par exemple $\alpha = 2400$ pour $\sigma = 40$. Cela peut-être expliqué par l'effet du filtre Gaussien qui, pour des valeurs élevées de σ, limite les fréquences du champ de déformation, qui requiert alors des valeurs d'autant plus grandes d'α pour produire des distorsions. Contrôler le niveau de distorsion est une tâche compliquée avec ce modèle, puisqu'il n'y a pas d'idée intuitive sur l'amplitude des déformations induites par α pour un σ donné. La figure 3.4 montre que pour une valeur donnée de σ, il existe une relation linéaire entre α et la distance de déplacement, calculée comme la norme des vecteurs de déplacement, avec une pente qui n'est en revanche pas proportionnelle à σ. La section suivante se concentre sur des déformations basées sur des points de contrôle, qui pallient cette lacune.

3.3.2 Déformations basées sur des points de contrôle

Les déformations basées sur des points de contrôle sont très courantes et de nombreuses méthodes d'interpolation ont été adaptées pour cette tâche. Nous nous donnons un ensemble de points de contrôle source $\mathbf{C}' = \{\mathbf{c}'_k\}_{k=1}^{K}$, les emplacements cibles $\mathbf{C} = \{\mathbf{c}_k\}_{k=1}^{K}$ sont calculés grâce à des déplacements aléatoires. Comme suggéré par les auteurs de U-Net [RFB15], nous considérons ici que les amplitudes locales des déplacements suivent une loi normale centrée avec un écart-type σ, c'est-à-dire :

$$\mathbf{C} = \mathbf{C}' + \sigma \overrightarrow{\mathbf{V}} \tag{3.4}$$

où $\overrightarrow{\mathbf{V}} = \{\vec{\mathbf{v}}_k\}_{k=1}^{K}$ est tel que $\mathbf{v}_k \sim \mathcal{N}(0{,}1)$.

Nous calculons le champ de déplacement $\overrightarrow{\mathbf{D}}$ en interpolant les vecteurs de déplacement entre les points de contrôle cible $\mathbf{C}$, c'est-à-dire :

$$\overrightarrow{\mathbf{D}} = \mathcal{T}(\mathbf{P}, -\overrightarrow{\mathbf{V}}) \tag{3.5}$$

où $\mathcal{T}$ est une fonction d'interpolation.

Si les relations topologiques entre les points de contrôle sont disponibles, par exemple sous la forme d'une grille régulière, il est possible de choisir des fonctions d'interpolation bilinéaires ou bicubiques, comme suggéré toujours par les auteurs de U-Net [RFB15]. Un plus haut degré de régularité peut être obtenu avec des splines ou une interpolation par déplacement de moindres carrés (*Moving Least Squares*, MLS), comme nous le verrons par la suite.

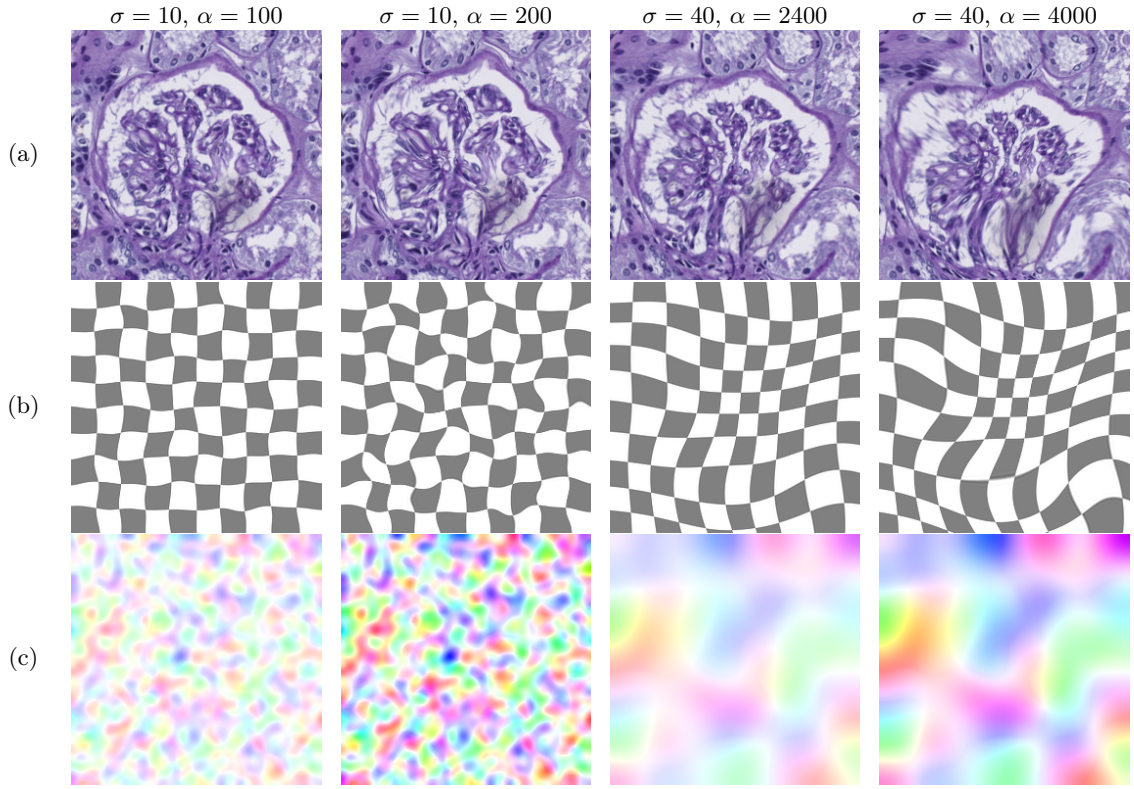


Figure 3.3 : Exemples de déformations RDF appliquées à des patchs contenant un glomérule, pour différentes valeurs de σ et α. Les directions des vecteurs de déplacement sont les mêmes dans les quatre exemples. (a) Patch déformé. (b) Déformation appliquée à un échiquier. (c) Champ de déplacement codé en HSV.

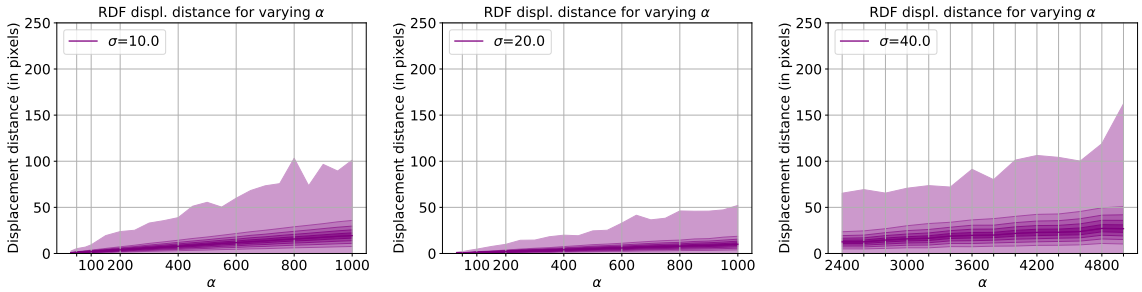


Figure 3.4 : Distribution des distances de déplacement pour les RDFs, pour différentes valeurs de paramètres. Pour chaque ensemble de paramètres, les distances de déplacement ont été échantillonnées à partir d'un ensemble aléatoire de 4 champs de déplacement de taille $N = 256$ (c'est-à-dire environ 262 000 mesures). La courbe centrale représente la médiane de la distribution, tandis que les autres courbes correspondent aux percentiles.

Avec cette approche, l'ensemble des points de contrôle source $\mathbf{C}'$ et le paramètre d'échelle σ sont contrôlés par l'utilisateur. Nous détaillerons plus tard les différentes options à notre disposition pour générer ces points de contrôle.

3.3.2.1 Interpolation par splines

Les splines polyharmoniques sont une famille de fonctions d'interpolation utilisée pour l'interpolation de données dispersées, ce qui nous permet de les appliquer à nos points de contrôles, structurés ou non. La fonction interpolante a la forme suivante :

$$f(\mathbf{p}) = \sum_{k=1}^{K} w_k \, \phi(\|\mathbf{p} - \mathbf{c}_k\|) + \mathbf{a}^\top \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{3.6}$$

où le premier terme est une somme pondérée de fonctions en base radiale (*Radial Basis Function*, RBF) polyharmoniques, avec pour centres $\mathbf{C}$, et dont le second terme est polynomial. Nous considérons ici les deux choix suivants pour ϕ :

- $\phi(r) = r^2 \ln(r)$, qui correspond à la *Thin Plate Spline* (TPS) [Boo89], qui a une longue histoire dans la communauté de l'imagerie médicale pour le recalage [WL19] ;
- $\phi(r) = r^3$

Les poids $\{w_k\}_{k=1}^K$ et les coefficients du polynôme $\mathbf{a}$ sont estimés de manière à ce que les valeurs de la fonction interpolante soient égales à celle de la fonction aux points de contrôle $\mathbf{C}$, ce qui est obtenu en résolvant un système linéaire.

3.3.2.2 Déformations *Moving Least Squares*

Les déformations *Moving Least Squares* (MLS) ont été introduites par Schaefer *et al.* [SMW06]. À partir de deux ensembles de points de contrôle (source et destination), les déformations MLS trouvent une fonction de transformation $\mathcal{D}$ satisfaisant trois conditions :

1. Interpolation : $\mathcal{D}(\mathbf{C}) = \mathbf{C}'$;
2. Régularité : $\mathcal{D}$ doit produire des déformations régulières ;
3. Identité : si $\mathbf{C}' = \mathbf{C}$, alors $\mathcal{D}$ doit être la fonction identité.

Pour tout emplacement de pixel $\mathbf{p}$ à transformer, $\mathcal{D}(\mathbf{p})$ est construit en utilisant des moindres carrés mobiles pour minimiser :

$$\sum_{k=1}^{K} w_k \|(\mathbf{c}_k - \bar{\mathbf{c}})\mathbf{M} - (\mathbf{c}'_k - \bar{\mathbf{c}}')\|^2, \tag{3.7}$$

où $\mathbf{M}$ est une matrice de transformation affine, et les poids w_k sont définis comme $w_k = \frac{1}{\|\mathbf{c}_k - \mathbf{p}\|^4}$. Les points $\bar{\mathbf{c}}$ et $\bar{\mathbf{c}}'$ sont respectivement les centroïdes pondérés des points de contrôle cible et source, c'est-à-dire $\bar{\mathbf{c}} = \frac{\sum_k w_k \mathbf{c}_k}{\sum_k w_k}$ et $\bar{\mathbf{c}}' = \frac{\sum_k w_k \mathbf{c}'_k}{\sum_k w_k}$.

Nous nous concentrons ici sur la version « aussi rigide que possible » (*as-rigid-as-possible*) des déformations MLS (MLS-ARAP), où la contrainte $\mathbf{M}^\top\mathbf{M} = \mathbf{I}$ est introduite, c'est-à-dire la matrice $\mathbf{M}$ ne contient pas de facteur d'échelle. La fonction de transformation correspondante est la suivante :

$$\mathcal{D}(\mathbf{p}) = \|\mathbf{p} - \bar{\mathbf{c}}\| \frac{\overrightarrow{\mathbf{D}}(\mathbf{p})}{\left\|\overrightarrow{\mathbf{D}}(\mathbf{p})\right\|} + \bar{\mathbf{c}}', \tag{3.8}$$

où $\overrightarrow{\mathbf{D}}(\mathbf{p}) = \sum_k (\mathbf{c}'_k - \bar{\mathbf{c}}')\mathbf{A}_k$, avec :

$$\mathbf{A}_k = w_k \begin{bmatrix} \mathbf{c}_k - \bar{\mathbf{c}} \\ (\mathbf{c}'_k - \bar{\mathbf{c}}')^\perp \end{bmatrix} \begin{bmatrix} \mathbf{p} - \bar{\mathbf{c}} \\ -(\mathbf{p} - \bar{\mathbf{c}})^\perp \end{bmatrix}^\top \tag{3.9}$$

où $\perp$ est un opérateur sur des vecteurs 2D tel que $[x, y]^\perp = [-y,x]$. L'objectif de la méthode MLS-ARAP est de produire des résultats réalistes en préservant autant que possible les formes et les textures de l'image d'entrée lorsque les points de contrôle sont déplacés. Pour ce faire, il faut placer les points de contrôle sur des parties de l'image considérées comme rigides, tandis que les autres parties sont considérées flexibles.

3.3.2.3 Points de contrôle sur une grille régulière

En calculant un pavage de l'image d'entrée, il est facile de générer des points de contrôle sur une grille régulière. Nous pouvons utiliser cette méthode pour déformer aléatoirement tout type d'image et de contenu. Nous réunissons ce type de méthode sous l'appellation « déformation basée sur une grille » (*Grid-Based Deformation*, GBD). L'espacement entre les points de la grille détermine la corrélation des vecteurs de déplacement entre des pixels voisins, fixant ainsi la fréquence maximale du champ de déplacement. La valeur du facteur d'échelle σ détermine quant à elle la quantité de distorsion. L'espacement des points de la grille est en général fixé en choisissant le nombre de points sur la grille pour une taille d'image donnée, que nous notons $n^2 = K$. Ainsi, la distance en pixels correspondante entre des points de contrôle voisins est $d = N/(n-1)$. Pour une même valeur de d, un grand facteur d'échelle σ produira des distorsions plus importantes qu'un petit facteur d'échelle. Le lecteur pourra trouver en annexe les courbes de taux de repli, de préservation des distances et de préservation des angles correspondantes. Pour une valeur donnée de σ, le déplacement maximum en pixels selon un axe obéit à la loi des trois σ : environ 68% des déplacements seront de moins d'un σ pixel, tandis que 28% seront compris entre σ et 2σ, et enfin 4% seront compris entre 2σ et 3σ. Cela permet à l'utilisateur un contrôle plus intuitif que les *Random Displacement Fields*. La figure 3.5 montre des exemples de déformations basées sur des points de contrôle sur une grille régulière avec une interpolation cubique (GBD à l'ordre 3, ou GBD-3). Parmi les valeurs de paramètre utilisées sont inclues celles proposées pour U-Net [RFB15], mises à l'échelle pour notre taille de patch, ce qui nous donne $n = 3$ et $\sigma = 5$. Les valeurs de paramètres proposées dans l'article donnent des distorsions de très faibles fréquences et amplitudes, ce qui nous amène à considérer de plus fortes déformations. Les autres exemples ont pour paramètres $n = 5$ et un σ variant de 5

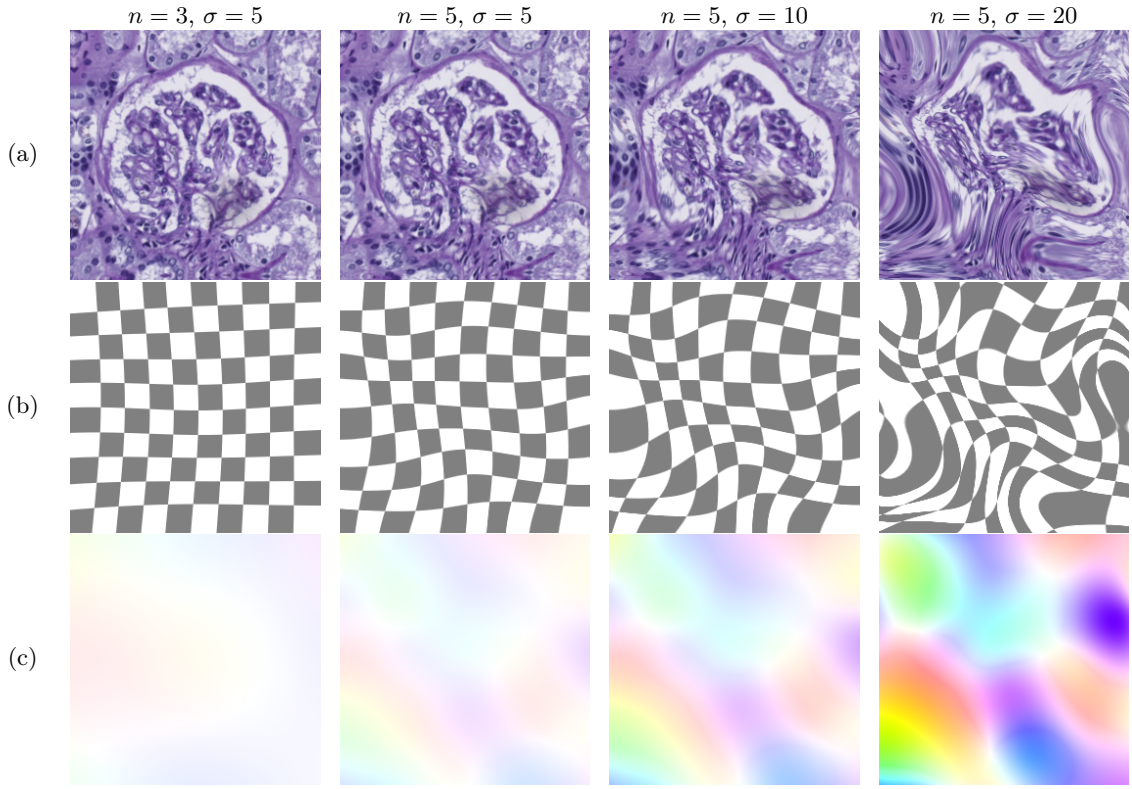
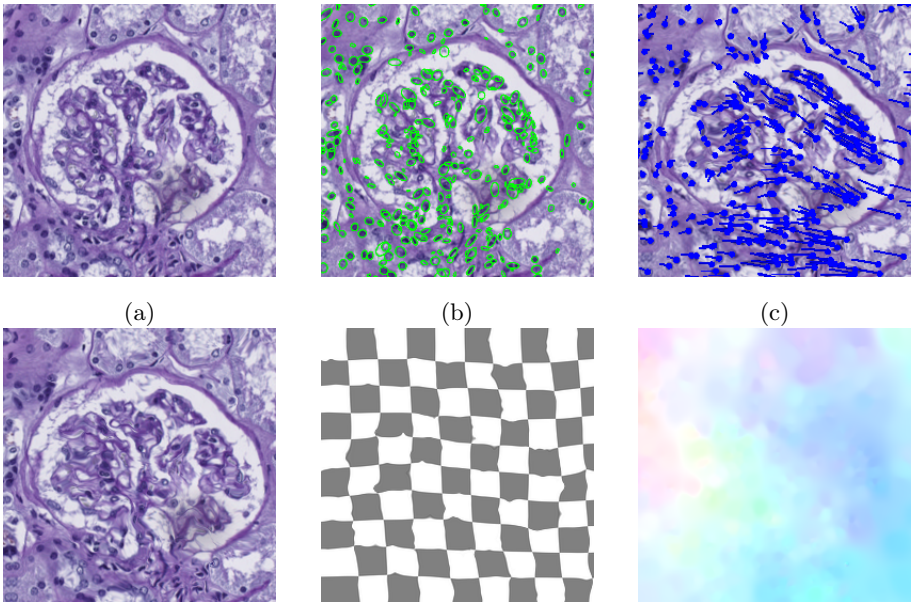


Figure 3.5 : Exemples de déformations basées sur une grille régulière, avec une interpolation à l'ordre 3, appliquées à des patchs contenant un unique glomérule, pour différentes valeurs de n et σ. Les directions des vecteurs de déplacement sont les mêmes pour les trois exemples où $n = 5$. (a) Patch déformé. (b) Déformation appliquée à un échiquier. (c) Champ de déplacement codé en HSV.

à 20, ce qui accentue l'effet des distorsions. Les figures A.2 et A.3 en annexe montrent le comportement de déformations *grid-based* aléatoires avec une interpolation TPS (ordre 2) et cubique (ordre 3) respectivement, avec différentes valeurs de n et σ, pour une image de taille $N = 256$. Quasiment aucune différence n'est observée entre les deux interpolations. La probabilité d'apparition du repli devient non nulle et augmente quand $\sigma \approx 10$ pour $n = 3$ ($d \approx 85$), et $\sigma > 5$ pour $n = 5$ ($d \approx 51$) ou $n = 10$ ($d \approx 26$). Une manière simple de drastiquement limiter le repli serait de considérer une borne supérieure de $d/2$ pour σ. Toutefois, cette approche exclut certaines déformations sans repli qui pourraient contribuer à l'augmentation de données. Par exemple, environ 30% des déformations avec $n = 3$ et $\sigma = 50$ sont sans repli (figures A.2 et A.3 en annexe). Dans notre étude de l'impact de l'augmentation de données sur les performances de segmentation des glomérules, nous examinerons les effets des déformations avec et sans repli.



(a) (b) (c)

(d) (e) (f)

Figure 3.6 : Exemple de déformations CNB-MVN-MLS appliquées à des patchs contenant un unique glomérule, pour $K = 310$ et $\sigma = 12$. (a) Patch d'entrée. (b) Noyaux de cellules détectés. (c) Déplacement des noyaux de cellule. (d) Image déformée. (e) Déformation appliquée à un échiquier. (f) Champ de déplacement codé en HSV.

3.3.2.4 Points de contrôle non structurés

Appliquer des déformations basées sur des points de contrôle non structurés nécessite de placer les points de contrôle source $\mathbf{C}'$ sur des emplacements appropriés de l'image d'entrée. Plutôt que de les positionner à des emplacements aléatoires, nous proposons de le faire en fonction du contenu des images (*content-aware*). Les noyaux de cellules observables sur les patchs de WSI étant presque invariants sur toutes les images, nous proposons de détecter et utiliser le centre de ces noyaux comme points de contrôle. Nous supposons que les noyaux sont des parties rigides de nos images, et nous cherchons donc à préserver leur forme grâce à une interpolation MLS-ARAP. À partir d'un patch à déformer, nous procédons en 3 étapes :

1. Détection des centres des noyaux de cellules, qui deviennent les points de contrôle source $\mathbf{C}'$ (une fois par patch) ;

2. Calcul des vecteurs de déplacement aux points de contrôle source pour obtenir les points de contrôle cible $\mathbf{C}$;

3. Interpolation des vecteurs de déplacement $\{\overrightarrow{\mathbf{c}'_i\mathbf{c}_i}\}_{i=1}^{K}$ entre les points de contrôle source et cible pour obtenir les emplacements des pixels en sortie $\mathbf{P}$.

Nous utilisons la méthode décrite par Mahmood *et al.* [MBC+20] pour la détection de noyaux de cellules dans des patchs de WSI, même si n'importe quelle méthode peut être utilisée. Il ne s'agit pas d'obtenir une détection parfaite mais seulement de détecter un certain nombre de points de contrôle. Les noyaux étant disposés irrégulièrement sur l'image, déplacer les points de contrôle de manière indépendante rend les résultats sujets au repli. À la place, nous obtenons les vecteurs de déplacement des points de contrôle en échantillonnant une distribution normale multivariée $\mathcal{N}(0,\Sigma)$, où Σ est une matrice de covariance de taille $K \times K$ prenant en compte les distances entre les centres des noyaux. Plus précisément, nous définissons Σ comme suit :

$$\Sigma_{i,j} = \sigma \exp\left(-\frac{\|\mathbf{c}'_i \mathbf{c}'_j\|}{\sqrt{2}N}\right) \tag{3.10}$$

pour $i,j \in [1,K]$, où σ est un facteur d'échelle contrôlant l'amplitude maximale des déformations. Les noyaux étant répartis sur toute l'image, nous normalisons les distances par la taille de la diagonale $\sqrt{2}N$. De fait, le seul paramètre utilisateur pour cette méthode est le coefficient σ. Nous nommons cette méthode CNB-MVN-MLS pour *Cell Nuclei Based, Multivariate Normal, MLS-ARAP*. Avoir un seul paramètre est avantageux par rapport aux autres méthodes considérées, aussi bien pour le réglage utilisateur que pour l'estimation des paramètres, que nous verrons dans la suite. La figure 3.6 montre un exemple de résultat obtenu avec cette méthode pour $K = 310$ (déterminé automatiquement par la méthode de détection des noyaux), et $\sigma = 12$. Dans nos patchs, le nombre de noyaux détectés varie entre 250 et 320. D'autres résultats sont présentés à la figure 3.7. On peut voir que les déformations obtenues sont plus riches en fréquences (hautes ou basses) que les déformations produites par RDF ou GBD. C'est une conséquence de la corrélation spatiale des points de contrôle et de la contrainte de rigidité. Les distorsions ont un comportement similaire à celui des autres modèles de déformation, comme montré à la figure A.5 en annexe.

3.3.3 Déformations basées sur des champs de vitesse

Les déformations CPAB (*Continuous Piecewise Affine based deformations*) [FHBF15; FHBF17] reposent sur l'intégration de champs de vitesse continus et affines par morceaux (CPA) définis comme des combinaisons linéaires dans une base orthonormée de champs de vitesse CPA. Ce modèle est difféomorphique ; les déformations CPAB sont donc différentiables, inversibles et leurs inverses sont également différentiables. De fait il est adapté aux tâches telles que le recalage non rigide ou à l'utilisation dans les *Spatial Transformer Networks* [DFH18]. Le terme « par morceau » fait référence au pavage du domaine de l'image d'entrée Ω en $4K$ cellules triangulaires. L'espace des transformations CPAB correspondant est le suivant :

$$\mathcal{D}^\theta(\mathbf{p}) = \phi^\theta(\mathbf{p},1) \tag{3.11}$$

où $\phi^\theta : \Omega \times \mathbb{R} \to \Omega$ est la solution de l'équation intégrale :

$$\phi^\theta(\mathbf{p},1) = \mathbf{p} + \int_0^1 v^\theta(\phi^\theta(\mathbf{p},\tau))\, d\tau \tag{3.12}$$

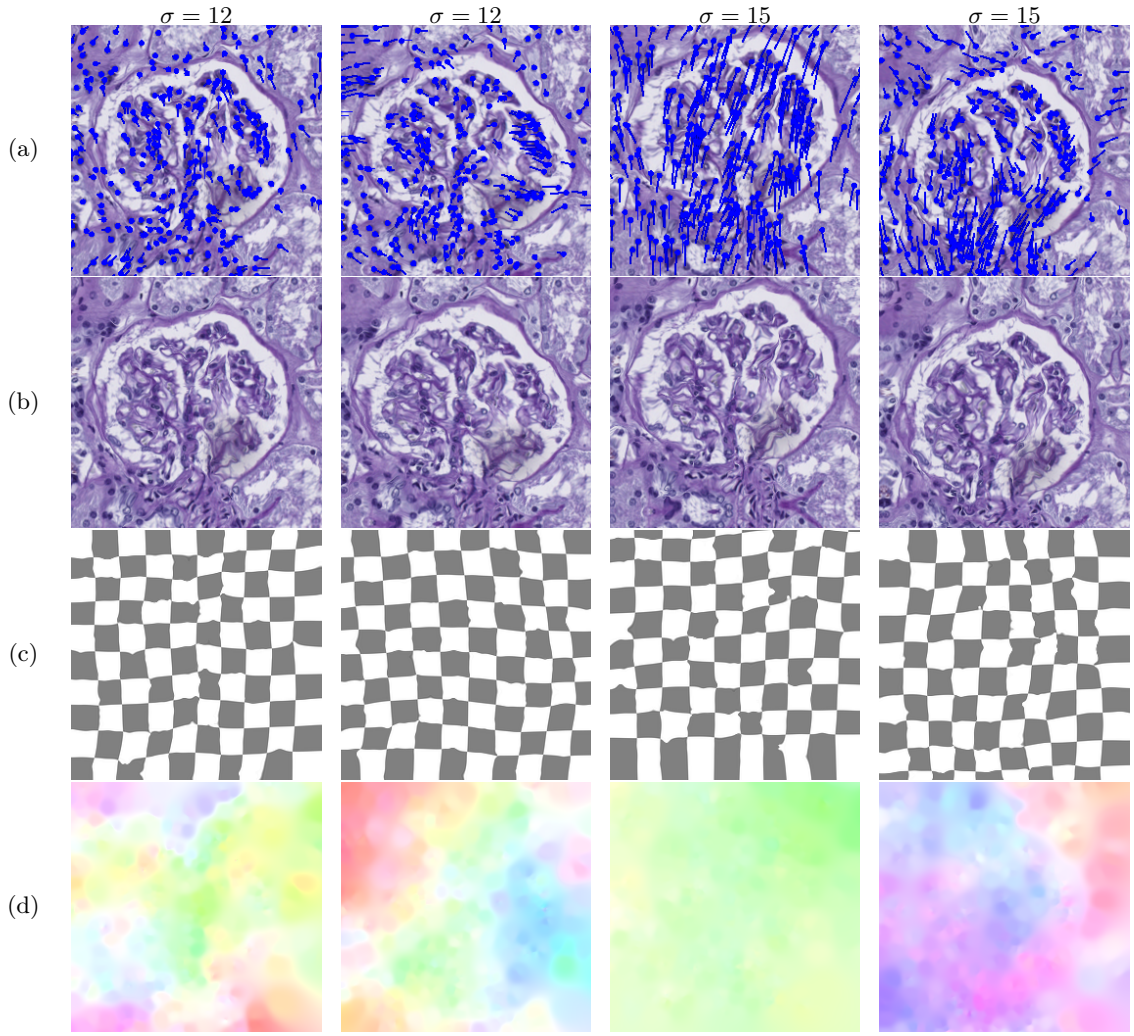


Figure 3.7 : Exemples additionnels de déformations CNB-MVN-MLS appliquées à des patchs contenant un unique glomérule, pour $K = 310$ et deux valeurs de σ. (a) Déplacements des noyaux. (b) Image déformée. (c) Déformation appliquée à un échiquier. (d) Champ de déplacement codé en HSV.

où v^θ appartient à l'espace linéaire des champs de vitesse CPA $\mathcal{V}$, et θ est le vecteur des poids dans la combinaison linéaire de champs CPA formant une base orthonormée de $\mathcal{V}$. Avec cette formulation il est possible d'imposer des contraintes de frontières et de préservation de volumes. Les auteurs proposent un solveur spécialisé qui calcule un compromis entre la solution exacte et une solution numérique approchée [FHBF15; FHBF17].

Pour des déformations CPAB aléatoires, le vecteur θ est choisi aléatoirement en échantillonnant une distribution normale multivariée $\mathcal{N}(0,\Sigma)$, où Σ est une matrice de covariance prenant en compte les distances entre les cellules afin d'assurer la régularité, et qui dépend d'un facteur d'échelle σ contrôlant l'amplitude des déformations. Les paramètres utilisateurs sont ainsi la taille du pavage K et le facteur d'échelle σ. Nous n'utilisons pas de

contraintes de frontière ni de préservation de volumes pour les déformations calculées dans toute la suite. Notons que le caractère difféomorphique de ce modèle garantit l'absence de repli.

La figure 3.8 montre des exemples de déformations obtenues avec le modèle CPAB, pour des pavages de taille $K = \{5 \times 5 = 25, 10 \times 10 = 100\}$, et $\sigma = \{1, 2\}$. Les déformations avec $K = 100$ sont plus régulières que les déformations avec $K = 25$, où le pavage sous-jacent est plus visible. Même dans les cas de grande amplitude, il n'y a aucun changement topologique.

La figure A.6 en annexe montre que les distances sont préservées, sauf pour le cas extrême $\sigma = 10$ qui induit probablement des problèmes numériques, ce qui est cohérent avec le fait que la méthode s'appuie sur des transformations affines. La distribution des changements d'échelle pour les angles est plus proche d'une distribution uniforme que celle des autres méthodes étudiées.

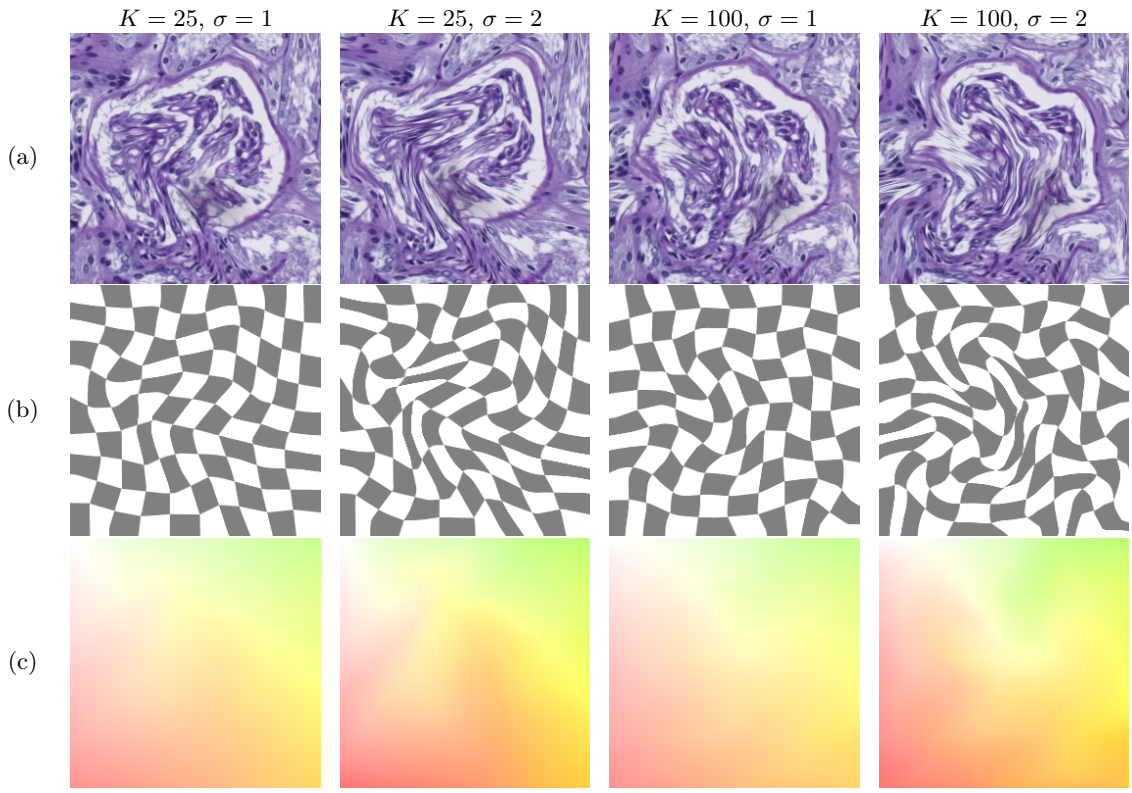


Figure 3.8 : Exemples de déformations CPAB appliquées à des patchs contenant un unique glomérule avec différentes valeurs de K et σ. Les directions des vecteurs de déplacement sont les mêmes dans les deux cas où $K = 25$, et dans les deux cas où $K = 100$. (a) Images déformées. (b) Déformation appliquée à un échiquier. (c) Champ de déplacement codé en HSV.

3.3.4 Déformations basées sur du mouvement brownien fractionnaire

3.3.4.1 Bruit de Perlin

La fonction de bruit de Perlin en 2D [Per85; EMP+02] est basée sur une grille régulière illimitée munie de vecteurs gradients unitaires fixés aux points de coordonnées entières. Les vecteurs gradient ont des directions aléatoires, échantillonnées aléatoirement dans le cercle unitaire, et sont utilisés pour calculer une fonction d'interpolation régulière entre les points de coordonnées entières. Notons $\vec{\mathbf{g}_A}$, $\vec{\mathbf{g}_B}$, $\vec{\mathbf{g}_C}$ et $\vec{\mathbf{g}_D}$ les vecteurs gradient aux coins d'une cellule de la grille. Pour tout point (x,y) à l'intérieur de la cellule ($x,y \in [0,1]$), la fonction de bruit est définie comme :

$$\begin{aligned}\eta(x,y) =& \vec{\mathbf{g}_A} \cdot (x,y)(1-u(x))(1-u(y)) + \\ & \vec{\mathbf{g}_B} \cdot (x,y-1)(1-u(x))u(y) + \\ & \vec{\mathbf{g}_C} \cdot (x-1,y)u(x)(1-u(y)) + \\ & \vec{\mathbf{g}_D} \cdot (x-1,y-1)u(x)u(y)\end{aligned}$$

où $u : [0,1] \to [0,1]$, définie par $u(t) = 3t^2 - 2t^3$, est une fonction d'interpolation régulière.

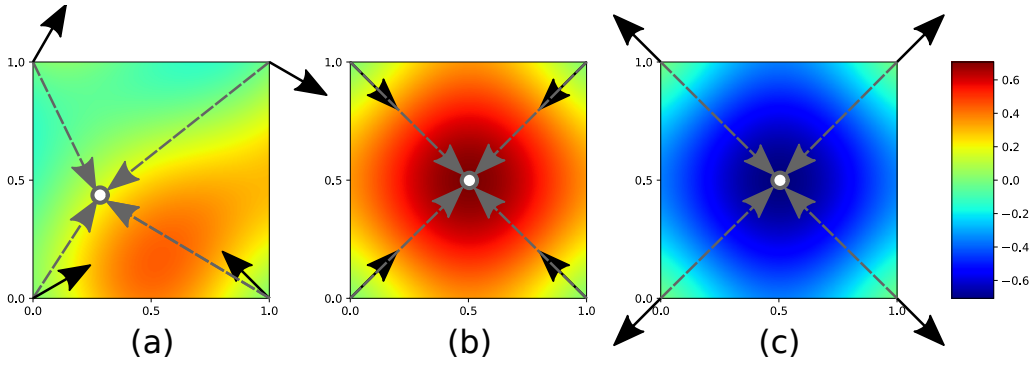


Figure 3.9 : Affichage de la fonction de bruit de Perlin 2D dans une cellule. (a) Un résultat avec des vecteurs gradient aléatoires. (b) Vecteurs gradient qui donnent le maximum global pour la valeur du bruit. (c) Vecteurs gradient qui donnent le minimum global pour la valeur du bruit. L'optimum est atteint au centre de la cellule. Les vecteurs sont raccourcis pour la figure, leur longueur réelle étant 1.

3.3.4.2 Mouvement brownien fractionnaire

Une technique procédurale bien connue au sein de la communauté Informatique Graphique permet de générer des textures aléatoires à partir de bruits passe-bas [EMP+02; LLC+10; PJH16]. Un motif élémentaire aléatoire est associé à chaque bande de bruit, avec un contenu fréquentiel limité à un certain intervalle. De manière plus formelle, les motifs sont calculés

en utilisant une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ définie comme étant la somme de n contributions d'une fonction de bruit, de la manière suivante :

$$f(\mathbf{p}) = \sum_{j=0}^{n} w_i \eta(s_j \mathbf{p}) \tag{3.13}$$

où la fonction de bruit η renvoie une valeur dans l'intervalle $[-1, 1]$ pour tout point $\mathbf{p} = (x,y)$ du plan 2D, multipliée par un facteur s_j. La valeur de bruit résultante est elle-même multipliée par un facteur d'amplitude w_j. Un choix courant pour η est le bruit de Perlin 2D [Per85] tel que défini à la section précédente.

Avec ce choix, l'intervalle réel des valeurs de η est $[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$, ce qui peut être montré algébriquement. Le maximum global $\frac{1}{\sqrt{2}}$ est atteint par η au point $(\frac{1}{2},\frac{1}{2})$ pour les vecteurs gradient $\vec{\mathbf{g}_A} = (\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})$, $\vec{\mathbf{g}_B} = (-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})$, $\vec{\mathbf{g}_C} = (-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, et $\vec{\mathbf{g}_D} = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$. Symétriquement, le minimum global $-\frac{1}{\sqrt{2}}$ est atteint au même point avec les vecteurs $\vec{\mathbf{g}_A} = (-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, $\vec{\mathbf{g}_B} = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, $\vec{\mathbf{g}_C} = (\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})$, et $\vec{\mathbf{g}_D} = (-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})$. En multipliant le domaine sur lequel η est évaluée, son contenu fréquentiel peut être facilement ajusté. Par exemple, les fréquences contenues dans $\eta(2\mathbf{p})$ sont deux fois plus élevées que dans $\eta(\mathbf{p})$. Les paramètres (w_j, s_j) sont typiquement définis par $s_j = 2^j s$ et $w_j = w/2^j$, où s et w sont contrôlés par l'utilisateur. Ce modèle produit des motifs fractals qui simulent du mouvement brownien Fractionnaire (*Fractional Brownian Motion*, FBM). Nous réécrivons la fonction de bruit fractal de la manière suivante :

$$f_{w,s,n}(\mathbf{p}) = w\, f_n(s\,\mathbf{p}) = w \sum_{j=0}^{n} \frac{1}{2^j} \eta(2^j s\,\mathbf{p}) \tag{3.14}$$

En se basant sur la fonction définie précédemment, nous considérons la grille de pixels $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{N^2}$ d'une image d'entrée carrée. Nous définissons une fonction de déformation spatiale de la manière suivante :

$$\mathcal{D}(\mathbf{p}_i) = \mathbf{p}_i + w \frac{N}{s} \begin{bmatrix} f_n(\tau_x(\mathbf{p}_i)) \\ f_n(\tau_y(\mathbf{p}_i)) \end{bmatrix} \tag{3.15}$$

où

$$\tau_x(\mathbf{p}_i) = \left(\frac{s}{N} x_i + r_x, \frac{s}{N} y_i\right)$$
$$\tau_y(\mathbf{p}_i) = \left(\frac{s}{N} x_i, \frac{s}{N} y_i + r_y\right)$$

transpose les coordonnées de $\mathbf{p}_i$ dans le domaine de la fonction de bruit en fonction des valeurs de s et w. Les deux scalaires de décalage r_x et r_y sont des valeurs arbitraires aléatoires qui permettent de s'assurer que la fonction de bruit η est échantillonnée dans différentes régions sans chevauchement pour chaque composante du champ de déplacement. Nous utilisons $r_x = 2$ et $r_y = 5$ dans nos expériences.

Ce modèle produit des déformations riches contenant de multiples fréquences contrôlées, illustrées à la figure 3.10. Les paramètres w, s et n sont définis par l'utilisateur. Comme

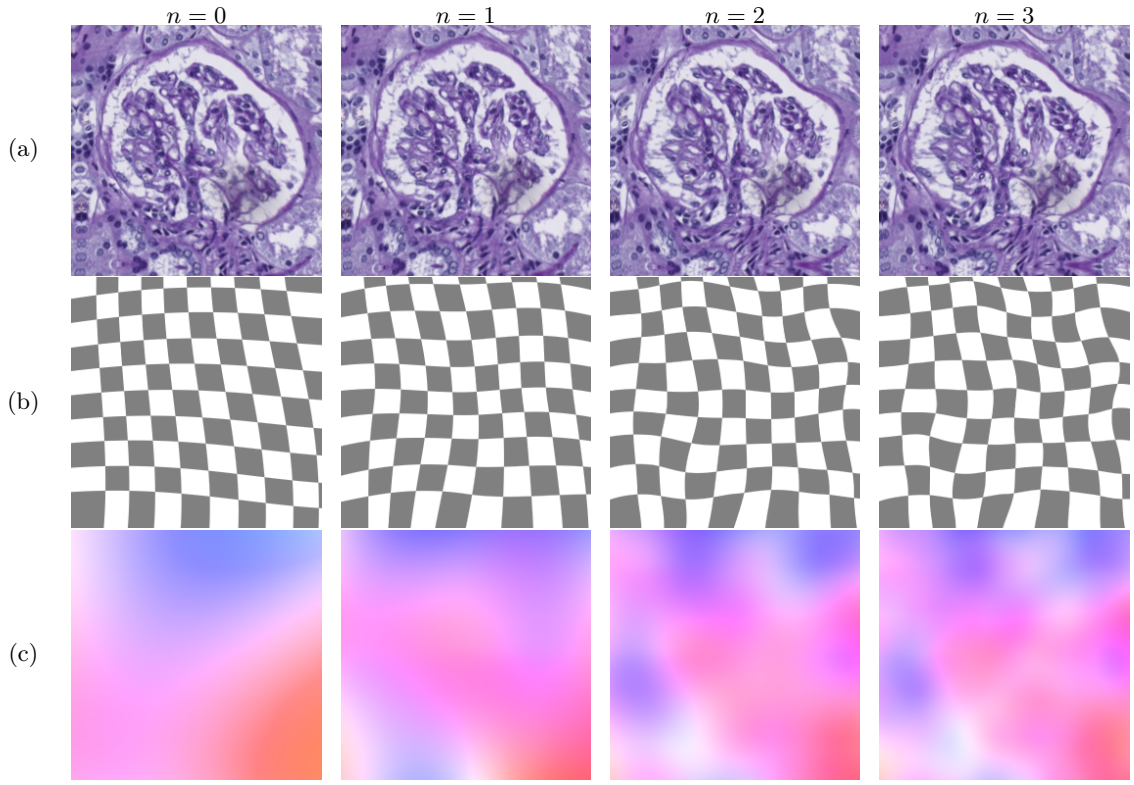


Figure 3.10 : Exemples de déformations FBM utilisant une fonction de bruit de Perlin 2D, pour $s = 1$ et $w = 0{,}2$. De gauche à droite, le nombre de termes de bruit augmente, de $n = 0$ à $n = 3$. (a) Images déformées. (b) Déformation appliquée à un échiquier. (c) Champ de déplacement codé en HSV.

pour le modèle RDF, l'amplitude maximale appropriée pour les déformations est difficile à déterminer intuitivement. La figure 3.11 montre les distributions des distances de déplacement pour différentes valeurs de s et w, avec $n = 3$, et met en évidence une relation linéaire entre w et les distances pour une valeur fixée de s. La figure A.7 en annexe montre que le comportement du modèle est similaire aux autres modèles étudiés en ce qui concerne le taux de repli et la préservation des angles et des distances. Pour la suite nous fixons $n = 3$, afin d'obtenir des déformations riches à différentes échelles.

3.4 Implémentation et protocoles d'évaluation

3.4.1 Implémentation

Nous avons implémenté les modèles de déformation décrits dans la section 3 en python3. Pour les *Random Displacement Fields* et les *Grid-Based Deformations*, nous avons utilisé le module *Tensorflow Addons*, qui fonctionne sur GPU. Pour les déformations CPAB, nous

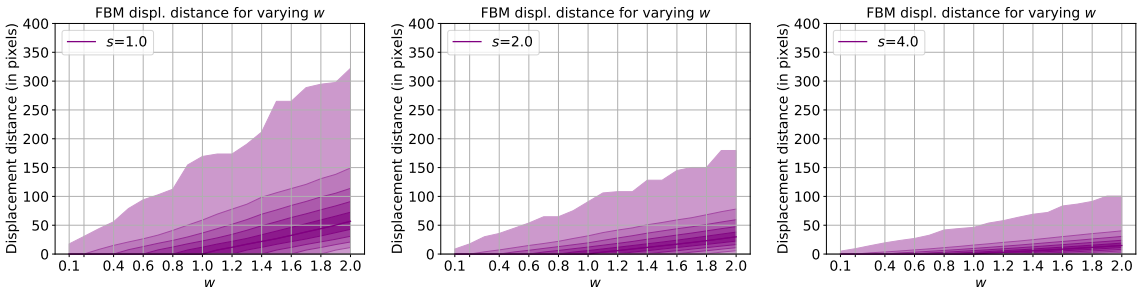


Figure 3.11 : Distribution des distances de déplacement pour les déformations FBM, pour différentes valeurs de paramètre. Pour chaque ensemble de paramètres, les distances de déplacement ont été échantillonnées depuis un ensemble aléatoire de 4 champs de déplacement de taille $N = 256$ (environ 262 000 mesures). La courbe centrale représente la médiane de la distribution, tandis que les autres courbes correspondent aux percentiles.

avons utilisé une implémentation optimisée sous Pytorch par Detlefsen *et al.* [Det18]. Pour CNB-MVN-MLS et FBM, les modèles ont été implémentés sur CPU, avec une implémentation sur GPU en cours d'investigation. Le temps requis pour échantillonner un champ de déplacement et déformer une image de taille 256×256 varie selon la méthode et le matériel. Pour CNB-MVN-MLS et FBM, il faut compter environ 1.5 secondes, tandis que RDF et GBD requièrent 60 millisecondes par image. Pour les déformations CPAB, le temps de calcul varie fortement en fonction de la finesse du pavage. Avec $K = 100$, il faut environ 20 secondes pour produire la base des champs CPA et calculer la matrice de covariance de la distribution normale multivariée, tandis qu'il faut moins de 2 secondes pour $K = 25$. Les déformations aléatoires sont ensuite obtenues en quelques millisecondes.

Notre code est disponible publiquement sur un dépôt GitHub[1]. Nous fournissons une chaîne de traitement complète pour l'évaluation de l'impact des modèles de déformation considérés pour l'augmentation de n'importe quelle base d'images 2D et leur segmentation par un U-Net.

3.4.2 Protocoles d'évaluation

Nous formulons la segmentation de glomérules comme un problème à deux classes où chaque pixel d'un patch de WSI reçoit un label glomérule ou un label tissu. Dans ce chapitre, chaque patch peut contenir un glomérule complet, ou jusqu'à trois glomérules partiels, ou aucun glomérule, c'est-à-dire seulement du tissu. Pour évaluer l'impact de l'augmentation par déformation sur les résultats de segmentation du U-Net, nous proposons deux protocoles distincts, avec une préparation des données différente pour chacun.

[1] https://github.com/ASTex-ICube/spatial_deformations_aug

3.4.2.1 Comparaison des modèles de déformation

Le premier protocole vise à effectuer une comparaison des différents modèles de déformation. Pour ce faire, il faut s'assurer que les paramètres utilisés pour chaque modèle soit aussi proche que possible des valeurs de paramètres optimales. C'est ici qu'intervient la notion de méta-apprentissage évoquée au chapitre 2. Nous utilisons une stratégie de méta-apprentissage inspiré de *RandAugment* [CZSL20b]. Pour chaque modèle, nous considérons le ou les paramètres décrits dans la section 3, et sélectionnons à la main, pour chacun, quelques valeurs. Nous entraînons ensuite un U-Net pour chaque combinaison de paramètres. Ces valeurs sont choisies pour couvrir de larges intervalles de fréquence et d'amplitude, afin d'obtenir différentes quantités de distorsion. Il faut noter toutefois que le méta-apprentissage est très coûteux (un entraînement de U-Net pouvant prendre entre 2 et 8 heures sur une GPU Nvidia P100 selon les cas), nous ne pouvons donc pas chercher les paramètres optimaux de manière exacte. Nous cherchons à fournir de bons intervalles pour les paramètres et mettre en lumière l'influence des caractéristiques des déformations. Nous effectuons en pratique cinq entraînements pour chaque jeu de paramètres et prenons la moyenne des résultats. Les valeurs de paramètres sélectionnées pour chaque modèle de déformation considéré sont les suivantes :

- RDF (σ, α)
  - $\sigma = \{5, 10, 20\}$
  - $\alpha = \{50, 100, 200, 400\}$
- GBD ordre 2 (TPS) (n, σ)
  - $n = \{3, 5, 10\}$
  - $\sigma = \{5, 10, 20, 50\}$
- GBD ordre 3 (n, σ)
  - $n = \{3, 5, 10\}$
  - $\sigma = \{5, 10, 20, 50\}$
- CNB-MVN-MLS (σ)
  - $\sigma = \{5, 10, 15, 30, 100\}$
- CPAB (K, σ)
  - $K = \{9, 25, 100\}$
  - $\sigma = \{1, 2, 5, 10\}$
- FBM (s, w)
  - $s = \{1, 2, 4\}$
  - $w = \{0{,}1, 0{,}35, 0{,}7, 1\}$

Pour ce protocole, les augmentations sont calculées « hors ligne », avant l'entraînement. Nous sélectionnons aléatoirement 20 patchs de taille 512×512 dans la base d'entraînement, 10 dans la base de validation et 50 dans la base de test, tous avec glomérule. Nous ajoutons le même nombre de patchs à chaque base, cette fois sans glomérule. Nous choisissons ici volontairement un petit nombre d'exemples afin de pouvoir plus facilement mettre en évidence les différences dans les résultats. De chaque patch, nous extrayons cinq sous images de taille 256×256. Ces nouveaux patchs seront présentés en entrée du U-Net. La base de test restera la même pour toutes les expériences de ce chapitre. Pour chaque modèle de déformation et chaque jeu de paramètres, nous appliquons 10 déformations à chaque image d'entraînement. Chacune de ces 10 déformations est échantillonnée aléatoirement, ce qui veut dire que nous calculons 2 000 champs de déplacement pour chaque jeu de paramètres de chaque méthode. Ensuite, aux jeux d'entraînement et de validation, nous appliquons les augmentations de base, c'est-à-dire toutes les rotations à 90° et symétries possibles, ce qui multiplie le nombre d'images par 7. Nous avons donc au total 14 000 images dans chaque jeu d'entraînement.

3.4.2.2 Impact en fonction de la taille des jeux de données

Le deuxième protocole vise à trouver la manière la plus pertinente d'utiliser une méthode de déformation donnée et l'impact de la dite méthode quand la taille du jeu de données augmente. Pour les expériences de cette section, nous considérons uniquement le modèle GBD à l'ordre 3, avec pour paramètres $n = 3$ et $\sigma = 20$. Les augmentations sont ici calculées « en ligne » durant l'entraînement, avec une probabilité de 0,5 pour chaque augmentation parmi les rotations et les symétries. C'est la manière classique d'utiliser les augmentations dans la plupart des travaux autour du *Deep Learning*. Nous appliquons en plus une déformation avec une probabilité p prenant sa valeur dans {0, 0,25, 0,5, 0,75, 1} selon les cas. Les tailles originales des jeux d'entraînement et de validation (N_{train}, N_{val}) prennent ici les valeurs (20, 10), (100, 30), (300, 100) et (600, 200). Comme pour le premier protocole, nous doublons chaque jeu par des patchs sans glomérules et nous extrayons cinq images

de taille 256×256 par patch. Le jeu de test reste le même. De nouveau, pour chaque jeu de données et chaque probabilité de déformation, nous effectuons cinq entraînements et moyennons les résultats.

3.4.2.3 Comparaison entre les expériences

Puisque nous avons un faible nombre de répétitions (cinq entraînements par répétition), nous utilisons le test statistique de Wilcoxon–Mann–Whitney [Wil45; MW47] pour obtenir une indication sur l'équivalence de deux jeux de paramètres par rapport à leur indice de Dice obtenu. En considérant les indices de Dice de deux expériences comme des observations indépendantes de deux distributions X and Y, nous testons l'hypothèse nulle H0 : les distributions des deux populations sont égales. En notant X_1,...,X_n et Y_1,...,Y_m les échantillons tirés de chaque distribution, nous pouvons définir la statistique U comme :

$$U = \sum_{i=1}^{n} \sum_{j=1}^{m} S(X_i, Y_j), \tag{3.16}$$

avec

$$S(X,Y) = \begin{cases} 1 & \text{if } X > Y, \\ \frac{1}{2} & \text{if } Y = X, \\ 0 & \text{if } X < Y. \end{cases} \tag{3.17}$$

Avec $n = m = 5$, nous pouvons rejeter l'hypothèse nulle quand $U \leq 2$ ou $U \geq 23$ d'après les tables [Wil45].

3.5 Modèles de déformation : expériences et résultats

3.5.1 Modèles de déformation : résultats de segmentation

3.5.1.1 Premier protocole

Chaque entraînement dure 30 époques, ce qui est expérimentalement suffisant pour dépasser le point de sur-apprentissage, et nous gardons le modèle avec l'erreur de validation la plus basse. Au moment du test, nous évaluons le modèle selon quatre métriques : l'indice de Dice (ou F1-score), la précision, le rappel et la spécificité. Toutes sont moyennées sur cinq répétitions. Nous rappelons les formules de ces métriques :

- Dice score $= \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$
- Précision $= \frac{|\text{Vrais positifs}|}{|\text{Vrais positifs}| + |\text{Faux positifs}|}$
- Rappel $= \frac{|\text{Vrais positifs}|}{|\text{Vrais positifs}| + |\text{Faux négatifs}|}$

- Spécificité $= \frac{|\text{Vrais négatifs}|}{|\text{Vrais négatifs}|+|\text{Faux positifs}|}$

Par souci de clarté, nous nous concentrerons sur l'indice de Dice moyen (exprimé en %), mais les résultats pour toutes les métriques se trouvent en annexe. Les indices de Dice moyens pour tous les modèles de déformation sont présentés dans la tableau 3.7. Le meilleur score est indiqué en rouge, tandis que les scores bleus sont les scores équivalents au meilleur score d'après le test de Wilcoxon. Nous notons toutefois qu'il peut y avoir une différence jusqu'à 6 points d'indice de Dice moyen pour deux expériences considérées comme équivalentes au sens du test de Wilcoxon. Nous ne pouvons donc pas considérer les paramètres de ces deux expériences comme interchangeables. Cependant, le test de Wilcoxon nous donne un intervalle à l'intérieur duquel nous pouvons supposer, avec une bonne confiance, que se trouvent les paramètres optimaux. Cette information peut servir à affiner la grille de recherche des paramètres. En particulier, afin de donner les meilleures limites possibles, nous avons essayé d'avoir un score non équivalent aux extrémités de chaque ligne comprenant le meilleur indice de Dice. C'est la raison pour laquelle quelques valeurs supplémentaires se trouvent dans les tableaux, par rapport à celles annoncées dans la section précédente. Nous donnons aussi l'indice de Dice moyen dans le cas où seules les augmentations de base sont utilisées, c'est-à-dire sans les déformations : $62{,}31 \pm 4{,}01$. Il faut noter qu'il peut y avoir plusieurs maxima locaux dans le champ des indices de Dice potentiels. Les valeurs de paramètre choisies déterminent un maximum qui n'est pas garanti d'être le maximum global.

En se concentrant sur un seul modèle de déformation à la fois, l'impact des paramètres apparaît clairement. Par exemple, dans le cas de la méthode RDF, certains paramètres font baisser les performances, comme ($\sigma = 20$, $\alpha = 50$), tandis que d'autres peuvent augmenter l'indice de Dice moyen de 62% à plus de 75%. Toutes les valeurs de paramètres sélectionnées pour la méthode GBD-3 améliorent les résultats, le couple ($n = 3$, $\sigma = 20$) apportant la meilleure amélioration.

σ \ α	100	200	400	800
5	75,86 ± 2,33	72,75 ± 3,87	53,42 ± 3,99	48,07 ± 4,48
10	72,50 ± 5,23	77,48 ± 1,84	78,92 ± 3,70	77,87 ± 1,75
20	66,53 ± 5,19	73,67 ± 4,81	77,29 ± 2,92	78,59 ± 2,56

σ \ α	2400	5000
40	75,54 ± 2,92	68,99 ± 2,23

(a) Indice de Dice moyen en fonction des paramètres de RDF.

n \ σ	5	10	20	50	70
3	76,88 ± 1,35	81,77 ± 1,84	85,93 ± 1,62	85,10 ± 1,39	80,25 ± 1,47
5	79,31 ± 2,07	84,07 ± 3,44	81,14 ± 3,19	73,81 ± 5,42	
10	81,13 ± 3,11	84,48 ± 1,65	75,06 ± 3,28	70,11 ± 1,97	

(c) Indice de Dice moyen en fonction des paramètres de GBD-3.

σ	5	10	15	30	100
	82,12 ± 1,16	85,01 ± 1,21	85,25 ± 1,82	83,37 ± 1,61	70,41 ± 2,98

(e) Indice de Dice moyen en fonction des paramètres de CNB-MVN-MLS.

n \ σ	5	10	20	50
3	78,84 ± 2,24	81,76 ± 3,83	84,34 ± 1,81	85,02 ± 1,58
5	80,14 ± 1,16	83,20 ± 2,43	85,41 ± 1,25	78,88 ± 3,51
10	80,28 ± 2,02	84,02 ± 3,17	76,48 ± 3,21	71,71 ± 4,73

(b) Indice de Dice moyen en fonction des paramètres de GBD-2.

s \ w	0,1	0,35	0,7	1
1	82,30 ± 2,86	85,21 ± 2,39	80,21 ± 2,98	75,79 ± 5,68
2	78,05 ± 4,06	85,18 ± 2,02	82,70 ± 3,45	80,91 ± 3,13
4	76,65 ± 1,66	83,91 ± 1,89	84,48 ± 2,73	80,27 ± 3,29

(d) Indice de Dice moyen en fonction des paramètres de FBM.

n \ σ	0,5	1	2	5	10
3	78,71 ± 1,93	83,34 ± 4,11	83,99 ± 2,98	81,54 ± 2,22	77,38 ± 3,62
5	81,92 ± 2,49	84,76 ± 1,96	84,64 ± 2,26	81,94 ± 1,36	73,97 ± 3,77
10	81,03 ± 3,61	84,83 ± 1,74	83,93 ± 2,56	77,90 ± 3,03	73,71 ± 1,71
15	82,19 ± 2,88	83,84 ± 1,83			

(f) Indice de Dice moyen en fonction des paramètres de CPAB.

Table 3.7 : Performances de segmentation de glomérule du U-Net, mesurées par l'indice de Dice moyen, en fonction des valeurs de paramètres des méthodes de déformation considérées. Le meilleur score est indiqué en rouge, tandis que les scores bleus sont les scores équivalents au meilleur score d'après le test de Wilcoxon.

Nous résumons à la figure 3.12 les meilleurs scores obtenus pour chaque modèle de déformation et les valeurs de paramètres correspondantes. Le meilleur score est obtenu avec la méthode GBD-3, mais nous notons que mis à part la méthode RDF, tous les résultats sont très proches. En considérant la variance et le faible nombre de répétitions, nous pouvons affirmer que toutes les méthodes (à part RDF) ont des performances équivalentes. Cette affirmation est corroborée par le test de Wilcoxon, utilisé pour une comparaison entre les meilleurs scores de chaque méthode.

Il est important de noter que la méthode RDF n'est pas nécessairement inférieure : nous n'avons pas trouvé de paramètres donnant des performances similaires aux autres méthodes, ce qui indique qu'ils n'existent pas ou qu'ils sont plus difficiles à trouver. Cela rejoint l'observation faite à la section 3, selon laquelle la méthode RDF est plus difficile à contrôler que les autres méthodes. Nous notons également que le changement d'interpolation pour la méthode GBD n'apporte pas de changement notable aux résultats, avec un écart de seulement 0,5 point. Le modèle CNB-MVN-MLS donne des performances à hauteur des autres modèles, mais avec un seul paramètre à optimiser, ce qui réduit significativement la complexité de la recherche du paramètre optimal. Les résultats obtenus nous

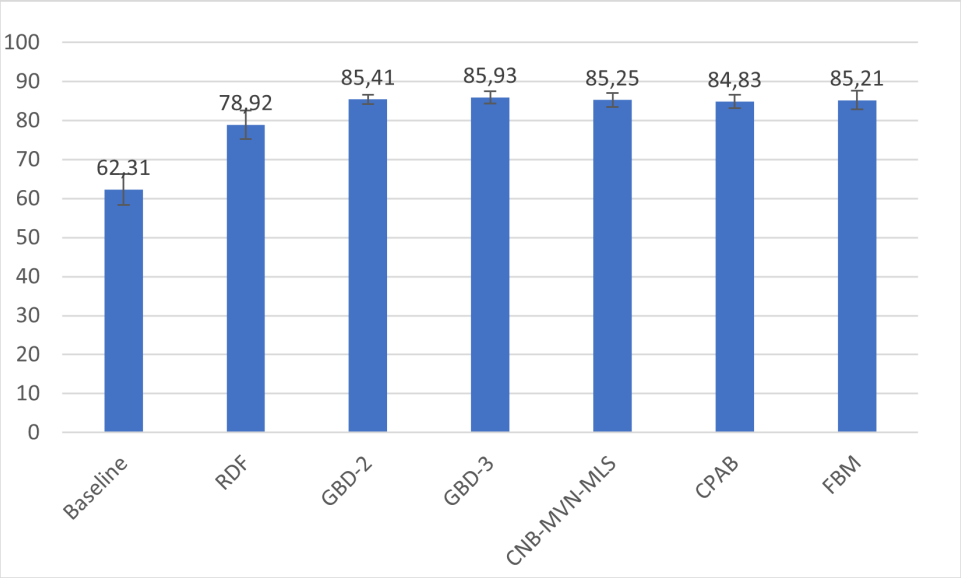


Figure 3.12 : Meilleur indice de Dice moyen pour chaque méthode de déformation.

mènent à la conclusion que le choix de la méthode de déformation pour l'augmentation n'est pas crucial, contrairement au réglage des paramètres. Par ailleurs, puisque toutes les méthodes ont des scores maximaux similaires, nous pouvons affirmer avec une bonne confiance que les valeurs obtenues sont proches des maxima globaux. Les performances observées sont cohérentes avec le fait que toutes les méthodes reposent sur l'échantillonnage d'un bruit blanc filtré, comme vu à la section 3.

Pour être complets, nous avons réalisé une expérience complémentaire où le jeu de données d'entraînement mélange des déformations obtenues avec plusieurs valeurs de paramètres. En particulier, nous avons appliqué le premier protocole avec la méthode GBD-3 en fixant $n = 3$ et en tirant au hasard un σ parmi les valeurs 10, 20 et 50 pour chacune des 2 000 transformations à effectuer. Nous obtenons avec cette stratégie un indice de Dice moyen de $81{,}16 \pm 2{,}87$, le même que pour le couple $(n = 3, \sigma = 10)$. Ainsi, il apparaît que mélanger des valeurs de paramètres n'ajoute pas de diversité au jeu de données et semble même retenir les performances au plus petit score correspondant aux paramètres utilisés.

3.5.1.2 Deuxième protocole

Les résultats obtenus avec le second protocole sont résumés au tableau 3.8. Comme attendu, les indices de Dice moyens augmentent avec le nombre d'images d'entraînement et de validation. Nous notons également que la variance est plus faible pour les bases de 300 et 600 images que pour les bases de 20 et 100 images, ce qui indique une meilleure robustesse du U-Net. Quand $p > 0$, l'amélioration est marginale entre 300 et 600 images, ce qui est attesté par le test de Wilcoxon.

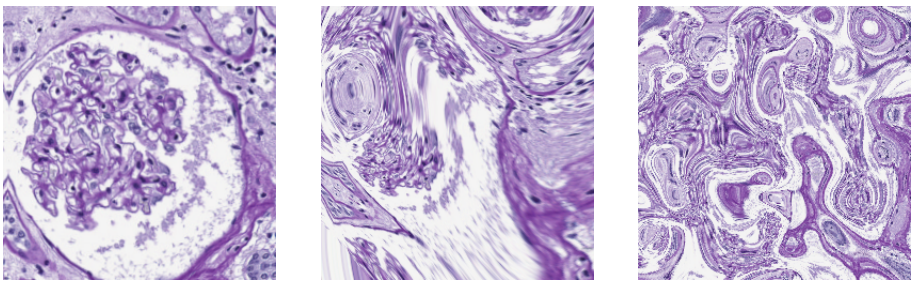
p \ N_{train}	20	100	300	600
0	64,58 ± 2,54	70,98 ± 4,19	83,91 ± 1,98	88,13 ± 1,45
0,25	68,51 ± 3,69	76,02 ± 2,23	88,45 ± 0,59	90,36 ± 1,33
0,5	70,50 ± 2,28	76,31 ± 3,55	88,96 ± 0,55	89,65 ± 0,72
0,75	73,03 ± 2,48	78,02 ± 3,92	87,56 ± 2,25	90,75 ± 1,36
1	73,71 ± 1,96	80,05 ± 2,75	87,57 ± 1,64	88,05 ± 2,88

Table 3.8 : Dice score moyen en fonction de la taille des jeux de données et de la probabilité d'utilisation des déformations.

En ce qui concerne l'impact de p, nous notons que plus le jeu de données est petit, plus l'amélioration potentielle est importante. C'est particulièrement clair pour $N_{train} = 20$: l'indice de Dice moyen augmente avec la probabilité de déformation. Pour For $N_{train} = 300$, augmenter p au dessus de 0,25 n'apporte pas d'amélioration significative. Pour $N_{train} = 600$, p n'a pas d'impact clair sur les résultats. Toutes ces observations sont corroborées par le test de Wilcoxon.

Comme dit précédemment, ce second protocole, consistant à appliquer un ensemble de transformations avec une certaine probabilité p, avec $p = 0{,}5$ en général, correspond en fait à l'utilisation habituelle de l'augmentation de données pour le *Deep Learning*. Toutefois, nous observons de meilleures performances avec le premier protocole. Ce dernier applique plusieurs transformations, dont 10 déformations, à chaque image avant même la phase d'entraînement, ce qui amène potentiellement à un jeu de données montré au modèle de segmentation plus divers, et ce dès les premières époques, au coût d'un temps d'entraînement plus long.

3.5.1.3 Impact des distorsions



(a) (b) (c)

Figure 3.13 : Exemples de déformations obtenues avec la méthode GBD-3. (a) $n = 3$, $\sigma = 5$. Peu de distorsion et +15 points d'indice de Dice moyen. (b) $n = 3$, $\sigma = 50$. Distorsions plus importantes et +23 points d'indice de Dice moyen. (c) $n = 10$, $\sigma = 50$. Haut taux de repli (plus de 40%), +8 points d'indice de Dice moyen.

Nous observons sur les images déformées (figure 3.13) que les déformations avec peu d'amplitude, souvent conservées pour garder les exemples « réalistes », donnent de moins

bonnes performances d'augmentation que des déformations de plus haute amplitude, qui peuvent sembler moins réalistes. Comme présenté à la section 3, les déformations de haute amplitude impliquent moins de préservation des distances et des angles dans les images résultat. Le fait de modifier les angles a des conséquences importantes sur les formes, ce qui peut mener à une plus grande robustesse du modèle de segmentation. Les déformations provoquant du repli dégradent les performances de segmentation à mesure que le taux de repli augmente. Cependant, des déformations contenant un peu de repli peuvent apporter de meilleures performances. Par exemple, les déformations GBD-3 avec $n = 3$ et $\sigma = 5$ n'induisent que peu de distorsions et mènent à une augmentation de 15 points d'indice de Dice moyen par rapport à la base. Pour le même modèle, les déformations avec $n = 3$ et $\sigma = 50$ provoquent plus de distorsions, avec un tiers des images contenant du repli (voir figure A.3 en annexe), et donne un gain de 23 points. Les déformations avec $n = 10$ et $\sigma = 50$ ont un haut taux de repli (plus de 40%) et une amélioration de seulement 8 points. Notre interprétation est que le repli crée de nouveaux motifs introuvables dans les images d'origine, introduisant ainsi une variété bénéfique pour l'entraînement de U-Net. Cependant, si ces motifs deviennent dominants, l'apparence originale des images (en particulier les structures topologiques) peut être complètement perdue, dégradant ainsi les performances de segmentation. La comparaison des gains de performance révèle également que la richesse fréquentielle des champs de déplacement n'est pas critique. Par exemple, la méthode GBD-3 avec $n = 3$ produit typiquement des déformations de basse fréquence et donne des gains similaires aux gains obtenus avec FBM pour $s = 1$ ou CNB-MVN-MLS, qui mélangent des basses et des hautes fréquences. Les déformations qui ne contiennent que des hautes fréquences tendent à être moins bonnes que les déformations contenant des basses fréquences, comme nous pouvons le voir avec la méthode RDF avec $\sigma = 5$. Notons que les résultats obtenus en termes de performances changeront en fonction des images utilisées, puisque ces dernières contiennent un contenu différent avec leurs propres fréquences. L'habileté des modèles de déformation à reproduire ces fréquences peut être un point crucial et mériterait d'autres expériences avec des images au contenu fréquentiel varié.

3.5.2 Modèles de déformation : limitations

Nos protocoles étant basés sur la stratégie RandAugment [CZSL20b], ils sont extrêmement coûteux en temps de calcul. Pour le premier protocole, nous avons lancé 420 entraînements, ce qui correspond à environ 840 heures GPU sur des NVidia Tesla P100 SXM2 avec 16 Go RAM. Notre choix de paramètres et les tests statistiques nous fournissent seulement des bornes sur les meilleures valeurs possibles, et estimer des bornes plus précises ou les valeurs exactes pour chaque méthode demanderait des ressources en calcul supplémentaires. Aussi, plus de répétitions pour chaque jeu de paramètres permettrait d'obtenir des bornes plus claires entre les paramètres.

Il est important de noter que nos conclusions ne peuvent être généralisées à d'autres tailles de jeu de données ou d'autres types d'images [CZSL20a] : la taille du jeu d'entraînement a un impact sur le processus d'optimisation des hyperparamètres d'un modèle de *Deep*

Learning. Afin de mettre en évidence les différences entre les gains de performances permis par les modèles de déformation et leurs paramètres, nous n'avons utilisé que 20 images de notre jeu initial composé de 660 images, sans compter les images sans glomérules, dont le nombre n'est pas critique ici. Ce choix nous a permis de montrer que l'usage de déformations comme technique d'augmentation de données permet un gain potentiel de plus de 20 points d'indice de Dice moyen, mais au coût d'une variance relativement élevée. Le tableau 3.8 montre que les variances sont plus élevées dans le cas où 20 ou 100 images sont utilisées que dans le cas où 300 ou 600 sont utilisées. Ainsi le faible nombre d'images est un des facteurs, avec le nombre de répétitions, contribuant aux larges bornes obtenues pour l'estimation des meilleurs paramètres.

En ce qui concerne d'autre types de jeu de données, l'impact des modèles de déformation est potentiellement différent, puisque changer les images change les formes et les textures. Nous n'avons pas conduit d'expériences avec des images autres que des patchs de coupes complètes rénales, mais le cadre et le code que nous fournissons permet de travailler sur n'importe quel type d'images et/ou taille de jeu de données.

3.6 Conclusion et pistes sur les modèles de déformation

Dans ce chapitre, nous avons évalué l'impact de déformations spatiales aléatoires, utilisées comme augmentation dans le cadre de la segmentation d'images histopathologiques à l'aide d'un modèle U-Net. Nous nous sommes concentrés sur l'étude de déformations résultant du filtrage d'un bruit blanc. Nous avons examiné et comparé une sélection de modèles de l'état de l'art, en considérant les aspects statistiques et géométriques. Nous avons également proposé une méthode originale sensible au contenu, CNB-MVN-MLS, basée sur la détection et le déplacement de centres de noyaux de cellules, et impliquant un seul paramètre à ajuster.

Nous avons comparé les performances offertes par ces modèles de déformation lorsqu'utilisés pour l'augmentation et utilisé une stratégie de méta-apprentissage pour optimiser leurs paramètres et trouver des bornes appropriées. Nous avons trouvé que les différents modèles ont des performances similaires (mesurées en indice de Dice moyen) et qu'utiliser le bon intervalle de paramètres peut améliorer l'indice de Dice moyen jusqu'à 23 points sur notre tâche de segmentation. Nous avons montré que les déformations avec une amplitude relativement élevée donnent les meilleures performances, alors que les précédents travaux utilisent seulement des déformations de faible amplitude. Les déformations comprenant des basses fréquences donnent également les meilleurs performances. Notre modèle CNB-MVN-MLS donne d'aussi bons résultats que les autres modèles considérés mais avec seulement un seul paramètres à contrôler, ce qui facilite l'estimation du meilleur paramètre. Nous avons également montré que la manière d'utiliser les déformations (pré-calculées ou appliquées aléatoirement pendant l'entraînement) peut avoir un impact sur les résultats.

Pour des travaux futurs, il serait intéressant de faire le pont entre nos deux protocoles et de déterminer précisément la meilleure manière d'utiliser les déformations (en ligne ou hors ligne). En utilisant plus d'images et/ou d'entraînements, nous pourrions également diminuer la variance entre les entraînements et ainsi obtenir de meilleurs bornes pour les paramètres optimaux. En suivant la piste de notre méthode CNB-MVN-MLS, nous pourrions développer des modèles améliorant encore la corrélation entre les vecteurs de déplacement et le contenu des images. Des méthodes basées sur de la simulation physique [NMK+06] prennent en compte les propriétés matérielles et mécaniques, au prix d'un coût de calcul prohibitif pour la génération de champs de déformation, ce qui nécessiterait d'identifier un compromis. Il serait également intéressant de considérer des avancées telles que les Réseaux de Transformation Spatiale (*Spatial Transformer Networks*, STN) [JSZK15; DFH18]. Ces derniers sont des modules qu'il est possible d'ajouter à tout réseau de neurones et qui appliquent des transformations aux images d'entrée. Ils sont entraînés comme une partie intégrante du modèle. Les STNs ont été utilisés pour de nombreuses applications [SWL+16; KJC16; DFH18], y compris pour la segmentation d'images médicales [LCC+19; LPP+19; PCSD20]. À notre connaissance, ils n'ont pas été envisagés pour l'analyse d'images histopathologiques. Nous pourrions étudier les performances d'un U-Net couplé à un STN effectuant des déformations spatiales. Les modèles auto-attentifs (*self-attention models*) ou Réseaux de Transformation (*Transformer networks*) [VSP+17] ont aussi été récemment introduits dans le champ de la segmentation d'images médicales en combinaison avec des modèles U-Net [PTR+21; GZM21]. La contribution à l'augmentation de données de ces approches reste à explorer, en particulier en ce qui concerne les déformations spatiales. Il existe enfin des modèles apparentés tels que les Réseaux Convolutifs Déformables (*Deformable Convolutional Networks*, DCN) [DQX+17], qui apprennent des décalages par pixel plutôt que des transformations spatiales. Les DCNs ont été également appliqués aux images médicales, en combinaison avec un modèle U-Net [JMP+19; LPZQ20; ZLXL20]. Nous ne suivons pas cette approche ici, car elle augmente considérablement le nombre de paramètres à estimer.

Chapitre 4

Synthèse de texture par l'exemple

Nous nous intéressons dans ce chapitre à une première approche de synthèse de glomérules, basée sur l'utilisation d'outils issus de la communauté de l'informatique graphique. Nous commencerons par poser le problème de la synthèse de texture du point de vue informatique graphique, avant de présenter les grandes familles de méthodes existantes. Enfin, nous discuterons des résultats de synthèse de glomérule obtenus grâce à une méthode par réseaux de neurones pré-entraînés : la synthèse de texture par réseaux de neurones guidée multi-échelles (*Neural Multi-scale Guided Texture Synthesis*). En particulier, nous n'utiliserons pas les méthodes se basant sur un apprentissage dans cette partie. Notre contribution consiste en la combinaison d'outils issus de l'état de l'art en synthèse de texture pour la génération de patchs de glomérule, et de l'évaluation de ces patchs pour l'augmentation de données dans le cadre de la segmentation supervisée de glomérules.

4.1 Contexte et définitions

4.1.1 Les textures en informatique graphique

En informatique graphique, les textures sont des images 2D qui définissent l'apparence d'une scène. Ces scènes sont constituées de modèles géométriques 3D, en général sous la forme de surfaces triangulées munies de paramétrisations. Les textures sont plaquées sur ces surfaces et peuvent simuler leur relief ou des propriétés photométriques des matériaux tels que la couleur diffuse ou spéculaire. La figure 4.1 montre un exemple de texture pouvant définir l'apparence d'une surface.

De plus en plus d'applications proposent de naviguer en temps réel dans des environnements virtuels complexes, que ce soit dans un cadre vidéoludique (figure 4.2), muséographique (figure 4.3), architectural, ou encore de formation. Il peut notamment s'agir de

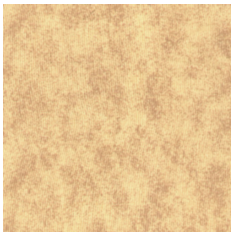


Figure 4.1 : Exemple de texture. Tiré de *Describable Texture Dataset* (DTD) [CMK+14]. DTD est une base de données comprenant 5640 textures réparties dans 47 catégories.

paysages naturels, de scènes urbaines, d'intérieurs de bâtiments, etc. Les textures contribuent au réalisme de ces environnements virtuels.

Figure 4.2 : Exemple de monde virtuel, issu de *Elden Ring* (From Software, Bandai Namco).

Il existe différents types de textures. Sylvain Lefebvre propose la classification suivante [Lef14] :

- Stochastique non structuré : aspect complètement aléatoire (pas de contours observables), propriétés de localité et de stationnarité.
- Stochastique structuré : aspect aléatoire avec contours observables.
- Structuré organisé : motifs structurés (motifs décoratifs par exemple).
- Régulier et presque-régulier : cas particulier de la catégorie précédente, avec en plus des propriétés de périodicité.

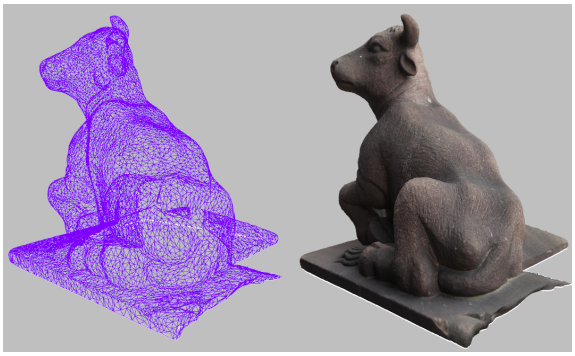


Figure 4.3 : Exemple d'objet numérisé.

La figure 4.4 présente un exemple pour chacune des catégories. En pratique, une texture peut appartenir à une ou plusieurs catégories selon l'échelle d'analyse. Ainsi, elle peut avoir un aspect régulier globalement, mais un aspect stochastique non structuré localement.

Figure 4.4 : De gauche à droite : exemple de texture stochastique non structuré, stochastique structuré, structuré organisé et régulier.

4.1.2 Synthèse de texture par l'exemple

La synthèse de texture par l'exemple (*by-example synthesis*) a pour objectif de créer de nouvelles textures à partir d'un échantillon donné en entrée, l'exemple. La texture de sortie peut, selon le besoin, être plus grande que l'originale. Les deux textures doivent sembler avoir été produites par un même processus sous-jacent lorsque observées par un humain [WLKT09], tout en étant différentes. Pour ce faire, deux étapes sont nécessaires [WLKT09] : l'analyse, qui cherche à définir un modèle pouvant avoir généré l'exemple, et la synthèse, qui produit de nouvelles textures grâce à ce modèle.

La synthèse de texture par l'exemple est un domaine particulier, car elle repose en général sur l'optimisation d'un critère de similarité entre deux images, dont la solution est connue : il s'agit de l'exemple lui-même. Or nous souhaitons produire de nouvelles textures, donc cette solution est à écarter. Tout dépend donc de la capacité de la méthode considérée à approcher suffisamment l'optimum pour que le résultat soit reconnu comme la même texture, tout en étant suffisamment éloigné pour que le résultat ne soit pas une

simple copie. La figure 4.5 montre un exemple de synthèse de texture par une méthode classique [PS00].

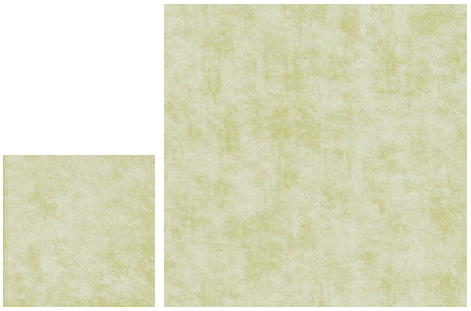


Figure 4.5 : À gauche : l'exemple. À droite : résultat d'une synthèse avec la méthode de Portilla *et al.* [PS00].

Pour répondre à la demande croissante de contenus hautement détaillés dans l'industrie graphique, la recherche en informatique graphique a développé de nombreuses méthodes de synthèse de texture par l'exemple [GSV+14; LSA+16; GSDC17; LSD21]. Wei *et al.* [WLKT09] et Raad *et al.* [RcDDM17] passent en revue le domaine à deux époques différentes, ce qui permet notamment de se rendre compte, à nouveau, de la place grandissante des réseaux de neurones dans la littérature. Le principal enjeu actuel est de parvenir à mettre au point des méthodes permettant de générer des textures non-homogènes (voir section suivante) très réalistes et de grande taille de façon contrôlée, avec possibilité d'édition interactive, et dans des représentations compactes adaptées au rendu en temps réel. Une difficulté réside par ailleurs dans l'évaluation de la qualité des résultats produits, cette dernière étant fondamentalement subjective et réalisée par un panel d'utilisateurs dans la plupart des articles [SBCH11; KDC17; WFE+17].

4.1.3 Les textures non-homogènes

En conséquence des attentes croissantes en termes de fidélité et de richesse des contenus texturés à étudier, le champ de l'informatique graphique doit faire face à des textures toujours plus complexes. C'est le cas des textures dites « non-homogènes » qui sortent de la classification évoquée précédemment. Une texture non-homogène présente des variations globales, qui vont de la présence de plusieurs motifs à la non-uniformité de la distribution d'un même motif (voir figure 4.6). L'homogénéité est donc une propriété qui dépend de l'échelle d'analyse et de la taille des motifs observables [Sau18]. Une même texture observée à plusieurs échelles, pourra être perçue comme homogène ou non-homogène, comme montré sur la figure 4.7. Les textures non-homogènes se trouvent à la frontière entre les textures classiques telles que définies précédemment et les images naturelles.

Le cas des textures non-homogènes est difficile à traiter pour la plupart des méthodes classiques de synthèse de texture par l'exemple, en raison notamment de l'ambiguïté sur le résultat recherché. Prenons l'exemple de la figure 4.8 : à partir de l'exemple proposé, il est

Figure 4.6 : Exemples de textures non-homogènes. À gauche, la texture présente deux motifs différents (brique et ciment). À droite, la texture présente un motif, une fleur, qui n'est pas réparti de manière homogène dans l'image.

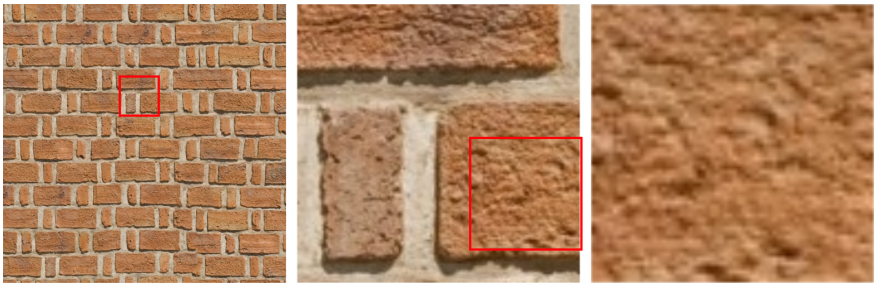


Figure 4.7 : De gauche à droite : texture d'origine, zoom ×8, zoom ×16. Sur la texture d'origine, le motif de brique est petit face à l'échelle d'observation, et sa répétition donne l'aspect homogène à l'image. Sur la texture du milieu, le motif est grand face à l'échelle d'observation, et ne se répète plus. L'image a un aspect hétérogène, avec deux motifs : brique et ciment. Sur la dernière texture, un seul motif de brique est visible, réparti de manière homogène.

possible de répéter le cadre dans l'espace, conserver la structure globale (un seul cadre) ou encore retirer le cadre pour conserver uniquement la texture de bois.

Il est tout de même possible d'obtenir des résultats satisfaisants avec certaines méthodes comme *Texture Optimization* [KEBK05] (voir figure 4.9), avec des défauts visibles (mélange local des motifs par exemple). Une manière de lever l'ambiguïté est de demander à l'utilisateur de la méthode de synthèse de fournir une carte de contrôle qui permettra de guider l'organisation spatiale des motifs lors de la synthèse, ce qui requiert une intervention manuelle. Lockerman *et al.* proposent une méthode pour extraire automatiquement une carte de contrôle représentant la répartition des motifs d'une texture d'entrée [LSA+16], puis une carte de sortie est tracée manuellement en utilisant les labels définis par la carte d'entrée.

Il est possible d'étudier le cas des glomérules, qui nous intéresse ici, du point de vue de l'informatique graphique. Les tissus sont composés de motifs qui se répètent entre et au sein des images, ce qui permet de parler de texture. Nous sommes dans le cas d'une texture non-homogène : le glomérule en lui-même a un aspect différent du tissu environnant. De plus,

FIGURE 4.8 : Exemple de texture non-homogène. Tiré de *Texture Stationarization* [MJH+17].

les deux régions peuvent être classées comme stochastique structuré. Nous sommes donc en présence d'un cas difficile à traiter par les méthodes de synthèse de texture classiques.

4.2 Les algorithmes de synthèse

4.2.1 Méthodes par ré-arrangement de patchs (non paramétriques)

Comme leur nom l'indique, les méthodes par ré-arrangement de patchs utilisent les pixels issus de l'exemple pour créer une nouvelle texture, en copiant des simples pixels [EL99] ou des patchs entiers [EF01] (*quilting*). Ce type de méthode ne construit pas de modèle statistique de l'exemple ; il n'y a donc pas d'étape d'analyse. Les textures sont ici traitées comme des champ de Markov : la valeur d'un pixel dépend uniquement de ses voisins. Pour chaque pixel de l'image résultat, l'algorithme sélectionne aléatoirement dans l'exemple un voisinage parmi les k voisinages les plus proches du pixel considéré. Ce dernier reçoit la valeur du pixel central du voisinage retenu [EL99], ou le voisinage complet est copié [EF01]. Dans le cas où la synthèse est effectuée par patch, il est nécessaire d'ajouter une étape pour corriger les erreurs au niveau des bords. Deux paramètres permettent de contrôler la synthèse : la taille des patchs P et le seuil de conservation des patchs ϵ. Si P est trop grand, l'algorithme risque de copier tels quels des morceaux de l'entrée, et si P est trop petit, la sortie peut perdre des structures présentes en entrée ou ne pas respecter certaines périodicités. Un ϵ trop grand entraîne la génération d'un contenu totalement déstructuré (*growing garbage* dans l'article).

Kwatra *et al.* [KEBK05] effectuent une synthèse par pixel et par patch. Leur algorithme synthétise plusieurs pixels à la fois en minimisant une fonction d'énergie quadratique, grâce à un algorithme du type Expectation-Maximisation (EM) : une étape optimise l'ensemble des voisinages par rapport à la valeur du pixel courant, l'autre étape optimise la valeur du pixel par rapport au voisinage courant.

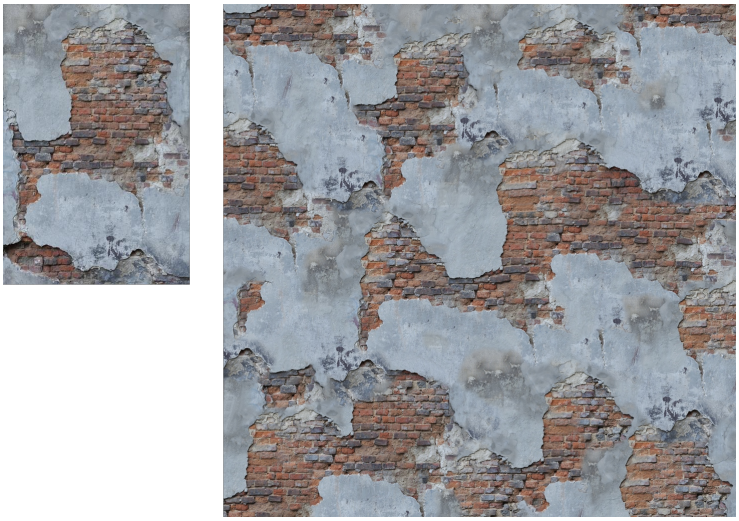


Figure 4.9 : Exemple de synthèse de texture non-homogène par la méthode *Texture Optimization* [KEBK05].

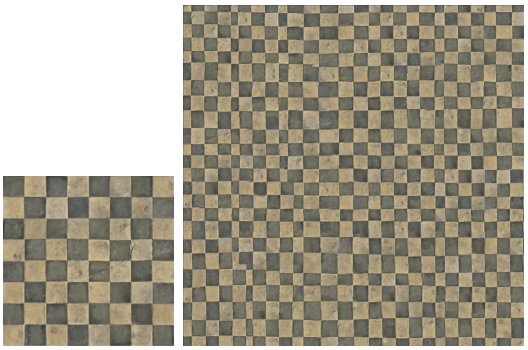


Figure 4.10 : À gauche : exemple en entrée. À droite : synthèse réalisée par la méthode de *quilting* [EF01]. La périodicité de l'exemple n'est pas respectée.

Les méthodes par ré-arrangement de patchs permettent d'obtenir des textures très réalistes puisque le contenu est copié de l'exemple, et sont capables de reproduire des structures locales, jusqu'à une certaine taille de motif. La structuration globale n'est à l'inverse pas toujours respectée, notamment les périodicités, comme montré sur la figure 4.10.

4.2.2 Méthodes statistiques (paramétriques)

Les méthodes statistiques, ou paramétriques, tentent de construire un modèle statistique de l'exemple, à partir duquel il sera possible de générer de nouvelles textures respectant ces statistiques. En général, la sortie est initialisée comme une image de bruit et est optimisée suivant une certaine fonction de coût. L'enjeu est donc de déterminer l'ensemble minimal de statistiques représentant au mieux chaque texture. Les différentes méthodes présentées

dans cette section diffèrent par les statistiques utilisées et/ou la méthode d'optimisation utilisée pour générer la nouvelle texture.

4.2.2.1 La pyramide de Heeger et Bergen

Heeger *et al.* [HB95] proposent de caractériser une texture par ses statistiques de premier ordre et ses réponses à un ensemble de filtres multi-échelles et multi-orientations organisé dans une pyramide orientable. Un histogramme des niveaux de gris est calculé pour chaque réponse, et les valeurs des pixels de la texture de sortie sont modifiées de manière itérative pour correspondre à ces histogrammes. La méthode est caractérisée par trois paramètres : le nombre d'itérations, le nombre d'échelles et le nombre d'orientations de la pyramide. Le nombre d'échelles est particulièrement important, afin de pouvoir capturer les motifs à différents niveaux. La pyramide de Heeger et Bergen est très efficace pour modéliser et synthétiser des textures stochastiques [RcDDM17], mais échoue à traiter des textures plus structurées (périodiques, mosaïques aléatoires, ou textures avec plus d'une orientation dominante). Les statistiques de premier ordre ne peuvent pas capturer de relations inter-pixels et ne sont donc pas suffisantes pour modéliser les régularités d'une image [PCR11].

4.2.2.2 L'algorithme de Portilla et Simoncelli

Portilla *et al.* [PS00] étendent l'algorithme de Heeger et Bergen en utilisant des statistiques d'ordre supérieur à 1, toujours sur la base d'un décomposition de la texture d'entrée sur une pyramide orientable. La méthode calcule un plus grand nombre de statistiques, parmi lesquelles l'auto-corrélation et la corrélation croisée, le tout inter et intra-échelles, ainsi que les moments d'ordre un à quatre. À partir d'une image de bruit blanc, l'algorithme consiste en la répétition itérative des étapes suivantes :

1. Décomposition sur la pyramide. Chaque sous-image est modifiée pour faire correspondre les statistiques correspondantes ;
2. Reconstruction de l'image résultat à partir de la pyramide. Elle est ensuite modifiée pour faire correspondre les statistiques d'ordre 1.

Les résultats obtenus sont meilleurs que pour la version précédente de l'algorithme, mais ce dernier échoue toujours sur les textures présentant des motifs structurés, comme montré sur la figure 4.11.

4.2.2.3 Méthodes par randomisation de phase

Les méthodes par randomisation de phase (*Random Phase Noise*, RPN) [GGM11; GSV+14] modifient la phase de la transformée de Fourier de la texture d'entrée. Ce type de méthode

Figure 4.11 : À gauche : exemple en entrée. À droite : synthèse réalisée par la méthode de Portilla et Simoncelli [PS00]. La structure n'est pas respectée : les pierres ne sont pas bien formées.

est très rapide puisqu'il ne nécessite que le calcul de deux transformées de Fourier rapides (*FFT, Fast Fourier Transform*) et fonctionne particulièrement bien pour la synthèse de textures stochastiques non-structurées. De plus, les modifications de phases locales introduites par Gilet *et al.* [GSV+14] permettent de reproduire certains motifs structurés, notamment les motifs périodiques.

4.2.3 Méthodes par réseaux génératifs adversaires

Bergmann *et al.* se sont intéressés à l'utilisation des GANs pour la génération de textures et ont proposé les *Spatial GANs* [JBV16] (SGANs) puis les *Periodic Spatial GANs* [BJV17] (PSGANs). L'architecture utilisée (voir figure 4.12) pour les réseaux est sensiblement la même que celle des DCGANs, sauf pour l'entrée du réseau générateur et la sortie du réseau discriminateur. G prend ici un tenseur Z de taille $l \times m \times d$ à la place d'un vecteur, l et m étant les dimensions spatiales et d la dimension de canal. D va lui donner en sortie une matrice de taille $l \times m$ dont chaque élément est la probabilité que la partie de l'image correspondante soit issue des données réelles ou des données générées. Ces deux modifications permettent au modèle de se focaliser sur les propriétés locales des images.

Les résultats de synthèse sont visuellement satisfaisants, notamment sur des textures stochastiques structurées, même si la méthode peine à modéliser et reproduire des structures globales. Le contrôle du tenseur d'entrée tel qu'introduit par les PSGANs permet tout de même de générer des textures plus régulières et notamment périodiques, comme montré sur la figure 4.13.

Les SGANs ont la capacité d'interpoler plusieurs textures provenant d'images différentes pour en créer de nouvelles. De plus, par une modification sur le tenseur d'entrée, il est possible de générer des tuiles sans couture (*seamless texture tiles*), qui pourront servir à paver une surface sans qu'une discontinuité ou une répétition soit observable.

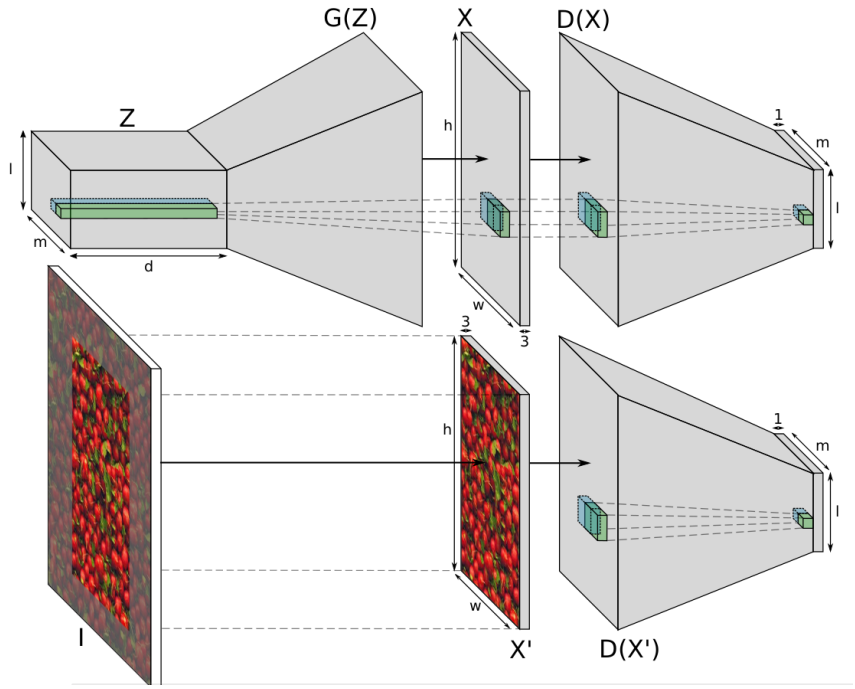


Figure 4.12 : Architecture utilisée des SGANs [JBV16]. Un tenseur aléatoire de taille $l \times m \times d$ est échantillonné puis projeté sur une image $l \times m \times 3$ par le générateur G. Le discriminateur D reçoit une image qu'il projette sur une matrice de probabilités de taille $l \times m$. Les SGANs peuvent être considérés comme un déploiement de GANs, car chaque vecteur de Z (en vert ou en bleu sur la figure), subit le même type de transformation que dans un GAN classique.

4.2.4 Conclusion intermédiaire

Les méthodes vues jusqu'ici ne sont pas adaptées à la synthèse de glomérules, car en général elles ne permettent pas la synthèse de textures non-homogènes, comme illustré par la figure 4.14. En particulier, les méthodes par GANs sont plus orientées vers la synthèse de motifs périodiques. Nous verrons des méthodes par GANs plus adaptées à notre problématique dans le prochain chapitre et nous concentrons ici sur des méthodes sans entraînement. La prochaine section est consacrée à une famille de méthodes utilisant des réseaux de neurones pré-entraînés. Nous aurions pu l'inclure dans le groupe des méthodes paramétriques, mais nous avons choisi de la traiter à part au vu de son importance.

4.3 Transfert de style par réseaux de neurones

Nous nous intéressons dans cette section à une famille de méthodes en particulier, dont l'origine se trouve à l'intersection des méthodes paramétriques vues précédemment et des

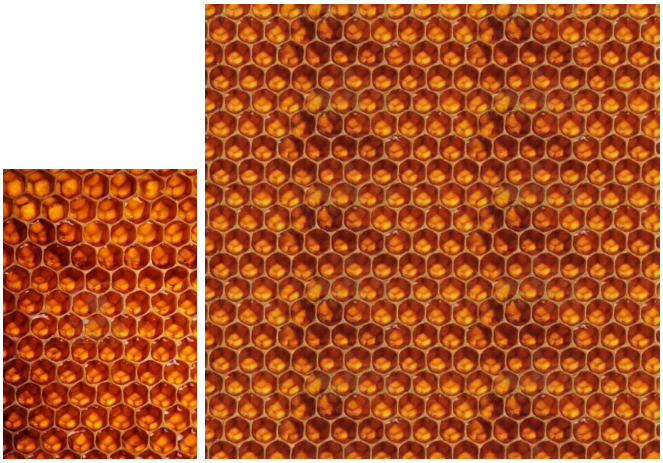


Figure 4.13 : De gauche à droite : l'exemple issu de la catégorie *honeycombed* de DTD, et une synthèse réalisée par les PSGANs. Ces derniers ont parfaitement reproduit la structure périodique de l'exemple.

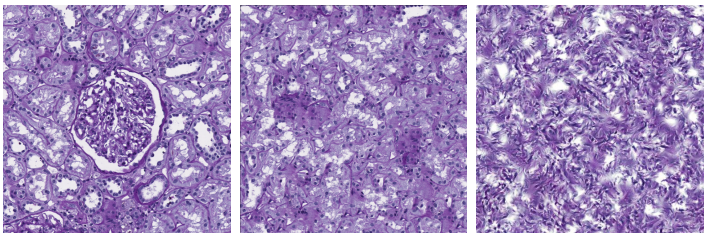


Figure 4.14 : De gauche à droite : exemple, méthode par *quilting*, méthode de Portilla et Simoncelli.

méthodes par réseaux de neurones. L'objectif de ces méthodes est de tirer parti de la puissance de représentation de réseaux pré-entraînés sur des images générales [YCBL14].

4.3.1 Synthèse de texture par réseaux de neurones

La méthode de synthèse de texture par réseaux de neurones de Gatys *et al.* [GEB15], connue également sous le nom de Transfert de Style par réseaux de neurones, et que nous appelons parfois simplement « méthode de Gatys », est une extension des méthodes statistiques, où les statistiques utilisées sont les filtres appris par un réseau de neurones pré-entraîné pour de la classification d'images générales, VGG-19 [SZ15]. Les statistiques utilisées sont donc ici potentiellement plus nombreuses et plus complexes, mais inconnues. Le réseau VGG-19 a été entraîné sur la base de données ImageNet et est disponible publiquement. La méthode de Gatys se base sur le calcul des matrices de Gram des cartes d'activation de VGG-19. En notant l la couche du réseau considérée et i et j les indices de deux cartes d'activation

F, la matrice de Gram G^l_{ij} est le produit scalaire entre les deux cartes d'activations :

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{kj} \tag{4.1}$$

À une constante près, les matrices de Gram représentent la corrélation entre les cartes d'activation. Pour générer une nouvelle image, la méthode compare les matrices de Gram issues de l'exemple et de la nouvelle image, initialisée comme une image de bruit, et minimise la fonction de coût ainsi définie. Notons G la matrice de Gram de la texture d'entrée et O la matrice de Gram de la texture de sortie. Nous pouvons écrire la contribution de chaque couche l de VGG-19 :

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (G^l_{ij} - O^l_{ij})^2, \tag{4.2}$$

où N_l est le nombre de cartes d'activations de la couche l et M_l le nombre de pixels de ces cartes d'activations. La fonction de coût totale E_{Grm} à minimiser est la somme de ces contributions :

$$E_{Grm} = \sum_l w_l E_l, \tag{4.3}$$

où w_l est le poids attribué à la couche l. Pour chaque couche, la dérivée de E_l est donnée par la formule suivante :

$$\frac{\partial E_l}{\partial F^l_{ij}} = \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - O^l))_{ij} \tag{4.4}$$

La rétro-propagation de cette valeur dans le réseau permet d'obtenir les gradients de la fonction E. Une procédure d'optimisation de type quasi Newton, L-BFGS-B [LN89], permet de mettre à jour les pixels de l'image résultat. La méthode complète est résumée par le schéma de la figure 4.15. En pratique, il est possible d'utiliser une seule couche par niveau de résolution de VGG-19 et d'obtenir des résultats équivalents à l'utilisation du réseau complet, ce qui permet de réduire le nombre de paramètres (852 000 à 177 000), et donc le temps de calcul. Des méthodes *ad hoc* permettent encore de réduire ce nombre, mais elles ne seront pas étudiées ici. Dans tous les exemples utilisant la méthode de Gatys, et ce jusqu'à mention contraire, nous utilisons les couches précédant les couches de *pooling*.

Comme montré dans l'article, les auteurs obtiennent des résultats meilleurs qu'avec les méthodes paramétriques classiques. La méthode est en particulier efficace sur les textures stochastiques (structurées ou non), mais échoue sur les textures régulières (figure 4.16) : sans information supplémentaire, elle ne parvient pas à reproduire des motifs périodiques. Il est également possible d'observer des instabilités dans la synthèse, avec notamment des variations de contraste et de luminosité non souhaitées, se traduisant souvent par des parties floues ou grisâtres dans les images produites. Les extensions de la méthode de Gatys permettent de palier ces défauts, par l'ajout de termes de régularisation.

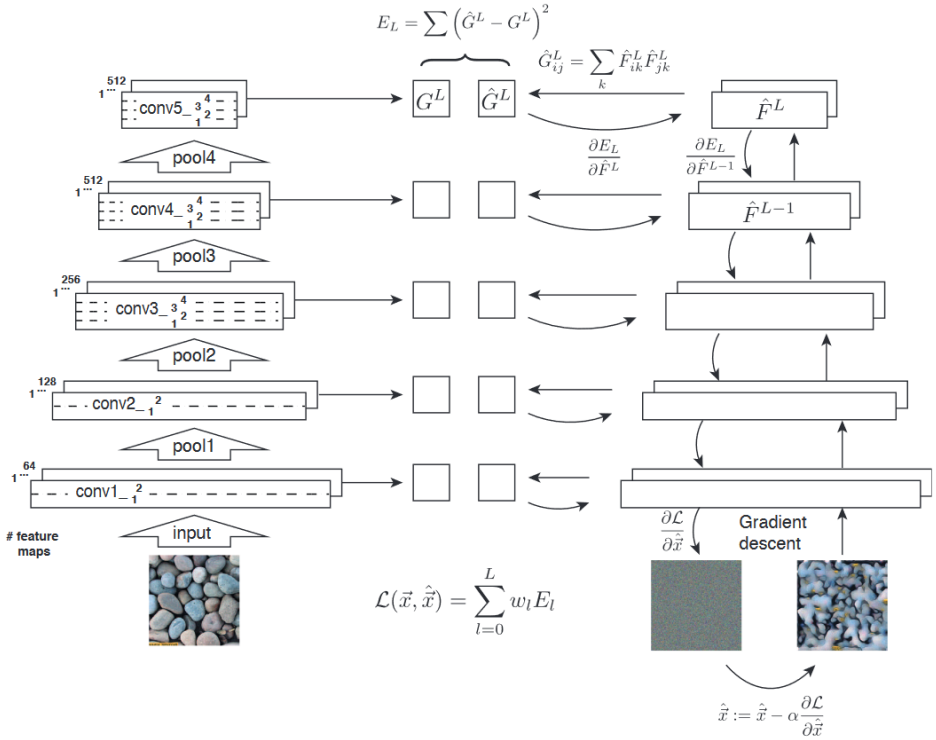


Figure 4.15 : Méthode de synthèse de texture par réseau de neurones de Gatys *et al.* [GEB15]. La partie gauche correspond à l'analyse, la partie droite à la synthèse. Les deux images (entrée et sortie) sont passées à travers VGG-19, où leurs matrices de Gram sont calculées à différentes couches. La rétro-propagation de l'erreur globale entre ces matrices permet de modifier les pixels de l'image en sortie et d'obtenir ainsi une nouvelle texture.

4.3.2 Extensions de la méthode de Gatys

4.3.2.1 *Deep Correlations*

La méthode *Deep Correlations* [SCO17] permet de synthétiser des images régulières et d'améliorer la qualité visuelle des résultats, par l'ajout de trois termes à la fonction de coût :

- E_{DCor} : même principe que les matrices de Gram, sauf que les corrélations sont cette fois des auto-corrélations. C'est ce terme qui permet de préserver les régularités de l'exemple.

- E_{Div} : terme de diversité. Il intervient pour pénaliser une synthèse trop proche de l'exemple.

- E_{Smooth} : terme de préservation des contours. Permet d'obtenir des textures moins floues.

La fonction de coût à minimiser devient alors :

$$E = E_{Grm} + \alpha E_{DCor} + \eta E_{Div} + \gamma E_{Smooth} \tag{4.5}$$

La combinaison des deux termes E_{DCor} et E_{Grm} permet une amélioration visuelle de la synthèse sur tout type de texture. En particulier, les auto-corrélations permettent la synthèse de motifs réguliers, comme montré sur la figure 4.16. Toutefois, le calcul des auto-corrélations allonge significativement le temps de calcul (d'un facteur cinq dans certains cas), et la méthode n'est pas toujours fiable, une modification de la taille de l'image d'entrée pouvant entraîner un échec complet de la synthèse d'après nos expériences.

FIGURE 4.16 : De gauche à droite : l'exemple, méthode de Gatys, *Deep Correlations*. La méthode de Gatys perd complètement la structure régulière, alors que *Deep Correlations* la reproduit parfaitement.

4.3.2.2 Utilisation d'histogrammes

Wilmot *et al.* [WRB17] proposent de compléter la fonction de coût E_{Grm} par une erreur sur les histogrammes de chaque caractéristique des cartes d'activations. L'objectif est de corriger l'erreur sur les moyennes et les variances, puisque deux cartes d'activation avec la même matrice de Gram peuvent avoir des moments statistiques très différents, ce qui provoque les instabilités dans la synthèse. La fonction à minimiser devient :

$$E = E_{Grm} + E_{histrogram} \tag{4.6}$$

4.3.2.3 Approche multi-échelles

Le champ récepteur des couches d'un réseau de neurones tel que VGG-19 reste limité, même pour les couches les plus profondes, ce qui ne permet pas à la méthode de Gatys de capturer les relations potentielles entre des pixels éloignés. Snelgrove *et al.* [Sne17] proposent alors une approche multi-échelle, qui consiste à calculer la fonction de coût entre les matrices de Gram sur des versions sous-échantillonnées et filtrées des images d'origine, sur plusieurs niveaux (pyramide gaussienne). Cette approche permet de synthétiser des images de haute résolution, de corriger les instabilités et de reproduire les motifs pertinents, comme le montre la figure 4.17.

Figure 4.17 : À gauche : exemple en entrée. À droite : synthèse réalisée avec une pyramide gaussienne.

4.3.2.4 *Texture Networks*

Ulyanov *et al.* [ULVL16] reprennent la fonction de coût sur les matrices de Gram et l'utilisent pour entraîner un réseau générateur. Il est alors possible, après convergence, de produire des textures en une seule passe, de qualité comparable aux images produites par la procédure d'optimisation. Les *Texture Networks*, de part leur rapidité d'exécution, permettent des applications en temps réel.

4.3.2.5 GANs et méthode de Gatys

Zhou *et al.* [ZZB+18] s'intéressent à la synthèse de textures non-stationnaires (ou non-homogènes) par une méthode d'extension adversaire (*Adversarial Expansion*), combinant la méthode de Gatys et une approche par GANs. L'extension adversaire est conçue pour doubler les dimensions de l'exemple d'entrée, en inférant le contenu manquant. La méthode est résumée à la figure 4.18. Les résultats de cette méthode sont limités dans leur diversité par le nombre de patchs qu'il est possible d'extraire de l'image d'origine, le réseau générateur étant ici complètement déterministe.

4.3.2.6 *Sliced Wasserstein Loss*

Heitz *et al.* [HVCB21] proposent l'utilisation d'une *Sliced Wasserstein Loss* [BRPP14], en replacement des matrices de Gram. Cette nouvelle manière de comparer des textures permet de capturer l'ensemble des statistiques des exemples, contrairement aux matrices de Gram qui ne capturaient que les statistiques de premier ordre. Les auteurs ont montré que leur méthode obtient de meilleurs résultats que toutes les méthodes basées sur les matrices de Gram sans ajout de terme d'erreur supplémentaire, peu importe l'application (synthèse de texture, transfert de style, etc), et donc sans besoin de régler manuellement la balance entre les différents termes.

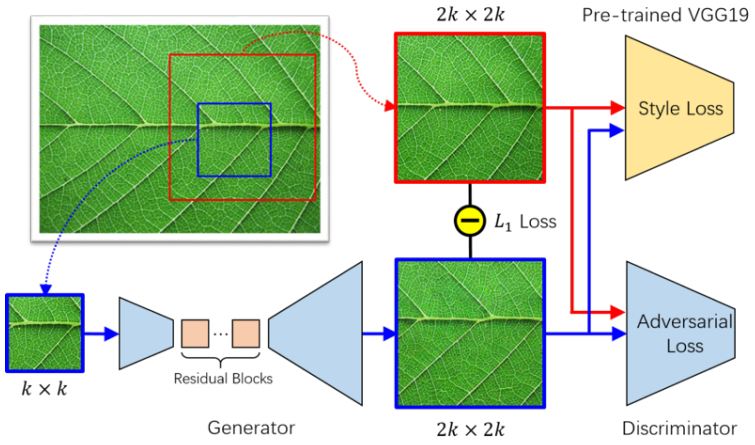


FIGURE 4.18 : (*Non-Stationary Texture Synthesis by Adversarial Expansion*) Architecture de la méthode de [ZZB+18]. Le générateur étend un patch de taille $k \times k$ issu de l'exemple en une image de taille $2k \times 2k$. Le discriminateur reçoit un patch de taille $2k \times 2k$ issu de l'exemple ou l'image générée. La *Style Loss* de Gatys est calculée à partir de ce patch et de l'image générée.

4.3.3 Guidage spatial de la synthèse

4.3.3.1 Transfert de style guidé

Gatys *et al.* [GEB+17] proposent de contrôler spatialement la synthèse de texture grâce à des cartes de guidage. L'objectif est de transférer plusieurs styles (représentés par la texture) sur différentes parties d'une image de contenu [GEB16] (utilisée comme initialisation), afin d'éviter notamment des mélanges de textures qui n'auraient pas de sens. Le guidage est effectué à l'aide de R cartes de guidage spatial binaires T^r, une pour chaque région de l'image de contenu à texturer. Ces cartes de guidage sont multipliées avec les cartes d'activations, afin d'obtenir R cartes d'activations guidées :

$$F_l^r = T_l^r . F_l \tag{4.7}$$

Les R nouvelles matrices de Gram sont calculées de la manière habituelle :

$$G_l^r = F_l^{rT} F_l^r \tag{4.8}$$

Et la fonction de coût devient :

$$E = \sum_l \frac{1}{N_l} \sum_{r=1}^{R} \sum_{ij} (G_l^r - O_l^r)_{ij}^2 \tag{4.9}$$

La figure 4.19 montre la différence entre un transfert de style simple et un transfert de style guidé. Le guidage spatial permet de répartir les textures de manière cohérente et de ne

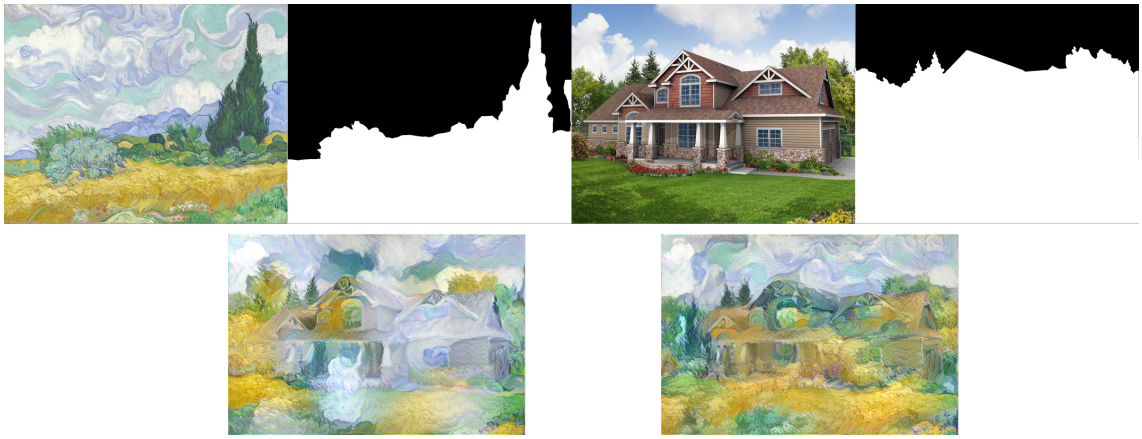


Figure 4.19 : Exemple de résultat obtenu par transfert de style guidé. Sur la première ligne, de gauche à droite : image de style, masque de l'image de style, image de contenu, masque de l'image de contenu. Sur la deuxième ligne : résultat obtenu sans masques (transfert de style classique), résultat du transfert de style guidé. Sans guidage, les textures de ciel et de sol se mélangent, pour un résultat peu naturel visuellement.

pas mélanger de l'herbe avec du ciel par exemple. Le guidage spatial est primordial pour notre application : il permet non seulement de synthétiser des textures non-homogènes, mais aussi de s'assurer que la méthode puisse générer des images de glomérule cohérentes avec un masque de vérité terrain, comme nous l'avions introduit dans le chapitre 1. Dans notre cas, nous n'avons pas besoin d'image de contenu, nous procédons donc simplement à une synthèse de texture guidée, avec initialisation par image de bruit.

4.3.3.2 Synthèse guidée multi-échelles

Nous avons trois éléments en entrée de la méthode de synthèse de texture guidée : une image de glomérule, son masque de segmentation associé, et un deuxième masque, dont le but est de contraindre la forme du glomérule synthétisé. La méthode de synthèse est la méthode de Gatys, avec guidage spatial. La figure 4.20 présente des résultats obtenus avec cette méthode. Le guidage permet bien de contraindre la synthèse est de transférer les zones texturées de manière cohérente.

Les images souffrent tout de même des instabilités récurrentes de la méthode de Gatys, avec des zones dont les statistiques de premier ordre ne sont pas cohérentes, apparaissant grises. De plus, les structures observables dans l'exemple ne sont pas reproduites. Parmi les extensions de la méthode de Gatys décrites à la section précédente, la synthèse multi-échelle est en particulier prometteuse, et nous retenons cette solution afin de proposer une synthèse de texture guidée multi-échelles. Dans notre cas, une autre manière de corriger certaines instabilités est d'utiliser le même masque de segmentation pour l'entrée et la sortie. Les résultats présentés à la figure 4.21 ont été obtenus avec la méthode de synthèse de texture guidée multi-échelles, en utilisant le même masque en entrée et en sortie, et

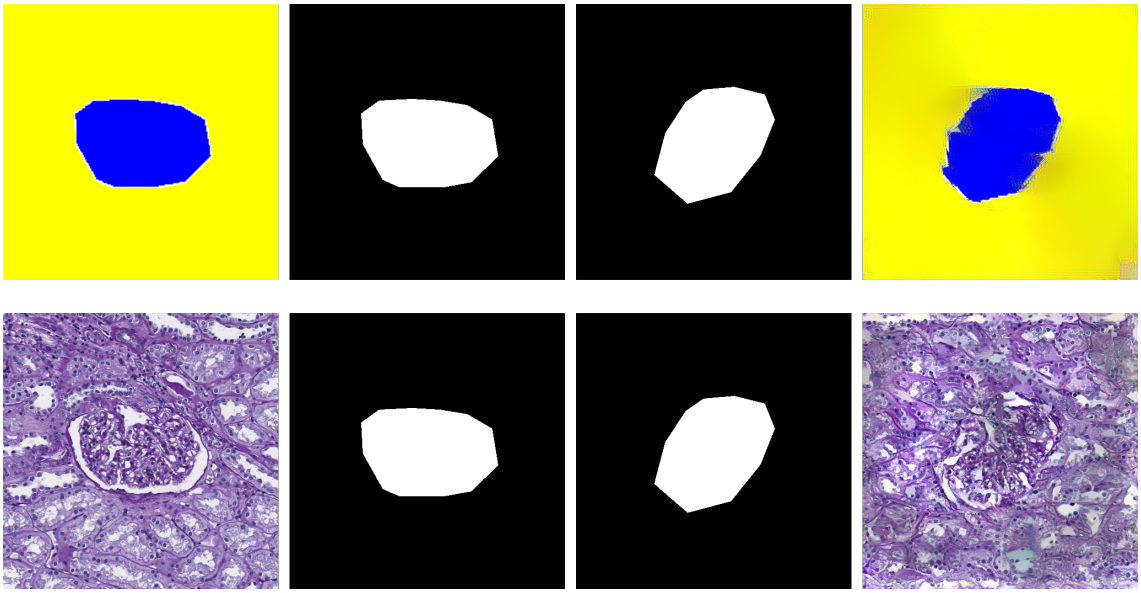


Figure 4.20 : Premiers résultats obtenus avec la méthode de synthèse de texture guidée de Gatys. Sur chaque ligne, de gauche à droite : exemple, masque lié à l'exemple, masque objectif, résultat. La première ligne présente le résultat sur une image synthétique pour contrôle, la deuxième ligne présente le résultat sur un glomérule. L'exemple synthétique permet de montrer clairement que la sortie a pris la forme attendue, ce qui est observable également pour le glomérule. L'un des défauts principaux de la méthode de Gatys est ici apparent : des zones sont mal synthétisées, ne respectant pas les statistiques de premier ordre.

en faisant varier le nombre de couches de VGG-19 et le nombre de niveaux de résolution (octaves) utilisés. Les résultats montrent que l'ajout de couches et de niveaux de résolution améliorent tout deux les performances. En pratique, il semble qu'utiliser plusieurs octaves permette d'obtenir des résultats plus stables et de meilleure qualité avec moins de couches, ce qui améliore la vitesse de synthèse. Pour toute la suite, nous utiliserons les couches 1, 4, 9, 14 et 19 (toutes les couches précédant une couche de *pooling*) de VGG-19 et 4 niveaux de résolutions.

L'utilisation de mêmes masques en entrée et sortie permet de reproduire fidèlement l'image de glomérule en entrée. Il n'est pas forcément évident de le voir sur les figures du manuscrit, mais les résultats produits sont légèrement variés : si l'apparence globale reste la même, les textures changent localement (voir la figure 4.22). Utiliser des masques différents peut augmenter la diversité des résultats, puisque l'organisation spatiale des motifs sera nécessairement différente, au prix d'instabilités visuelles dans les résultats, comme illustré par la figure 4.23. Chaque image présente des dégradations à des degrés divers.

Nous avons implémenté la méthode de synthèse de texture guidée multi-échelles en python3 et Tensorflow 1.15 sur la base des implémentations de Zirui Wang [Wan17] du transfert de style guidé et de Xavier Snelgrove [Sne] de la synthèse de texture multi-échelle.

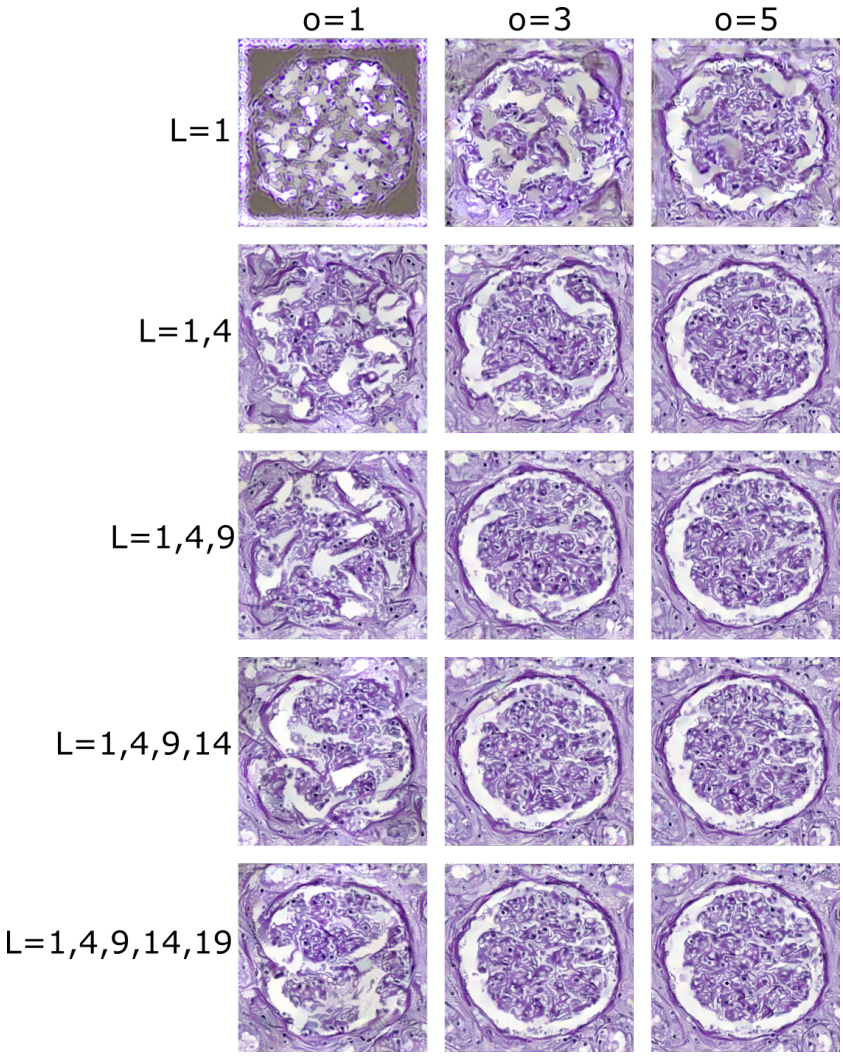


Figure 4.21 : Résultats de la méthode de synthèse de texture guidée multi-échelle. À chaque ligne, une couche est ajoutée dans la représentation, et à chaque colonne deux octaves sont ajoutés.

4.4 Entraînement d'un U-Net avec des images synthétiques

Nous avons désormais une méthode permettant de synthétiser des glomérules de bonne qualité et introduisant des variations de texture dans notre jeu de données d'origine. Nous pouvons donc utiliser cette méthode comme technique d'augmentation de données. Dans cette partie nous allons entraîner un U-Net avec différents jeux d'entrée, composés d'images réelles et/ou synthétiques, et tenter d'étudier l'impact en fonction des méthodes utilisées. Nous étudierons deux cas de figure :

1. les masques en entrée et en sortie sont les mêmes
2. les masques en sortis sont tirés au hasard

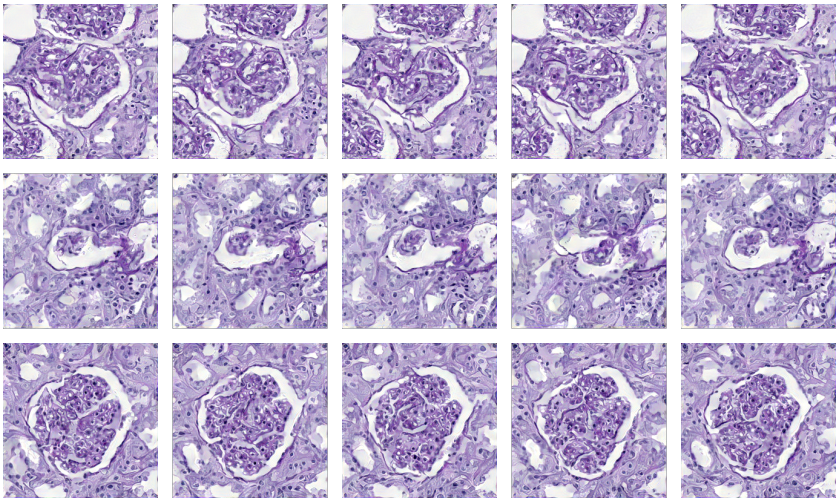


Figure 4.22 : Résultats de la méthode de synthèse de texture guidée multi-échelle lorsque deux masques identiques sont utilisés en entrée et en sortie. Chaque ligne correspond à un glomérule d'entrée différent. Pour chaque ligne, les 5 résultats sont obtenus avec une initialisation différente.

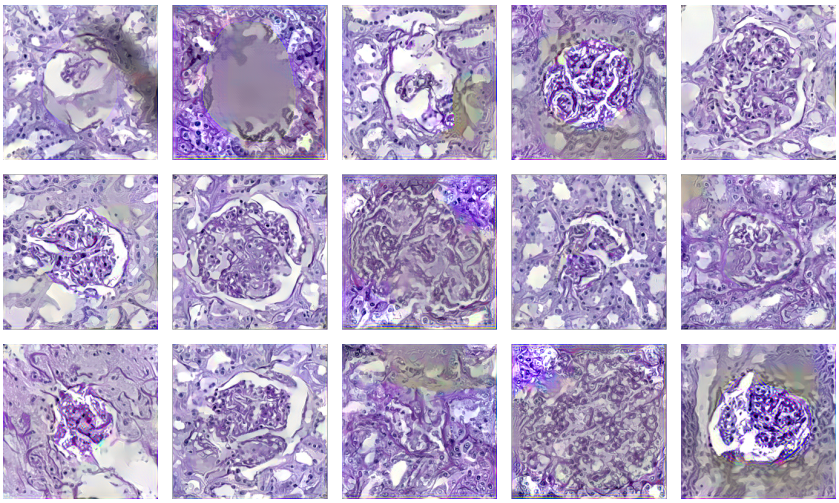


Figure 4.23 : Résultats de notre méthode lorsque deux masques différents sont utilisés en entrée et en sortie. Pour chaque image, le masque d'entrée est la vérité terrain du glomérule utilisé en exemple, et le masque de sortie est tiré au hasard parmi les masques existants dans la base de données.

Dans les deux cas, les images synthétiques seront générées avant l'entraînement du U-Net. En ce qui concerne les autres augmentations de données, elles comprendront des rotations et des symétries, ainsi que des déformations spatiales, grâce à la méthode GBD-3 du chapitre précédent avec $n = 3$ et $\sigma = 20$, et seront appliquées pendant l'entraînement avec une probabilité de 0,5. Par ailleurs, nous utilisons uniquement des images contenant un ou plusieurs glomérules, donc pas d'images contenant uniquement du tissu. La composition

N_r \ N_g	0	660	3300
0		$93{,}62 \pm 0{,}06$ (B)	$93{,}86 \pm 0{,}12$ (C)
660	$93{,}71 \pm 0{,}10$ (A)	$94{,}10 \pm 0{,}16$ (D)	$93{,}98 \pm 0{,}18$ (E)

Table 4.1 : Indice de Dice moyen en fonction du nombre d'images réelles et synthétiques dans le jeu d'entraînement (masques d'entrée et de sortie identiques).

des jeux de données avant augmentation est la suivante : 660 images d'entraînement, 400 images de validation et 431 images de test.

4.4.1 Masques en entrée et en sortie identiques

Pour chaque image du jeu d'entraînement, nous générons N images en utilisant en entrée l'image de glomérule et le masque, et en sortie le même masque. Nous nous limitons à $N = 1$ et $N = 5$, la génération des images étant chronophage. Nous obtenons ainsi deux jeux d'images synthétiques, un de 660 images et un autre de 3300 images. En notant N_r le nombre d'images réelles et N_g le nombre d'images générées, nous testerons toutes les combinaisons possibles pour $N_r = \{0{,}660\}$ et $N_g = \{0{,}660{,}3300\}$. Pour chaque combinaison, nous entraînons 5 fois un U-Net et moyennons les résultats, comme pour le chapitre précédent. Chaque entraînement dure ici 300 époques. Les indices de Dice obtenus sont présentés au tableau 4.1. Comme pour le précédent chapitre, les résultats complets (précision, recul, spécificité) sont disponibles en annexe. Les lettres entre parenthèses permettront de repérer les jeux de données dans l'analyse des résultats.

L'indice de Dice obtenu avec uniquement des images réelles (A) est de $93{,}71 \pm 0{,}10$. D'après le test de Wilcoxon (p-valeurs en annexe), tous les jeux de données donnent des résultats qui sont au moins équivalents aux résultats du jeu (A), y compris les jeux de données sans images réelles (B et C). Les jeux de données ayant des résultats significativement différents du jeu (A), c'est à dire les jeux (C, D et E) ont des résultats significativement supérieurs. Nous pouvons en tirer une première conclusion : les images synthétiques sont de suffisamment bonne qualité pour totalement remplacer les images réelles et obtenir des résultats comparables ou supérieurs. Ensuite, les meilleurs indices de Dice sont obtenus avec les jeux D et E, c'est à dire les jeux qui mélangent images réelles et images synthétiques. Les valeurs obtenues sont significativement plus élevées que celles des autres jeux de données. Les résultats des jeux D et E sont significativement équivalents, avec une moyenne légèrement supérieure pour le jeu D (images réelles et synthétiques à proportions égales) : 94,10. Mélanger images réelles et synthétiques permet donc d'obtenir les meilleurs résultats, avec un léger avantage pour une répartition égalitaire. Il est à noter que d'autres répartitions pourraient être intéressantes. Enfin, le gain de performance reste limité (+0,4% dans le meilleur des cas). Les variations introduites par les images synthétiques sont des variations de texture locales ne modifiant pas l'aspect global des images, d'où ce gain de performances modéré.

N_r \ N_g	0	660	3300
0		$90{,}32 \pm 0{,}49$ (B)	$90{,}98 \pm 0{,}30$ (C)
660	$93{,}71 \pm 0{,}10$ (A)	$93{,}70 \pm 0{,}10$ (D)	$93{,}98 \pm 0{,}10$ (E)

Table 4.2 : Indice de Dice moyen en fonction du nombre d'images réelles et synthétiques dans le jeu d'entraînement (masque de sortie aléatoire).

4.4.2 Masques choisis aléatoirement

Pour cette section, nous tirons au hasard N_g triplets (glomérule, masque d'entrée, masque de sortie) pour effectuer autant de synthèses. Le masque d'entrée est la segmentation vérité terrain du glomérule, tandis que le masque de sortie est tiré au hasard parmi l'ensemble des masques de segmentation. Comme précédemment, nous étudions les combinaisons possibles pour $N_r = \{0{,}660\}$ et $N_g = \{0, 660, 3300\}$. Les indices de Dice moyens obtenus sont présentés au tableau 4.2.

Les résultats montrent directement que les images générées par cette méthode sont de moins bonne qualité que dans la section précédente : pour les jeux de données B et C (entraînement sans images réelles), les indices de Dice sont 3 points en dessous de l'entraînement avec images réelles, avec respectivement 90,32% et 90,98%. Ces scores restent relativement élevés, ce qui montre que le U-Net peut tout de même apprendre des caractéristiques intéressantes, même si les images sont fortement dégradées. Par ailleurs, dans les cas D et E, nous obtenons des résultats qui sont égaux (D) ou supérieurs (E) à l'entraînement par images réelles. Nous retrouvons même le score obtenu dans la section précédente dans le cas E. Mélangées à des images réelles, des images synthétiques dégradées permettent donc au minimum de maintenir les performances. Le léger gain de performances pour le cas E (3300 images synthétiques) peut s'expliquer par le fait que plus de paires masques d'entrée/sortie « compatibles » (c'est-à-dire ayant une forme proche) seront tirées, permettant d'obtenir des images de bonne qualité. Nous aurons donc un jeu de données composé d'images réelles, d'images synthétiques de mauvaise qualité et d'images synthétiques de bonne qualité, dans une proportion aléatoire, ce qui permet d'introduire différents types de variations.

4.5 Conclusion et pistes sur la synthèse de texture

Dans ce chapitre, nous avons évalué l'impact de l'utilisation d'images synthétiques comme augmentation dans le cadre de la segmentation d'images histopathologiques à l'aide du modèle U-Net. Nous avons combiné des méthodes existantes dans l'état de l'art en synthèse de texture par l'exemple afin d'obtenir une méthode, la synthèse de texture guidée multi-échelle par réseaux de neurones, capable de synthétiser des images de glomérules de bonne qualité visuelle, et ne nécessitant pas d'entraînement préalable.

Pour déterminer la meilleure manière d'utiliser cette méthode, nous avons comparé les performances obtenues dans deux configurations : mêmes masques en entrée et en sortie, produisant des reproductions des images d'entrée avec des modifications de texture locales, puis masques différents, produisant des images de qualité variable mais qui peuvent introduire des variations de structure globales. Nous avons montré que l'ajout en proportions égales d'images synthétiques, produites avec les mêmes masques en entrée et en sortie, aux données réelles, permet d'obtenir la meilleure augmentation des performances de segmentation, avec un gain de 0,4% d'indice de Dice moyen. Le gain reste faible puisque l'augmentation ne consiste qu'en des variations locales de textures, ne modifiant pas l'apparence globale des images.

Une piste envisageable pour compléter les travaux autour de la synthèse de texture serait d'employer la méthode décrite par Heitz *et al.* [HVCB21], qui a l'avantage d'être plus générale que la méthode de Gatys que nous avons utilisée, et permet également une synthèse guidée. Une approche multi-échelle pourrait permettre d'obtenir des résultats satisfaisants. Néanmoins, comme nous l'avons vu dans ce chapitre, les performances des méthodes de synthèse de texture sont limitées pour notre application. Elles ne sont notamment pas capables d'inventer de nouvelles répartitions réalistes pour les motifs observables dans nos images de glomérules. Les contraintes imposées pour la synthèse permettent de reproduire fidèlement les structures existantes, mais par conséquent ne permettent que d'apporter de la variété localement. De plus, la lenteur de la synthèse peut être un frein au déploiement de ce type de méthodes.

Pour la suite de ce manuscrit, nous proposons donc de quitter le contexte de la synthèse de texture pour rejoindre le domaine plus général de la synthèse d'images. Nous utiliserons cette fois des méthodes par apprentissage, et notamment des réseaux génératifs adversaires.

Chapitre 5

Translation d'image-à-image

Dans tout ce chapitre, nous nous intéressons à la translation d'image-à-image (*Image-to-image translation*), une application des réseaux de neurones qui vise, comme son nom l'indique, à transformer une image en une autre. Il s'agit d'un sous domaine des GANs conditionnels, déjà évoqués au chapitre 2. Nous quittons ici le domaine de la synthèse de texture tel que vu par la communauté d'informatique graphique, même si nous verrons que les méthodes reviennent parfois à texturer des cartes sémantiques, comme nous avons pu le faire avec la synthèse de texture guidée. La figure 5.1 montre un ensemble d'applications possibles de la translation d'image-à-image.

Nous présenterons l'état de l'art des méthodes existantes et nous mettrons en évidence leurs limitations, notamment lorsque appliquées à notre jeu de données. Nous introduirons les divers solutions envisagées pour lever ces limitations et entraînerons un U-Net avec des images générées par une méthode de translation d'image-à-image. Les travaux de ce chapitre ont été présentés sous une forme légèrement différente lors de la conférence internationale IEEE-BHI-BSN le 29 septembre 2022 à Ioannina, en Grèce [AAWD22a].

5.1 Modèles de translation d'image-à-image

5.1.1 Pix2Pix

5.1.1.1 Le générateur : un U-Net

La méthode Pix2Pix [IZZE17] est une forme de GAN conditionnel [MO14], avec une image comme condition. Le générateur de Pix2Pix reçoit en entrée une image (et seulement une image, pas de bruit aléatoire ici), et renvoie en sortie une autre image. Son architecture

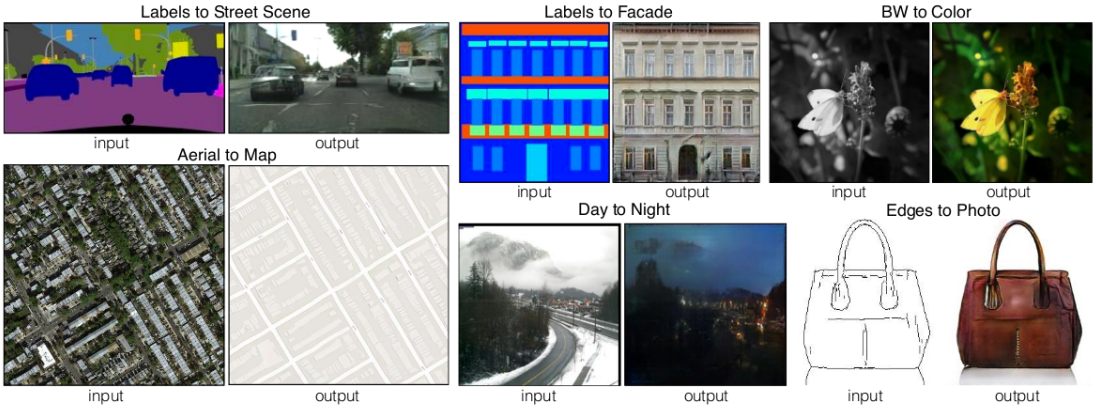


FIGURE 5.1 : Exemples d'applications de la translation d'image-à-image [IZZE17]. Il est possible de transformer une image de jour vers une image de nuit (et inversement), un croquis en objet texturé, une carte sémantique en image naturelle, une image en noir et blanc en image couleur etc.

n'est autre que celle d'un U-Net : encodeur-décodeur avec des connexions entre les deux branches.

5.1.1.2 Le discriminateur : patchGAN

Le discriminateur de Pix2Pix cherche à forcer la modélisation des hautes fréquences. Pour rappel, le discriminateur d'un GAN doit indiquer si l'image qu'il reçoit en entrée est réelle ou générée. Ici, il ne donne pas de réponse sur l'image complète mais sur des patches de taille N par N. Ces réponses sont ensuite moyennées pour donner la réponse globale du discriminateur. L'hypothèse sous-jacente est l'indépendance des pixels ne se trouvant pas dans le même patch (markoviannité). Les auteurs indiquent que leur patchGAN peut ainsi être compris comme une fonction de coût de texture ou de style, similaire à ce que nous avons étudié au chapitre 4.

5.1.1.3 Fonction de coût

Nous rappelons la formule modélisant l'apprentissage des GANs conditionnels, avec la même notation qu'au chapitre 2. ($\mathbf{x}$ est une donnée, $\mathbf{z}$ est un vecteur aléatoire, c une condition, G et D sont respectivement le générateur et le discriminateur) :

$$L_{cGAN}(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}|\mathbf{c}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c})))] \tag{5.1}$$

Dans notre cas, la condition est une image y. De plus, le générateur de Pix2Pix ne reçoit pas de bruit en entrée, seulement une image. La fonction de coût 2.3 devient alors (on

simplifiera les $\sim p_{data}$ dans toute la suite pour plus de lisibilité) :

$$L_{cGAN}(D,G) = \mathbb{E}_{\mathbf{x},\mathbf{y}(\mathbf{x},\mathbf{y})}[\log(D(\mathbf{x},\mathbf{y}))] + \mathbb{E}_{\mathbf{x},\mathbf{y}(\mathbf{x},\mathbf{y})}[\log(1 - D(G(\mathbf{y}),\mathbf{y}))] \tag{5.2}$$

À cette fonction de coût, les auteurs ajoutent une deuxième, qui est simplement une norme $L1$ entre l'image générée et l'image réelle (qui joue le rôle d'une vérité terrain) correspondant à la carte donnée en entrée :

$$L_{L1}(G) = \mathbb{E}_{\mathbf{x},\mathbf{y}(\mathbf{x},\mathbf{y})}[||\mathbf{x} - G(\mathbf{y})||_1] \tag{5.3}$$

Ce terme d'erreur permet de modéliser les basses fréquences contenues dans les images, le discriminateur s'occupant de modéliser les hautes fréquences. Finalement, la fonction de coût de Pix2Pix s'écrit de la manière suivante :

$$L_{pix2pix}(D,G) = L_{cGAN}(D,G) + L_{L1}(G) \tag{5.4}$$

5.1.2 Pix2PixHD

Les auteurs de Pix2PixHD [WLZ+18] reprennent les travaux précédents et cherchent à améliorer les résultats de synthèse, notamment en termes de résolution et de photo-réalisme. Les auteurs visent ainsi la synthèse d'images d'une taille pouvant aller jusqu'à 4096×2048, contre moins de 300×300 en général pour Pix2Pix. Pour ce faire, ils proposent de nouvelles architectures pour les deux réseaux antagonistes, ainsi qu'un nouveau terme supplémentaire pour la fonction de coût.

5.1.2.1 Générateur *coarse-to-fine*

Le générateur de Pix2PixHD, représenté à la figure 5.2 est composé de deux réseaux $G1$ et $G2$. $G1$ procède à la synthèse sémantique en tant que telle, suivant l'architecture de Johnson *et al.* [JAFF16]. Cette dernière utilise en particulier des blocs résiduels (*residual blocks*, permettant de sauter une couche puis d'ajouter la sortie d'une couche précédente [HZRS16]. Ce premier réseau travaille à une résolution de 1024×512. $G2$, dont l'architecture est semblable à $G1$, encapsule ce dernier et permet d'obtenir des images dont les tailles sont multipliées par 4. Les auteurs qualifient $G1$ de réseau « global », puisqu'il travaille sur toute la carte sémantique, et $G2$ de réseau « local », puisqu'il ajoute des détails au travail effectué par $G1$. À noter : les deux générateurs ne sont pas entraînés en même temps, mais bien successivement.

5.1.2.2 Discriminateur multi-échelle

Comme noté par les auteurs, l'augmentation des dimensions des images requiert des réseaux discriminateurs plus profonds, pour que le champ récepteur de ces derniers soit assez large

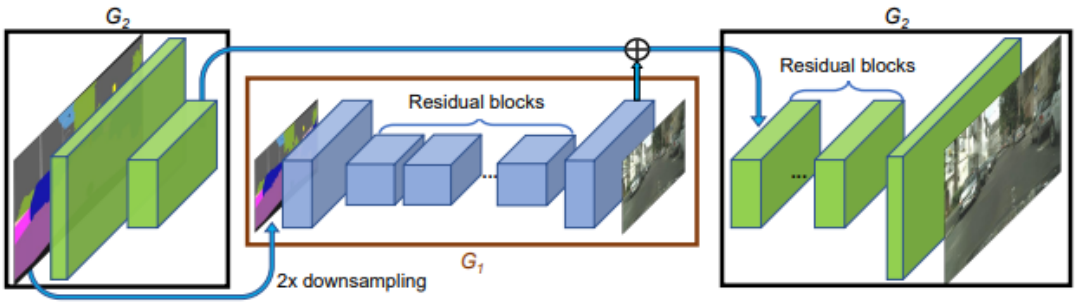


FIGURE 5.2 : Architecture du générateur de Pix2PixHD [WLZ+18]. Il est en réalité composé de deux générateurs entraînés l'un à la suite de l'autre et travaillant à des résolutions différentes.

et puisse capturer des informations pertinentes, ce qui peut se révéler coûteux et complexe à entraîner. Pour contourner ce problème, les auteurs proposent d'utiliser 3 discriminateurs identiques mais travaillant à des échelles différentes. Ils reçoivent chacun une version de plus en plus sous-échantillonnée des images à traiter, ce qui permet de faire varier les champs récepteurs sans avoir à entraîner de réseau trop profond. La fonction de coût résultante n'est que la somme des coûts des 3 discriminateurs :

$$L(D,G) = \sum_{k=1}^{3} L_{cGAN}(D_k,G) \tag{5.5}$$

5.1.2.3 *Feature matching*

Les auteurs introduisent une *feature matching loss*, autrement dit un terme d'erreur sur les cartes d'activations des discriminateurs, calculé entre une image générée et une image réelle. En notant L le nombre de couches dans un discriminant et N_i le nombre d'éléments dans la couche i, nous obtenons la fonction de coût suivante :

$$L_{FM}(D_k,G) = \mathbb{E}_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{L} \frac{1}{N_i} [||D_k^{(i)}(\mathbf{x},\mathbf{y}) - D_k^{(i)}(G(\mathbf{y},\mathbf{y}))||_1] \tag{5.6}$$

Cette fonction de coût force le générateur à générer des images réalistes à différentes échelles, en plus d'aider à stabiliser l'entraînement. La fonction de coût totale pour Pix2PixHD est la suivante :

$$L_{pix2pixHD}(D,G) = \sum_{k=1}^{3} L_{cGAN}(D_k,G) + \lambda \sum_{k=1}^{3} L_{FM}(D_k,G) \tag{5.7}$$

Remarquons la disparition du terme $L1$ utilisé dans Pix2Pix. λ est une valeur de pondération.

Figure 5.3 : Exemple de carte de frontières et impact sur les résultats [WLZ+18]. Première ligne : carte de labels à gauche et carte de frontières à droite. Deuxième ligne : résultat avec uniquement la carte de labels à gauche, résultat avec les deux cartes à droite.

5.1.2.4 Cartes de frontières

Les cartes de labels ne différencient pas les différents objets présents dans une image. Si ces objets sont collés ou se chevauchent, comme montré à la figure 5.3, le générateur n'est pas toujours capable de produire des images cohérentes avec des objets aux contours nets. Pour améliorer les résultats de leur méthode, les auteurs proposent d'ajouter une carte de frontières en entrée du générateur. Cette carte permet de séparer les différents objets, en particulier ici les voitures, ce qui permet d'obtenir des résultats plus nets au niveau des contours, tel que montré à la figure 5.3.

5.1.3 SPADE

5.1.3.1 *Spatially-Adaptive DEnormalization*

La normalisation de couches est une méthode standard pour accélérer et stabiliser l'entraînement des réseaux de neurones profonds. La plus utilisée est la normalisation par lot [IS15] (*Batch Normalization*), mais de nombreuses autres méthodes existent selon les besoins [SK16; UVL16; DSK16; HB17].

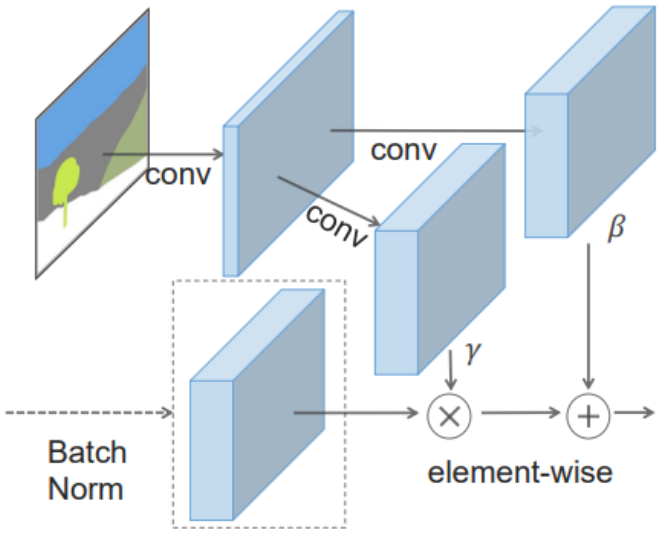


Figure 5.4 : Schéma de la couche SPADE [PLWZ19].

Park *et al.* [PLWZ19] proposent une nouvelle méthode de normalisation conditionnelle, appelée *Spatially-Adaptive DEnormalization* (SPADE, nom sous lequel nous désignerons le GAN entier pour la suite de ce manuscrit). L'objectif de cette nouvelle normalisation est la conservation de l'information sémantique dans les couches successives du générateur. Les auteurs ont en effet remarqué que la normalisation par lot a tendance à faire s'évanouir l'information sémantique, notamment dans les zones homogènes des cartes de label, ce qui conduit à la synthèse de zones grises en lieu et place de la texture attendue. La méthode peut être vue comme une extension des méthodes de normalisation conditionnelle précédentes [DSK16; HB17] et permet, grâce à une modification de l'architecture du générateur, de ne normaliser que les cartes d'activations et pas la carte de labels. La figure 5.4 schématise la couche : la carte de labels est d'abord convoluée de manière à produire deux tenseurs γ et β, qui sont respectivement multipliés et ajoutés à la carte d'activation (normalisée par lot) précédente.

5.1.3.2 Le générateur de SPADE

Comme évoqué dans la section précédente, le générateur de SPADE adopte une nouvelle architecture. Désormais, les cartes de labels entrent dans le réseau par les couches SPADE, à différents niveaux de résolution, comme le montre la figure 5.5. Ainsi, la partie encodeur des réseaux précédents est remplacée par des couches « SPADE ResBlk » qui utilisent à nouveau le bloc résiduel de ResNet [HZRS16]. Le réseau obtenu a une forme plus classique, pouvant prendre en entrée un vecteur de bruit. Pour ce qui est du reste (discriminant, fonction de coût), ce sont les mêmes que pour Pix2PixHD, à l'exception de l'utilisation d'une fonction charnière (*hinge loss*) pour entraîner le générateur.

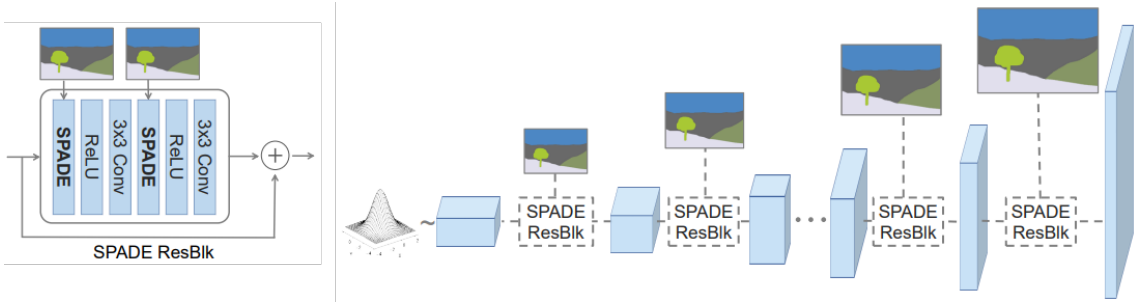


Figure 5.5 : Générateur de SPADE [PLWZ19]. La partie encodeur des réseaux précédents disparaît au profit des couches dites « SPADE ResBlk », utilisant encore une fois le bloc résiduel de ResNet [HZRS16].

5.1.3.3 Synthèse multi-modale

Outre la normalisation, l'élément qui différencie grandement SPADE de ses prédécesseurs est l'utilisation de bruit en entrée du réseau, ce qui permet une synthèse multi-modale [ZZP+17] : en variant le bruit, le résultat varie, contrairement à Pix2Pix et Pix2PixHD qui ne peuvent synthétiser qu'une sortie pour une carte de labels donnée. En utilisant un encodeur, il est possible d'extraire un vecteur de style d'une image, et d'utiliser ce vecteur en entrée de SPADE. Cet encodeur et le générateur forment alors un auto-encodeur variationnel (VAE [KW14]), que l'on peut entraîner en même temps que le GAN complet grâce à une divergence de Kullback-Leibler. La figure 5.6 montre un exemple de manipulations sémantiques et de styles obtenu grâce à cette combinaison de SPADE et d'un encodeur.

5.1.4 Modèles avec fonction de cohérence

Comme évoqué au chapitre 2, il existe des modèles particuliers de translation d'image-à-image qui permettent de passer d'un domaine A à un domaine B, puis du domaine B au domaine A en assurant la cohérence de l'opération. Ce type de méthode possède la particularité de ne pas avoir besoin de données par paires, c'est-à-dire image et carte de labels correspondante, pour apprendre, contrairement aux méthodes précédentes. Nous présentons dans cette section le modèle de référence, utilisé par Vasiljevic *et al.* [VFWL21b] pour effectuer des transferts de coloration.

Un *CycleGAN* [ZPIE17] est composé de deux GANs conditionnels entraînés conjointement, l'un à transformer les images du domaine X en une image du domaine Y et inversement, et liés par une fonction de cohérence. Notons G et F les deux générateurs. G effectue le transfert $X \to Y$, et F effectue le transfert $Y \to X$. Notons également D_X et D_Y les deux discriminants, travaillant chacun sur un des deux domaines. G et F ont la même architecture que le générateur de Pix2PixHD [WLZ+18], et les discriminants sont tout deux des patchGANs [IZZE17]. La fonction de cohérence vise à s'assurer que la com-

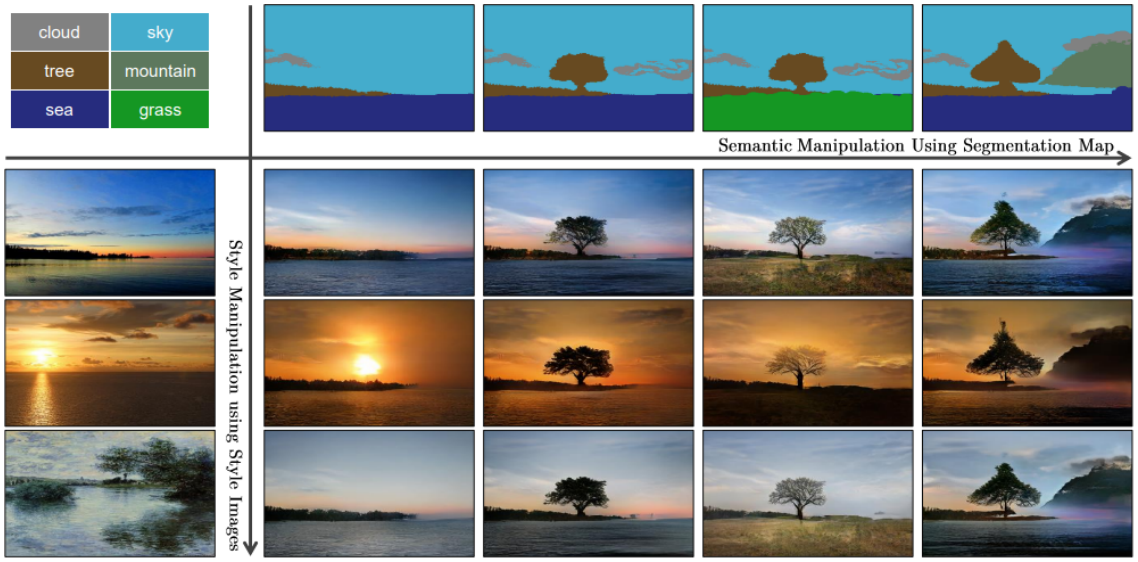


FIGURE 5.6 : Exemple d'utilisation de SPADE. Les cartes de labels permettent une manipulation sémantique de l'image de sortie, tandis que les images de style permettent de manipuler son aspect général.

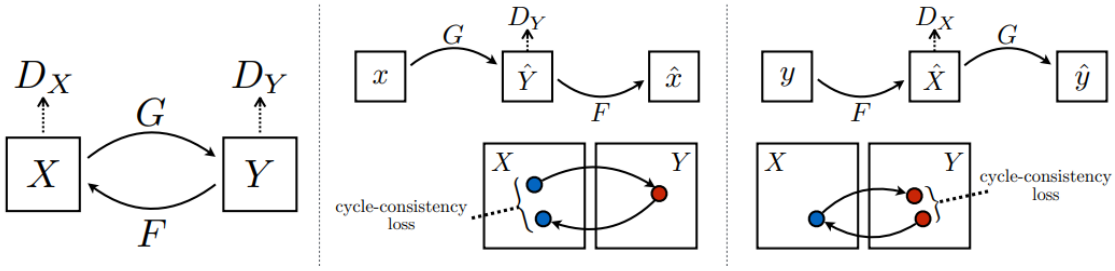


FIGURE 5.7 : Résumé de la méthode CycleGAN 5.7. Deux GANs conditionnels similaires à Pix2Pix et Pix2PixHD sont liés par une fonction de cohérence. Les deux générateurs appliqués successivement à une image doivent être aussi prêts que possible de l'identité.

position $F(G(\mathbf{x}))$ soit égale à $\mathbf{x}$ pour $\mathbf{x}$ une image du domaine X, et que $G(F(\mathbf{y}))$ soit égale à $\mathbf{y}$ pour $\mathbf{y}$ une image du domaine Y. La méthode est résumée à la figure 5.7.

La fonction de cohérence s'exprime comme une erreur de reconstruction entre l'image de départ ($\mathbf{x}$ ou $\mathbf{y}$) et le résultat de la composition des deux générateurs, et ce dans les deux sens ($X \to Y$ et $Y \to X$) :

$$L_{cyc}(G,F) = \mathbb{E}_{\mathbf{x}}[||F(G(\mathbf{x}) - \mathbf{x})||_1] + \mathbb{E}_{\mathbf{y}}[||G(F(\mathbf{y}) - \mathbf{y})||_1] \tag{5.8}$$

Ainsi la fonction objectif de CycleGAN peut simplement s'écrire :

$$L(G,F,D_X,D_Y) = L_{cGAN}(G,D_Y) + L_{cGAN}(G,D_X) + \lambda L_{cyc}(G,F) \tag{5.9}$$

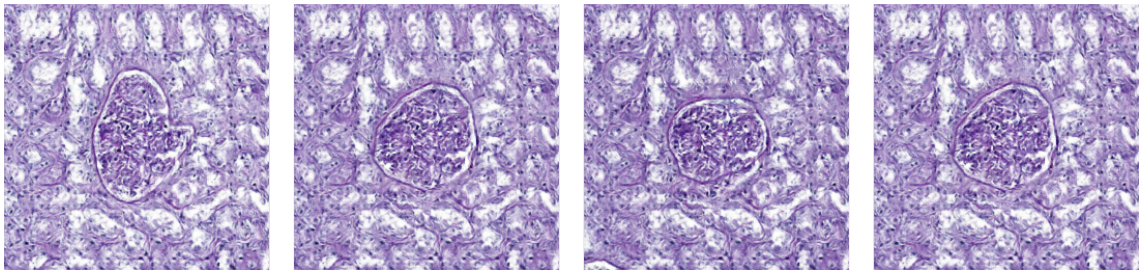


Figure 5.8 : Illustration de l'effondrement de mode. Toutes les images générées ont une texture identique malgré un masque différent en entrée du générateur.

5.2 Synthèse de glomérules et verrou scientifique

Nous avons implémenté un modèle Pix2Pix et un modèle SPADE avec augmentations adaptatives en python3 et Tensorflow 2, en utilisant les tutoriels proposés sur les sites de Tensorflow et Keras [pix; gau; ada] ainsi que l'implémentation de Kim Junho [Jun].

5.2.1 Premiers résultats : effondrement de mode

Nos premières expérimentations avec les modèles de translation d'image-à-image ont été réalisées avec Pix2Pix. Les images générées sont de bonne qualité. Cependant, un défaut inhérent aux GANs, et en particulier aux modèles de translation d'image-à-image, nous est très vite apparu : l'effondrement de mode (*mode collapse*). L'effondrement de mode a lieu lorsque le générateur du GAN, après convergence, produit des résultats peu divers : la distribution apprise ne couvre seulement que quelques modes de la distribution des données réelles. En pratique, le générateur ne synthétisera que quelques images différentes malgré des entrées différentes. Cela se manifeste dans notre cas par une texture de sortie identique avec n'importe quel masque en entrée, comme le montre la figure 5.8.

Ce problème est critique pour notre application : nous voulons utiliser les images générées pour entraîner un algorithme de segmentation. Notre modèle génératif doit donc être capable d'apprendre tous les modes de la distribution latente des données, ou au moins le plus possible. Le fait que les GANs soient connus pour être instables et produire des images peu diverses semble aller dans le sens d'un problème intrinsèque de ce type de méthode. Les modèles de translation d'image-à-image sont en particulier impactés. De plus, nos données sont elle-mêmes particulières : le niveau de détail est trop élevé pour les considérer comme de simples textures, ce qui oblige le générateur à tenter d'apprendre des structures et leur organisation spatiale, ce qui est une tâche difficile, comme nous l'avons vu au chapitre 4. De plus, notre jeu de données est de petite taille (quelques centaines), surtout comparé aux jeux de données utilisés dans les papiers portant sur les GANs tels que Cifar [Kri12] ou CelebsAHQ [KALL17], constitués de dizaines de milliers d'images.

5.2.2 La littérature autour de l'effondrement de mode

Dans cette section, nous présentons un ensemble de papiers traitant de l'effondrement de mode. Plusieurs leviers sont envisagés pour s'attaquer au problème : modification de l'architecture, nouvelles fonctions de coût, algorithmes d'apprentissage différents... Avant de commencer, il faut adresser un point évident : Pix2Pix est un modèle déterministe, dans le sens où l'entrée n'est pas un vecteur de bruit. L'ajout de bruit en entrée du générateur ne suffit toutefois pas à créer des résultats divers, car le modèle apprend à ignorer ce bruit, comme indiqué dans l'article original [IZZE17].

5.2.2.1 Utilisation de SPADE

Nous l'avons vu à la section précédente, l'architecture de SPADE permet une synthèse multi-modale, et notamment d'obtenir des images différentes à partir d'un même masque sémantique grâce à une manipulation du vecteur d'entrée. En théorie, l'utilisation de vecteurs aléatoires doit mener à la synthèse d'images différentes. Or, sur notre jeu de données, SPADE souffre du même problème que Pix2Pix. De fait, que ce soit en utilisant le même masque avec différents bruits d'entrée ou en utilisant différents masques, les images générées présentent des textures quasiment identiques.

5.2.2.2 *Unrolled GANs*

Nous rappelons que les GANs peuvent être exprimés comme un jeu à deux joueurs et à somme nulle, où le générateur et le discriminateur seraient en concurrence. L'idée derrière les *Unrolled GANs* (GANs déroulés) [MPPS16] est de donner des « coups d'avance » au générateur. Pour ce faire, à l'itération i, l'algorithme d'apprentissage effectue k étapes, c'est-à-dire k mises à jour par rétro-propagation du générateur et du discriminateur, puis utilise le dernier discriminateur, celui de l'étape $i+k$, pour entraîner le générateur i, qui voit donc k coups à l'avance. Le discriminateur i subit lui la procédure habituelle. L'objectif de l'opération est d'empêcher le générateur de tomber dans des minima locaux ou de sur-apprendre pour un discriminateur donné. Les auteurs ont montré expérimentalement le bien fondé de cette méthode sur des exemples simples (mélange de gaussiennes, voir figure 5.9) et des jeux de données classiques avec des images de petite taille (MNIST [LC10], Cifar [Kri12], de taille 32×32 ou moins).

À notre connaissance, les *Unrolled GANs* n'ont pas été appliqués à des images plus grandes ni dans le cas d'un modèle de translation d'image-à-image. Nous avons tenté de créer un « *Unrolled Pix2Pix* », sans succès. La descente de gradient ne converge pas.

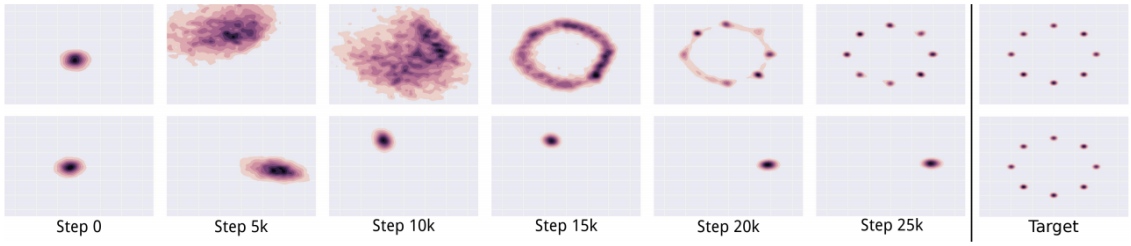


Figure 5.9 : *Unrolled GANs* [MPPS16] appliqués à un mélange de gaussiennes (première ligne). GAN classique (seconde ligne). Dans le premier cas, la distribution apprise par le générateur parvient à reproduire tous les modes du mélange de gaussiennes. Dans le second cas, le générateur alterne entre les différents modes, un seul à la fois.

5.2.2.3 *Wasserstein GANs*

La distance de Wasserstein (*Earth Mover's distance* ou *EM*) est définie comme le coût minimal du transport de masse pour transformer une mesure, éventuellement de probabilité si la somme des masses est normalisée à 1, u en une mesure v, autrement dit comme le coût du transport optimal [ACB17]. En reprenant les notations de l'article, nous avons P_r la distribution des données réelles, P_g la distribution des données générées, Π l'ensemble des transports γ possibles. Nous avons :

$$W(P_r,P_g) = \inf_{\gamma \in \Pi(P_r,P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||] \tag{5.10}$$

Les auteurs des Wasserstein GANs (WGANs) [ACB17] proposent d'utiliser cette distance comme nouvelle fonction de coût en lieu et place de la fonction classique, moyennant une astuce mathématique pour rendre l'expression 5.10 calculable. Les auteurs obtiennent la formule suivante, où f est une fonction 1-lipschitzienne :

$$W(P_r,P_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)] \tag{5.11}$$

Cette fonction est déterminée de la même façon que pour un GAN classique : elle est apprise par un réseau de neurones, que les auteurs nomment « critique ». Il vient ici remplacer le discriminateur, avec un rôle et une architecture analogue. La différence entre un discriminateur et un critique est que ce dernier renvoie un simple scalaire plutôt qu'une probabilité. Pour forcer la contrainte selon laquelle le critique doit être 1-lipschitzien, les poids du réseau sont maintenus dans un certain intervalle de valeurs contrôlé par un hyperparamètre c. Dans le cas d'un GAN classique, le gradient censé mettre à jour le générateur tend vers 0 si le discriminateur est trop performant et que le générateur ne l'est pas suffisamment. Dans ce cas le générateur n'apprend plus, il est donc nécessaire d'équilibrer soigneusement les performances des deux réseaux antagonistes, surtout que le discriminateur apprend en général plus rapidement, la tâche de classification étant plus facile que la

tâche de génération. Les WGANs permettent à l'inverse au générateur d'apprendre correctement même lorsque le critique est optimal, et le gradient est plus lisse. Cela leur permet de tirer parti du fait qu'un critique de qualité (voire optimal) fournit de bonnes informations pour améliorer le générateur. Les auteurs proposent alors de mettre à jour plusieurs fois de suite le critique, pour une seule mise à jour du générateur.

Expérimentalement, les auteurs ont montré que leur nouvelle fonction de coût a plus de sens que la précédente, puisque la diminution de la perte associée au générateur est corrélée à une meilleure qualité des résultats produits, contrairement aux GANs classiques. Ils montrent également une plus grande stabilité dans les apprentissages, et notamment une sensibilité moindre aux changements d'architecture des réseaux ou d'hyperparamètres. Enfin, ils prétendent n'avoir rencontré aucun effondrement de mode.

5.2.2.4 *Improved Wasserstein GANs*

Les *Improved Wasserstein GANs* [GAA+17] visent à combler certains défauts résiduels de la méthode précédente, dûs à la troncature des poids du critique. Le modèle est en particulier sensible au réglage de l'hyperparamètre c, et les auteurs ont montré que l'entraînement d'un WGAN reste instable dans certains cas. Pour ce faire, les auteurs proposent de forcer le caractère lipschitzien en pénalisant la norme du gradient issu du critique, afin que cette dernière soit proche de 1 pour les valeurs les plus grandes. Cette pénalisation est calculée à partir d'une interpolation d'une donnée générée et d'une donnée réelle. En notant $\mathbf{x}$ la donnée réelle, $\tilde{\mathbf{x}}$ la donnée générée et $\hat{\mathbf{x}}$ la donnée interpolée, nous avons : $\hat{\mathbf{x}} = t\tilde{\mathbf{x}} + (1-t)\mathbf{x}$, avec t tiré aléatoirement depuis la distribution uniforme sur [0,1]. Nous obtenons ainsi la nouvelle fonction de coût :

$$L = \mathbb{E}_{\tilde{\mathbf{x}}\sim P_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim P_r}[D(\mathbf{x})] + \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim P_{\hat{\mathbf{x}}}}[(||\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})||_2 - 1)^2] \tag{5.12}$$

où les deux premiers termes correspondent au coût original du critique et le dernier à la pénalité du gradient des *Improved Wasserstein GANs*. λ est fixé à 10.

L'utilisation de cette méthode ne permet pas de pallier l'effondrement de mode sur notre application et l'entraînement est parfois plus instable qu'avec la *Hinge Loss* utilisée pour SPADE.

5.2.2.5 *Mode Seeking GANs*

Mao *et al.* ont proposé les *Mode Seeking GANs* (GANs à recherche de mode), ou MSGANs [MLT+19], qui visent à augmenter la diversité de résultats générés par des GANs conditionnels. La méthode consiste en un terme de régularisation pénalisant le générateur si, pour deux entrées proches, il produit deux résultats proches :

$$L_{ms} = \max_G \frac{||G(c, \mathbf{z1}) - G(c, \mathbf{z2})||_1}{||\mathbf{z2} - \mathbf{z1}||_1} \tag{5.13}$$

où c est une condition (le masque d'entrée dans notre cas) et **z1** et **z2** des vecteurs de bruit.

Le terme d'erreur est ajouté au reste de la fonction de coût, ce qui nous permet de l'utiliser de manière transparente pour n'importe quel modèle :

$$L_{msgan} = L_{cgan} + \lambda_{ms} L_{ms} \tag{5.14}$$

Encore une fois, malgré le bien-fondé de la solution proposée et les résultats présentés dans l'article, les *Mode Seeking GANs* n'apportent pas de diversité dans les images générées avec notre jeu de données. Nos observations peuvent même se révéler contradictoires, selon l'utilisation qui en est faite : stabilisation de l'entraînement lorsque les augmentations adaptatives (voir section suivante) ne sont pas utilisées, dégradation des résultats lorsqu'elles le sont.

5.2.2.6 Augmentations adaptatives pour le discriminateur

Karras *et al.* [KAH+20] ont montré que les discriminateurs ont tendance à sur-apprendre (*overfit*) très tôt pendant l'entraînement, et en particulier, sans surprise, sur les petits jeux de données. Le système GAN se met dès lors à diverger, le générateur apprenant directement du discriminateur. Nous l'avons vu, le sur-apprentissage est une problématique récurrente de l'entraînement des modèles par réseaux de neurones, repoussée par l'augmentation de données. Les GANs ne font pas exception. Un ensemble de techniques d'augmentation est appliqué aux images montrées au discriminateur, qu'elles soient synthétiques ou réelles. Ces techniques sont globalement les mêmes que celles introduites dans le premier chapitre : rotations, déformations spatiales, modifications de couleur etc. Chaque augmentation est contrôlée par un paramètre de probabilité $p \in [0,1]$. Les auteurs travaillent sur ce paramètre de probabilité et cherchent à déterminer les valeurs les plus bénéfiques. Tout d'abord, pour que les augmentations ne se retrouvent pas dans les images générées, les auteurs déterminent expérimentalement une valeur de probabilité maximale de sécurité, d'environ 0,8. Ils montrent également que la valeur de p idéale varie selon la taille des jeux de données. En particulier pour ceux de petite taille (2 000 images dans leur exemple, nous en avons moins) les augmentations ayant le plus d'influence sont les rotations et les transformations géométriques. Afin de ne pas devoir faire une recherche exhaustive du meilleur p pour chaque jeu de donnée, les auteurs proposent des augmentations adaptatives (ADA), avec un p variant durant l'entraînement. Les auteurs définissent une heuristique $r = \mathbb{E}[sign(D_{train})]$ mesurant le sur-apprentissage du discriminateur. Si $r = 0$, il n'y a pas de sur-apprentissage, si $r = 1$, le discriminateur est en sur-apprentissage complet. Ainsi, le paramètre p est mis à jour pendant l'entraînement en fonction de la valeur de r. Si r est inférieur à une valeur de référence (0,6 dans le papier), alors p est diminué, et inversement. Concrètement, plus le discriminateur sur-apprend, plus les augmentations sont renforcées. Pour des petits jeux de données, p va varier de 0 au début de l'entraînement, jusqu'à la valeur limite fixée, 0,8 dans notre cas.

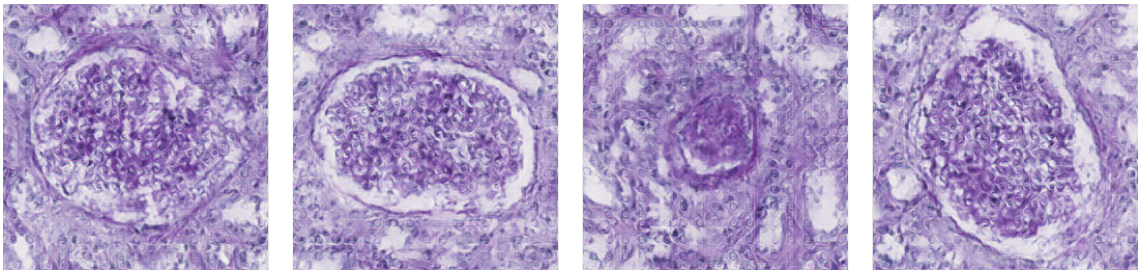


Figure 5.10 : Images générées par SPADE avec augmentations adaptatives.

En pratique, nous utilisons des augmentations classiques pour l'entraînement de SPADE : rotations, symétries, ainsi que des déformations basée sur une grille régulière avec interpolation à l'ordre 3, telles que définies au chapitre 3. L'utilisation des augmentations adaptatives permet de pallier l'effondrement de mode, mais seulement partiellement puisque des motifs se répètent tout de même entre les images, et au prix de résultats visuellement moins fidèles, comme illustré par la figure 5.10.

5.2.2.7 Augmentations *Remix*

Cao *et al.* [CHY+21] attaquent également le problème du sur-apprentissage par le prisme de l'augmentation de données, en se basant sur l'interpolation d'exemples dans l'espace des caractéristiques. Les auteurs reformulent la fonction de coût du générateur de la manière suivante (en reprenant nos notations) :

$$L(D,G) = L_{GAN}(D,G) + L_{cont}(\phi, G), \tag{5.15}$$

où ϕ est une fonction dont le rôle est d'extraire une représentation du contenu des images. Parmi les représentations vues dans ce manuscrit, ϕ pourrait être une fonction calculant les matrices de Gram ou la fonction de *feature matching* vue dans ce chapitre. En notant e_1 et e_2 deux cartes de caractéristiques intermédiaires issues de deux images x_1 et x_2 du jeu d'entraînement, leur interpolation est définie comme $e = \lambda e_1 + (1-\lambda)e_2$ avec $\lambda \in [0,1]$. L'interpolation directe des données d'entrée est un cas particulier de cette formulation. Durant l'entraînement du GAN, chaque lot (*batch*) de données est remplacé par sa version interpolée avec une probabilité p. Si l'ensemble n'est pas interpolé, alors l'itération se déroule normalement. Si l'ensemble est interpolé, deux ensembles $(x_i^1, t_i^1)_{i=1}^n$ et $(x_i^2, t_i^2)_{i=1}^n$ sont sélectionnés, où t^1 et t^2 sont les cibles, et n la taille de l'ensemble. Le poids d'interpolation λ est obtenu de la manière suivante :

$$\mu = \mathcal{B}(\alpha, \alpha) \tag{5.16}$$

où $\mathcal{B}$ est la loi de probabilité bêta, puis :

$$\lambda = \max(\mu, 1-\mu). \tag{5.17}$$

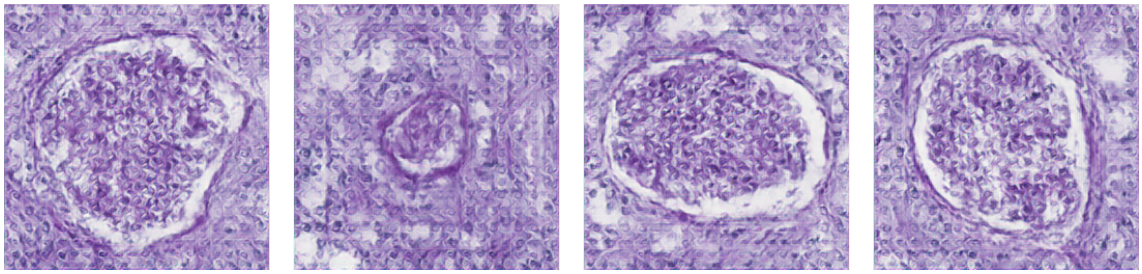


Figure 5.11 : Images générées par SPADE avec augmentations *Remix*.

Le paramètre α est fixé à 0,2, d'après la valeur utilisée dans l'article. Nous obtenons ainsi les entrées augmentées $\{e_i\}$.

L_{GAN} est calculée normalement. L_{cont} est calculée dans sa forme dite relative, notée L'_{cont}, et définie par $L'_{cont} = L_p + L_n$, avec :

$$L_p = \sum_i \max\{0, L_{cont}(\phi(G(e_i)), \phi(t_i^1)) - L_{cont}(\phi(G(e_i)), \phi(t_i^2))\} \tag{5.18}$$

et

$$L_n = \sum_i \max\{0, L_{cont}(\phi(G(e_i)), \phi(t_i^2)) - \bar{a}\} \tag{5.19}$$

Notons qu'avec la définition précédente, λ est toujours plus grand que 0,5, ce qui donne un poids plus important à e^1 dans l'interpolation. La méthode force donc les représentations d'images générées par entrées interpolées à s'approcher de la représentation de t^1. Par ailleurs, $\bar{a}$ est initialisé à 0 et est mis à jour durant l'entraînement :

$$\bar{a} \leftarrow 0{,}99 \cdot \bar{a} + 0{,}01 \cdot (a - \bar{a}), \tag{5.20}$$

où

$$a = \sum_{i \neq j} L_{cont}(\phi(G(e_i)), \phi(t_j^2)). \tag{5.21}$$

La variable a représente la distance moyenne entre les représentations des sorties et des cibles au sein du lot (*batch)* considéré.

La méthode ne semble pas apporter d'amélioration notable lors d'un examen visuel des résultats. Les images perdent même en réalisme puisque certains motifs sont placés de manière périodique, avec notamment des alignements de noyaux, comme montré sur la figure 5.11.

5.2.3 Cartes de structure

Les images produites par SPADE souffrent d'un manque de variété ou d'un manque de qualité visuelle, selon que les augmentations adaptatives sont utilisées ou non. Les GANs

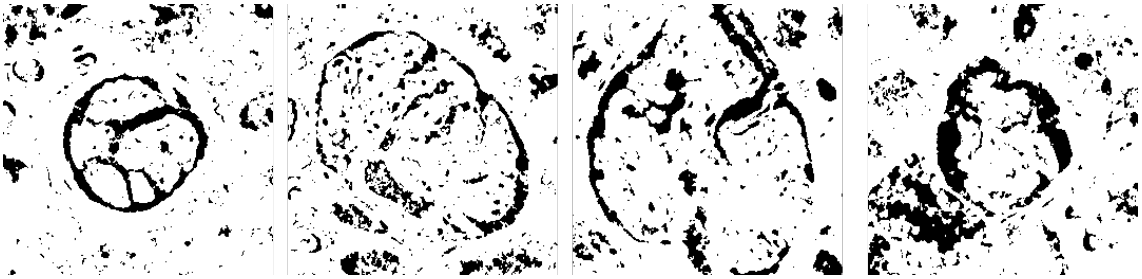


Figure 5.12 : Masques binaires de zones texturées, obtenus en seuillant la composante S d'images de glomérules converties en HSV.

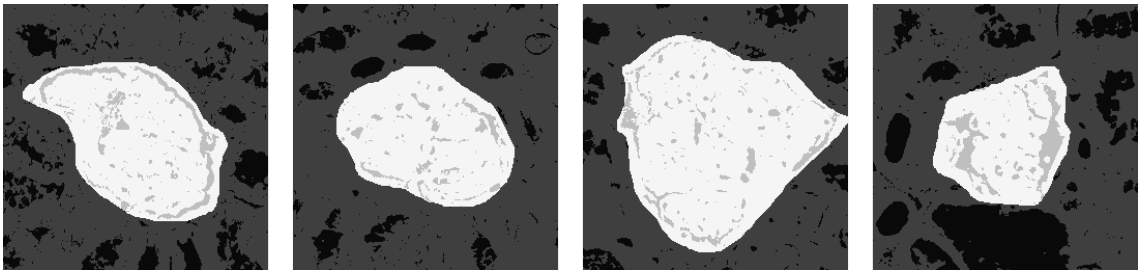


Figure 5.13 : Cartes de label structurées, obtenues en combinant des masques binaires de zones texturées avec les masques binaires de vérité terrain correspondants.

sont très performants pour la synthèse de texture et assez peu pour la synthèse de structure, malgré les techniques utilisées pour lever les problèmes rencontrés. Une idée, héritée des travaux de la communauté de l'informatique graphique tels que vus au chapitre précédent et des cartes de frontières utilisées pour Pix2PixHD [WLZ+18] consiste à guider le générateur en modifiant l'entrée. Dans notre cas, l'ajout d'une information portant sur la structure des images, c'est-à-dire l'organisation spatiale des motifs et leur forme, pourrait permettre une synthèse de meilleure qualité. Nous proposons alors d'agrémenter les cartes binaires de vérité terrain à l'aide de cartes de structure extraites directement des images de glomérule. Puisqu'il n'existe pas de vérité terrain associée, il est impossible de segmenter avec précision des sous-structures pertinentes. Nous avons choisi de séparer les zones non-texturées, correspondant à des zones sans tissu (coupes de vaisseaux, espace urinaire ou extérieur de la coupe par exemple), des zones texturées, correspondant aux glomérules et aux tissus environnant. Pour ce faire, nous admettons que les zones non-texturées ne sont pas affectées par la coloration ou par d'importants artefacts, et correspondent à la couleur de la lumière du microscope, qui est proche du blanc [KRP+14]. Nous convertissons les patches de glomérule en représentation HSV et seuillons la composante S pour obtenir des masques binaires des zones texturées. La figure 5.12 montre des exemples de tels masques. Nous fixons empiriquement le seuil à 0,1 pour notre jeu de données. Ensuite, nous combinons ces masques avec les vérités terrain correspondantes afin d'obtenir des cartes de labels à quatre classes que nous nommons « cartes de labels structurées », ou parfois simplement cartes de structure (voir figure 5.13).

Nous pouvons ensuite entraîner un modèle SPADE en utilisant ces cartes comme entrées,

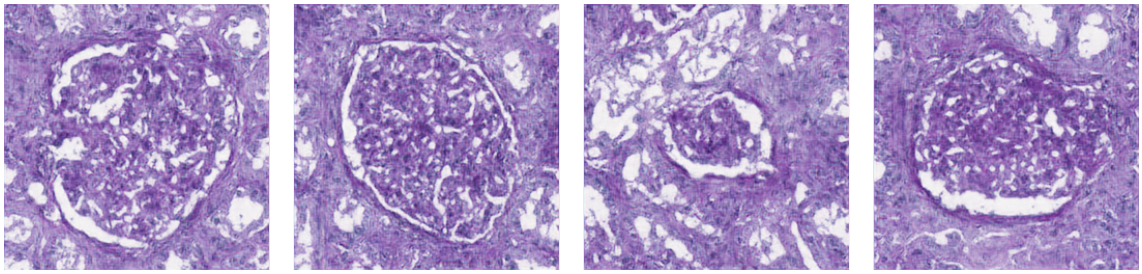


Figure 5.14 : Images produites par un modèle SPADE-ADA avec cartes de labels structurées en entrée.

ce qui nous permet d'obtenir des résultats visuellement plus satisfaisants (voir figure 5.14), même si des sous-structures restent absentes. Les noyaux de cellule en particulier sont moins nombreux que sur les images réelles. Cette méthode permet d'éliminer complètement l'effondrement de mode.

Toutefois, cette manière de procéder ne fait que reporter le problème : pour compléter la chaîne de traitement, il faut pouvoir synthétiser également ces cartes de structure. Des pistes ont été envisagées, notamment l'utilisation d'un Style-GAN2 [KLA+20], d'auto-encodeurs variationnels [KW14] ou d'un générateur procédural à base de bruit de Perlin, mais aucune de ces solutions n'est satisfaisante. Il reste de nombreuses pistes à explorer, pouvant constituer de futurs travaux. Pour la suite du chapitre, nous utiliserons des cartes de structure issues d'images réelles mises de côté à cet effet, afin de pouvoir tout de même étudier les bénéfices potentiels en termes d'augmentation de données sur notre problème. Ces images sont issues normalement de la base de test et n'ont donc pas vocation à être utilisées pour l'entraînement, mais nous utilisons cette répartition pour les besoins d'expérimentation. La répartition est la suivante :

- Entraînement du U-Net (baseline) et de SPADE : 660 images ;
- Validation du U-Net : 400 images ;
- Test du U-Net : 431 images ;
- Validation du SPADE : 158 images ;
- Test du SPADE, pour la génération de « nouvelles » images : 660 images.

Notons que cette répartition était déjà effective au chapitre précédent, et que donc les résultats sont directement comparables.

5.3 Entraînement du U-Net avec des images générées par un SPADE

5.3.1 Simulation de nouvelles images

Nous allons, pour cette section, entraîner différentes configurations du SPADE. Chaque modèle sera utilisé pour produire des images, qui serviront à entraîner un modèle U-Net. La configuration de base est constituée du modèle SPADE avec augmentations adaptatives, que nous nommerons SPADE-ADA. Ce modèle sera entraîné seul, puis nous ajouterons, séparément :

- les augmentations *ReMix* (SPADE-ADA-REMIX), avec p=0,25 ;
- l'erreur sur les matrices de Gram (SPADE-ADA-ST), avec un poids de 0,0001 ;
- la régularisation *Mode Seeking* (SPADE-ADA-MS), avec $L_{ms} = 1$.

Pour chaque version, nous étudierons la version sans puis avec cartes de structure. Comme indiqué à la fin de la section précédente, des paires glomérules/vérité terrain ont été gardées de côté pour extraire des cartes de structure et pouvoir produire de « nouvelles » images grâce à SPADE. Ces images se trouvent originellement dans la base de test et sont au nombre de 660. Nous doublerons donc notre base d'entraînement avec des images synthétiques. Comme pour le chapitre précédent, nous procéderons à des entraînements d'un modèle U-Net avec images synthétiques uniquement d'une part, et avec un mélange d'images réelles et synthétiques d'autre part. Chaque entraînement dure 300 époques.

Nous présentons d'abord au tableau 5.1 les indices de Dice moyens pour l'entraînement d'un U-Net avec images synthétiques uniquement. Nous rappelons le résultat pour un entraînement avec images réelles uniquement, obtenu au chapitre précédent : $93{,}71 \pm 0{,}10$. Les résultats obtenus ici sont tous significativement inférieurs, et notamment inférieurs aux résultats avec les images générées par synthèse de texture, qui étaient eux-mêmes significativement proches des images réelles. Nous ne parvenons donc pas à produire des images de bonne qualité, et ce peu importe la version de SPADE. Les meilleurs résultats sont obtenus grâce aux versions SPADE-ADA avec masque binaire et SPADE-ADA-ST avec carte de structure. La comparaison entre les deux types de cartes d'entrée est surprenante : les cartes de structure, qui produisent pourtant des images visuellement plus satisfaisantes, ne donnent pas forcément les meilleurs résultats de segmentation (cas SPADE-ADA et SPADE-ADA-REMIX).

Le tableau 5.2 montre les résultats de l'entraînement d'un U-Net avec un mélange d'images réelles et synthétiques, toujours en faisant varier les cartes d'entrée et la version du SPADE. La version SPADE-ADA avec masque binaire donne significativement les meilleurs résultats ici ($94{,}02 \pm 0{,}19$ contre $93{,}71 \pm 0{,}10$ pour l'entraînement avec images réelles uniquement). Toutes les autres méthodes n'apportent aucune amélioration significative.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	89,95 ± 0,94	79,58 ± 4,48	46,80 ± 14,45	83,19 ± 1,45
Carte de structure	80,88 ± 1,73	79,41 ± 1,30	84,80 ± 0,81	89,69 ± 1,35

Table 5.1 : Indice de Dice moyen en fonction de la version de SPADE et de l'entrée utilisée, pour des entraînements de U-Net avec images synthétiques uniquement.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	94,02 ± 0,19	93,68 ± 0,24	93,51 ± 0,19	93,10 ± 0,38
Carte de structure	93,90 ± 0,20	93,77 ± 0,11	93,07 ± 0,25	93,73 ± 0,11

Table 5.2 : Indice de Dice moyen en fonction de la version de SPADE et de l'entrée utilisée, pour des entraînements de U-Net avec images réelles et synthétiques à proportions égales.

Il est également possible d'entraîner un U-Net avec une base de données mélangeant images réelles, images générées par masques binaires et images générées par cartes de structure. Nous obtenons ainsi l'indice de Dice moyen suivant : 94,09 ± 0,14, légèrement supérieur au meilleur score obtenu précédemment (94,02±0,19), mais non significativement. Il est donc possible de tirer parti des différentes variations de bruit introduites par les deux versions afin d'améliorer légèrement les performances.

En complément, nous effectuons quelques entraînements supplémentaires, afin de déterminer les limites d'utilisation de SPADE-ADA dans notre contexte. Premièrement, nous entraînons un U-Net avec l'ensemble d'entraînement usuel augmenté des images réelles ayant servi à la génération des augmentations. Ce nouveau jeu d'entraînement permet ainsi d'obtenir une valeur limite : si SPADE-ADA produisait des images parfaites, il serait impossible d'avoir de meilleurs résultats. Nous entraînons ensuite un SPADE-ADA en utilisant les images réelles comme cartes de structure. Ainsi, SPADE-ADA devient un réseau de coloration, de niveaux de gris vers RGB. Le générateur est, dans ce cadre, aussi proche que possible de la fonction identité, et représente ainsi la limite expérimentale atteignable avec ce modèle. Nous entraînons un U-Net avec des images réelles et des images synthétiques issues de ce modèle. Les résultats obtenus sont les suivants :

- limite expérimentale pour des images parfaites : 95,02 ± 0,06 ;
- limite expérimentale pour le modèle SPADE-ADA : 94,91 ± 0,11.

Les deux valeurs sont significativement supérieures aux autres résultats, et également significativement proches entre elles. La limite théorique atteignable par un SPADE en tant que méthode d'augmentation de données est équivalente à l'ajout de nouvelles données réelles.

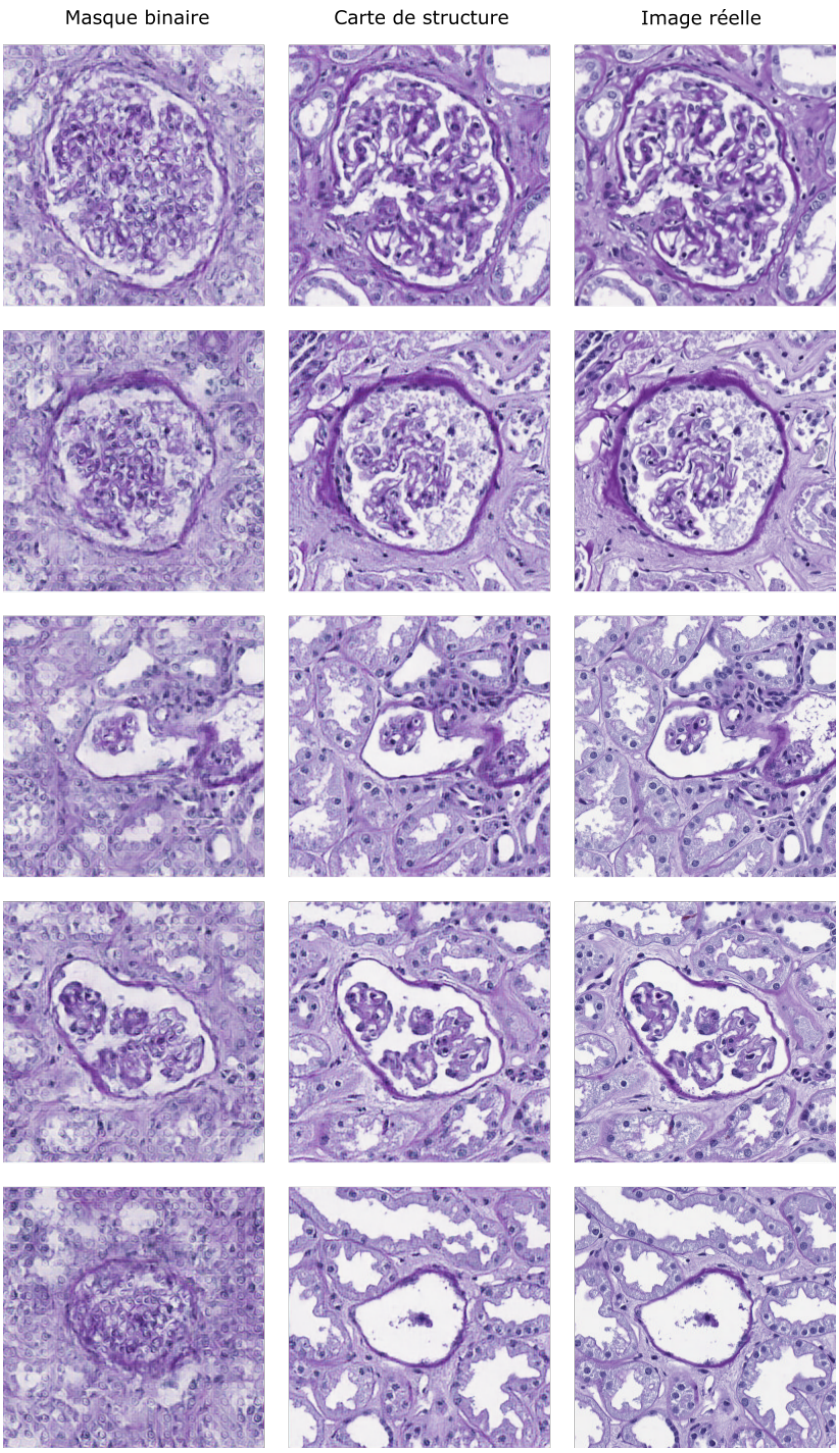


Figure 5.15 : Reproduction de la base d'entraînement par des modèles SPADE. Première colonne : images générées avec le masque binaire seul en entrée du générateur. Deuxième colonne : images générées avec une carte de labels structurée en entrée. Troisième colonne : image réelle.

5.3.2 Conditions réelles : reproduction de la base d'entraînement

Pour cette section, nous revenons à nos conditions réelles : nous n'utilisons pas les images que nous avions mises de côté. Pour produire des images synthétiques, nous utilisons les mêmes images que pour l'entraînement du modèle génératif. Nécessairement, les images synthétiques sont de meilleure qualité visuelle que les images générées à l'aide de cartes de labels inconnues, mais ne sont qu'une reproduction de la base d'entraînement, avec de légères variations de textures (voir la figure 5.15). En ce sens, nous retrouvons le même cadre que dans le chapitre précédent. Toutes les images produites sont de meilleure qualité visuelle que précédemment, en particulier celles générées avec les masques binaires seuls en entrée. Nous notons toutefois un effondrement de mode partiel, avec des motifs qui se répètent d'image en image. Les images générées à l'aide des cartes de labels structurées sont difficiles à distinguer des images réelles. Nous reprenons la même méthodologie que précédemment : entraînement d'un U-Net avec images synthétiques uniquement, avec mélange d'images réelles et synthétiques, générées avec ou sans cartes de structure. Nous utilisons uniquement la version SPADE-ADA ici. La figure 5.16 montre les indices de Dice moyens obtenus.

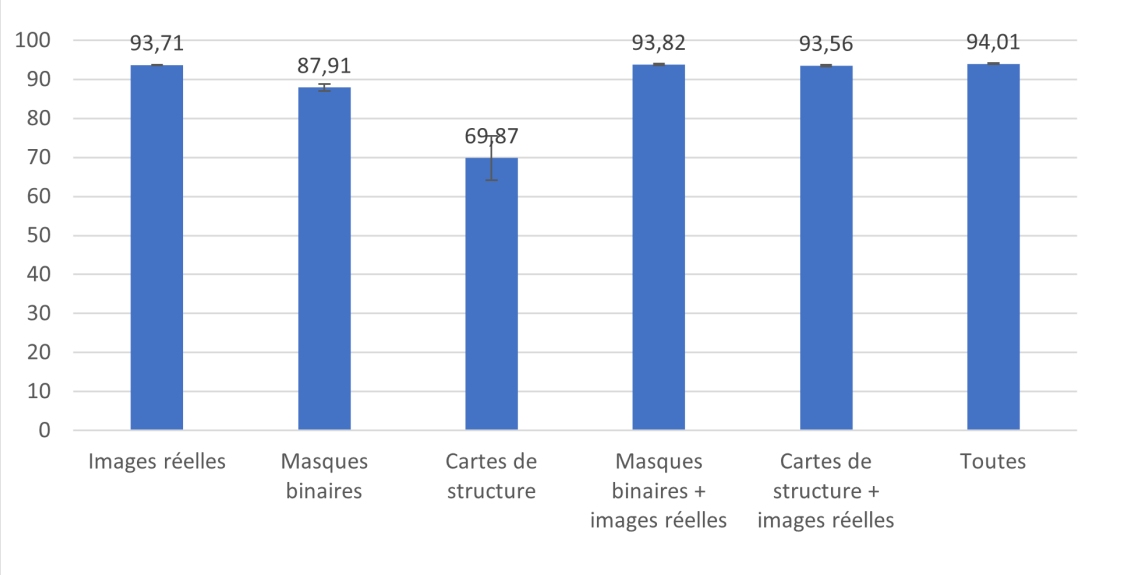


Figure 5.16 : Indices de Dice moyens obtenus pour chaque jeu de données. « Masques binaires » et « Cartes de structure » désignent un jeu de données synthétiques obtenu avec le type d'entrée indiqué.

Dans le cas des entraînements utilisant images synthétiques et images réelles, nous obtenons des résultats moins bons que précédemment. L'augmentation réelle consistant en des variations locales (pas de nouvelles images), ce résultat n'est pas surprenant. La seule version apportant une amélioration significative est celle où toutes les images (réelles, générées par masques binaires et générées par carte de structure sont utilisées), avec un indice de Dice moyen sensiblement identique à celui obtenu dans la section précédente pour la version SPADE-ADA avec masques binaires ($94{,}01 \pm 0{,}14$ et $94{,}02 \pm 0{,}19$ respectivement).

Il est donc possible, dans la configuration réelle (pas de nouvelles images) d'obtenir des résultats équivalents à l'ajout d'images générées complètement nouvelles, en tirant profit une nouvelle fois des différents types de modifications de texture locales introduites par les deux versions de SPADE-ADA (masques binaires et cartes de structure).

Les résultats obtenus en utilisant uniquement les images synthétiques sont en revanche inattendus. Les images étant de meilleur qualité, nous devrions avoir de meilleurs performances de segmentation que dans la section précédente, notamment dans le cas avec cartes de labels structurées où les images sont difficilement discernables des images réelles. Nous passons d'un indice de Dice moyen de $80{,}88 \pm 1{,}73$ à la section précédente, à un indice de $69{,}87 \pm 5{,}67$ ici. De manière générale, les résultats obtenus avec les cartes de label structurées sont décevants. Nous essayons dans la section suivante d'apporter des éléments de réponse.

5.3.3 Analyse visuelle des résultats de segmentation

La figure 5.17 montre des résultats de segmentation d'images de la base de test, après entraînement avec des images générées par un modèle SPADE-ADA avec cartes de labels structurées. Certaines images ne sont quasiment pas segmentées, comme c'est le cas à la deuxième et à la troisième ligne, ce qui explique l'écart entre les performances attendues et les performances réelles. D'autres images sont correctement segmentées, ce qui indique que le U-Net apprend tout de même correctement. Ces résultats sont spécifiques à l'utilisation des cartes de labels structurées, et ne se retrouvent avec aucun autre jeu de données.

De plus, l'utilisation d'un U-Net entraîné sur des images réelles et testé sur les images synthétiques donne un indice de Dice de 93,87, ce qui montre que les images générées par cartes de structure sont correctement segmentées. Nous ne sommes donc pas *a priori* dans le cadre d'une attaque adversaire [VFWL21a], c'est-à-dire d'images comportant un bruit imperceptible perturbant suffisamment le réseau pour faire échouer l'apprentissage. Par ailleurs, des résultats préliminaires montrent que l'entraînement d'un modèle U-Net avec un encodeur pré-entraîné donne des performances en adéquation avec ce que nous attendions, c'est-à-dire de meilleures performances dans le cas d'images générées avec des cartes de labels structurées en entrée que dans le cas avec masque binaire en entrée. Tirer une conclusion définitive sur les cartes de labels structurées est donc pour le moment difficile et nécessite plus d'investigation.

5.4 Conclusion et pistes sur les GANs

Dans ce chapitre, nous avons étudié les méthodes de translation d'image-à-image, dans le cadre de la génération de patches de glomérules, et en particulier des variations du modèle SPADE. Nous avons proposé diverses régularisations provenant de l'état de l'art afin de lever le problème de l'effondrement de mode et d'améliorer la qualité visuelle des résultats.

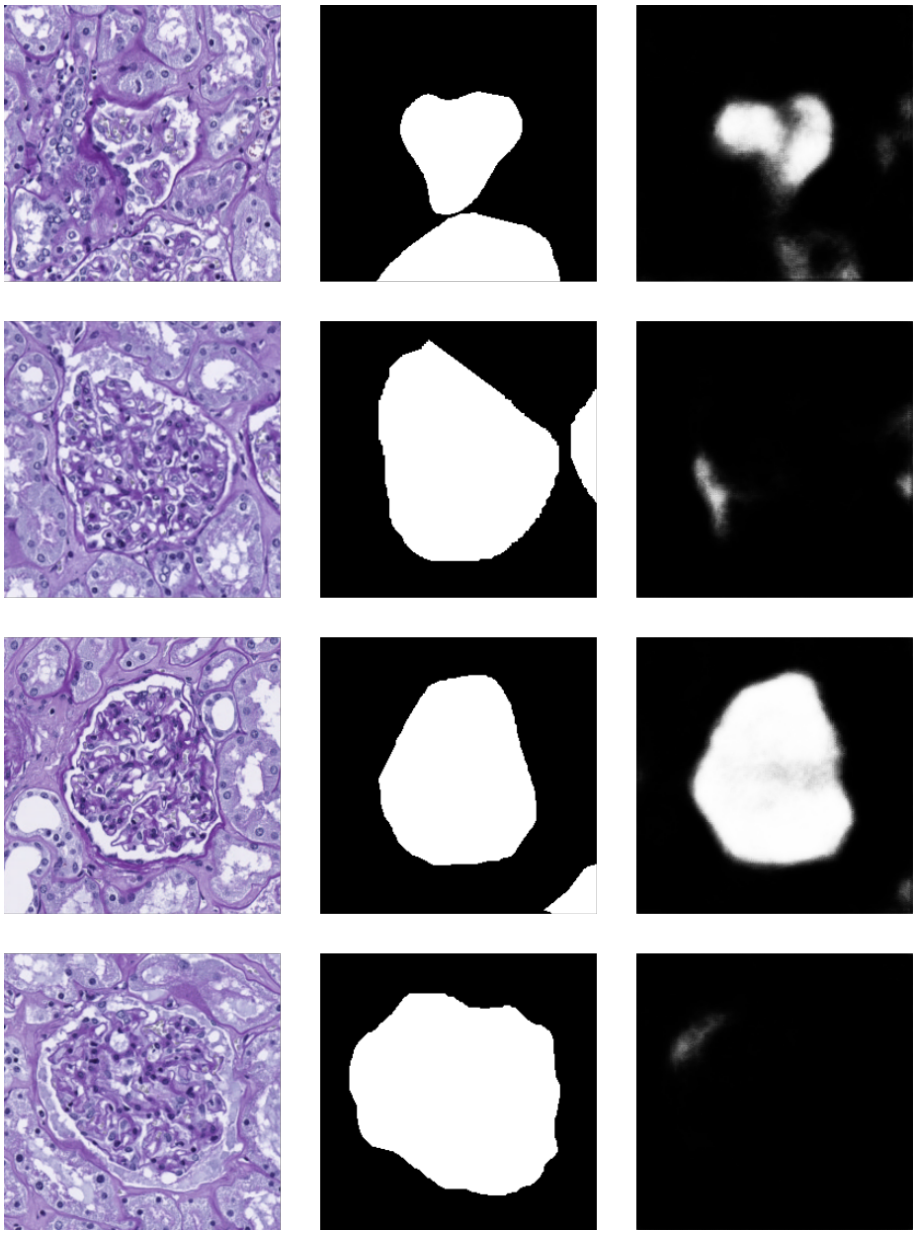


Figure 5.17 : Segmentations d'images de la base de test, après entraînement d'un U-Net sur des images SPADE-ADA avec cartes de labels structurées. Première colonne : image réelles, deuxième colonne : vérité terrain, troisième colonne : sortie du U-Net, avant seuillage.

En particulier, l'utilisation d'augmentations adaptatives permet de pallier l'effondrement de mode, et l'introduction de cartes de labels structurées extraites des images de glomérules permet d'obtenir des résultats avec plus de détails, notamment au niveau des structures. Nous avons ensuite utilisé les images générées pour augmenter notre jeu de données, en vue de l'entraînement de modèles de segmentation U-Net, afin de comparer les différentes approches. Nous avons également quantifié les résultats théoriques atteignables avec la technique d'augmentation de données employée dans ce chapitre.

Nous avons montré que le modèle SPADE-ADA, entraîné avec des masques de segmentation, ne parvient pas à générer de nouvelles images visuellement réalistes. Les images produites sont floues et manquent de détails, notamment certaines sous-structures. Elles sont toutefois utiles pour augmenter notre jeu de données pour la segmentation. Nous avons montré que l'ajout de ces images à la base d'entraînement permet d'obtenir un gain d'indice de Dice moyen de 0,3%. L'utilisation des cartes de labels structurées en entrée de SPADE-ADA permet d'obtenir des images visuellement plus proches des images réelles, mais ne présente pas de bénéfices pour l'augmentation de données. De même, la reproduction des images d'entraînement par SPADE-ADA, avec ou sans cartes de structures, bien que très fidèle, ne présente pas ou peu de bénéfices pour l'augmentation de données, et donne de moins bons résultats que les images moins réalistes utilisées précédemment. L'apparence visuelle des images générées ne donne donc pas d'information *a priori* sur leur utilité potentielle pour l'augmentation de données. Les instabilités observées avec des images générées par cartes de labels structurées restent à expliquer, d'autant que d'autres modèles de segmentation ne semblent pas impactés.

L'utilisation de GANs ne présente actuellement pas de bénéfice comparé aux méthodes de synthèse de texture, puisque nous ne parvenons pas à obtenir de meilleurs résultats de segmentation. Les GANs souffrent d'instabilités lors de leur entraînement, instabilités qui sont renforcées par la petite taille de notre jeu de données et la complexité des images qui le composent. De futurs travaux peuvent se concentrer sur des adaptations éventuelles du modèle SPADE pour dépasser ses limites actuelles, et notamment comprendre exactement pourquoi des images de très bonne qualité visuelle donnent de moins bonnes performances de segmentation. D'autres méthodes par GANs telles que StyleGAN2 [KAH+20; KLA+20] ou BigGAN [BDS19] peuvent être adaptées pour de la translation d'image-à-image, ou de toute autre manière qui pourrait être utile pour notre application, et sont en cours d'investigation. Il existe également d'autres types de modèles génératifs tels que les Auto-Encodeurs Variatonnels (*Variational Auto-Encodeurs*, VAE) [KW14; VK20], ne souffrant pas en théorie de certaines limitations des GANs, ainsi que les modèles de diffusion [RBL+22]. Ces derniers sont très prometteurs et attirent l'attention, présentant des résultats supérieurs aux GANs [DN21]. Ils commencent déjà à être appliqués à la synthèse d'images médicales [PTD+22], dans le cadre d'images IRM de cerveaux. En particulier, un modèle capable de générer plus de variété dans les motifs à l'échelle globale pourrait améliorer plus sensiblement les performances de la segmentation.

Chapitre 6

Conclusion

Dans cette thèse, nous nous sommes intéressés à la segmentation supervisée de glomérules, dans des patchs issus de coupes complètes de tissus rénaux. Nous avons utilisé un modèle de Deep Learning de type U-Net, faisant partie des modèles les plus utilisés pour l'analyse d'images médicales. Le peu d'images annotées disponibles, de l'ordre de quelques centaines, pose la question des augmentations de données. L'état de l'art en traitement d'images histopathologiques montre que les augmentations les plus utilisées comprennent des transformations affines (rotations, symétries), des déformations spatiales aléatoires et des transformations colorimétriques. Des méthodes par GANs sont également employées pour générer de nouveaux exemples ou faire des transferts de coloration. Dans le cadre de la segmentation de glomérules, les déformations spatiales aléatoires sont utilisées mais sans justification sur les méthodes employées ni leurs paramètres. Les transferts de coloration par GANs sont bien étudiés et l'impact sur les performances de segmentation quantifié. Il n'existe en revanche pas de travaux utilisant des GANs pour générer des patchs de glomérules dans le cadre de l'entraînement d'un modèle de segmentation. Nous avons donc proposé d'étudier deux types d'augmentation de données : les méthodes par déformations spatiales aléatoires, qui permettent d'introduire des variations géométriques, et les méthodes par génération d'exemples, qui permettent d'introduire des variations de texture. Pour le second type d'augmentation, nous avons étudié deux méthodes, une par synthèse de texture et une par modèles génératifs adversaires. Nous avons adapté des méthodes existantes aux caractéristiques des images de glomérules et développé des cadres de comparaison et d'évaluation des différentes méthodes. Les méthodes et cadres de comparaison employés sont utilisables pour d'autres images histopathologiques.

Nous avons montré que le choix et le paramétrage des augmentations est crucial pour tirer le meilleur bénéfice de chaque augmentation, et qu'une approche binaire (utilisation ou non) n'est pas suffisante. Concernant les méthodes de déformation, nous avons sélectionné et comparé plusieurs méthodes de l'état de l'art, et proposé une adaptation de méthodes existantes, que nous avons appelé CNB-MVN-MLS, permettant de diminuer le nombre de paramètres à contrôler. Nous avons montré que dans le contexte d'un très faible nombre

d'images, toutes les méthodes proposées offrent des performances d'augmentation similaires, et permettent d'améliorer significativement les résultats obtenus. Pour les méthodes génératives, les bénéfices apportés sont relativement modestes, les méthodes utilisées apportant des variations de texture à petite échelle mais n'étant pas capables de produire de nouveaux motifs. Nous avons proposé une variation des méthodes de synthèse de texture par l'exemple, la synthèse de texture guidée multi-échelles par réseaux de neurones, ne nécessitant pas d'entraînement et permettant de reproduire des images de glomérules avec fidélité, tout en introduisant des variations de texture locales. Par ailleurs, les GANs ont des difficultés à produire des résultats visuellement satisfaisants. Nous avons proposé l'utilisation de cartes de labels structurées pour guider la synthèse d'un modèle de translation d'image-à-image, SPADE, mais les images obtenues, bien que de meilleure qualité, ne permettent pas d'obtenir les meilleurs résultats de segmentation. Nous avons mis en évidence que la proximité visuelle entre les images issues des méthodes d'augmentation et les images réelles est nécessaire seulement jusqu'à un certain point et ne permet pas *a priori* de juger des gains potentiels pour la segmentation.

Notre travail présente un certain nombre de limites, héritées du domaine du Deep Learning. Les conclusions faites sur nos images ne sont pas transposables à d'autres jeux de données, que ce soit un jeu contenant des images différentes, même en histopathologie numérique, ou un jeu d'images de glomérules de taille différente. L'optimisation des paramètres des méthodes est à effectuer systématiquement, ce qui est coûteux en temps et en énergie. Les résultats obtenus dans cette thèse représentent des milliers d'heures de calcul sur GPU. Il est nécessaire de repenser le Deep Learning et l'apprentissage statistique de manière générale, en développant par exemple des approches capables d'apprendre à partir de peu d'exemples et donc moins consommatrices en ressources de calcul, telles que le *Few-shot Learning* [MPS+19; WYKN20] ou le *Transfer Learning* [ATAO+22; MAKM22]. Introduire des connaissances biologiques peut également améliorer les performances des modèles [BEQD17; FB20]. Par ailleurs, la promesse des réseaux de neurones d'incorporer la phase d'extraction de caractéristiques, censée permettre une application plus facile et plus rapide des modèles, cache le nombre d'hyper-paramètres qui restent à optimiser (nature du réseau, nombre de couches, taille des filtres, fonction de coût, augmentations et leurs paramètres, etc). S'il existe quelques modèles de référence très performants, le système de publication pousse les chercheurs à proposer de nombreuses variantes, souvent spécifiques à certains jeux de données, conduisant à une augmentation exponentielle des articles traitant ou utilisant du Deep Learning. L'approche expérimentale oblige à tester toutes les solutions proposées, tandis que les codes ne sont pas toujours disponibles et la reproduction des résultats montrés dans les articles parfois impossible. Par ailleurs, les différences en termes de moyens humains et matériels peuvent créer des déséquilibres entre recherche publique et privée.

Les pistes potentielles pour de futurs travaux sont nombreuses. L'utilisation d'autres modèles de déformation est envisageable, notamment ceux utilisant des réseaux de neurones (Spatial Transformer Networks [JSZK15; DFH18]). Concernant les modèles génératifs, il est possible d'adapter d'autres modèles existants à notre tâche. Les modèles de diffusion [RBL+22] en particulier sont prometteurs. Un modèle capable de reproduire les

structures dans nos images et de générer plus de variété dans les motifs à différentes échelles pourrait apporter de meilleures performances de segmentation. Nous pouvons également nous intéresser à la quantification de l'impact d'autres types d'augmentation de données, notamment en essayant d'introduire des variations topologiques grâce à des méthodes procédurales. Étudier l'impact des augmentations sur d'autres modèles de segmentation, en particulier les modèles avec un encodeur pré-entraîné, pourrait également se révéler intéressant. L'utilisation des augmentations considérées dans ce manuscrit et du cadre d'évaluation sur des jeux de données partagés est une étape importante afin de pouvoir proposer une meilleure comparaison des résultats obtenus.

Publications de l'auteur

[AAWD22a] Florian Allender, Rémi Allègre, Cédric Wemmert, and Jean-Michel Dischler. Conditional image synthesis for improved segmentation of glomeruli in renal histopathological images. In *IEEE-EMBS International Conference on Biomedical and Health Informatics, Sept. 27-30 2022, Ioannina, Greece*, Ioannina, Greece, September 2022.

[AAWD22b] Florian Allender, Rémi Allègre, Cédric Wemmert, and Jean-Michel Dischler. Data augmentation based on spatial deformations for histopathology : An evaluation in the context of glomeruli segmentation. *Computer Methods and Programs in Biomedicine*, 221 :106919, 2022.

Références

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow : Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[ABM] Agence de la biomédecine : greffe rénale. https://rams.agence-biomedecine.fr/greffe-renale-0.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[ACB+20] Nicola Altini, Giacomo Donato Cascarano, Antonio Brunetti, Francescomaria Marino, Maria Teresa Rocchetti, Silvia Matino, Umberto Venere, Michele Rossini, Francesco Pesce, Loreto Gesualdo, and Vitoantonio Bevilacqua. Semantic segmentation framework for glomeruli detection and classification in kidney histological sections. *Electronics*, 9(3), 2020.

[ada] gan-ada. https://keras.io/examples/generative/gan_ada/.

[AFF+15] G. Apou, F. Feuerhake, G. Forestier, B. Naegel, and C. Wemmert. Synthesizing whole slide images. In *2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 154–159, Sept 2015.

[ASN+16] Grégory Apou, Nadine S. Schaadt, Benoît Naegel, Germain Forestier, Ralf Schönmeyer, Friedrich Feuerhake, Cédric Wemmert, and Anne Grote. Detection of lobular structures in normal breast tissue. *Computers in Biology and Medicine*, 74 :91 – 102, 2016.

[ATAC+21] Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. Deep semantic segmentation of natural and medical images : a review. *Artificial Intelligence Review*, 54(1) :137–178, 2021.

[ATAO+22] Tayyab Aitazaz, Abdullah Tubaishat, Feras Al-Obeidat, Babar Shah, Tehseen Zia, and Ali Tariq. Transfer learning for histopathology images : an empirical study. *Neural Computing and Applications*, pages 1–12, 07 2022.

[AYH+19] Md Zahangir Alom, Chris Yakopcic, Mahmudul Hasan, Tarek M. Taha, and Vijayan K. Asari. Recurrent residual U-Net for medical image segmentation. *Journal of Medical Imaging*, 6(1) :1 – 16, 2019.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096, 2019.

[BEQD17] Shahin Boluki, Mohammad Shahrokh Esfahani, Xiaoning Qian, and Edward R Dougherty. Incorporating biological prior knowledge for bayesian learning via maximal knowledge-driven information priors. *BMC Bioinformatics*, 18(S14), 2017.

[BFCGLD20] Gloria Bueno, M. Milagro Fernandez-Carrobles, Lucia Gonzalez-Lopez, and Oscar Deniz. Glomerulosclerosis identification in whole slide images using semantic segmentation. *Computer Methods and Programs in Biomedicine*, 184 :105273, 2020.

[BJV17] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 469–477. PMLR, 06–11 Aug 2017.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12) :2481–2495, 2017.

[BLH+20] Laura Barisoni, Kyle J. Lafata, Stephen M. Hewitt, Anant Madabhushi, and Ulysses G. J. Balis. Digital pathology and computational image analysis in nephropathology. *Nature Reviews Nephrology*, 16(11) :669–685, Nov 2020.

[Boo89] F.L. Bookstein. Principal warps : thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6) :567–585, 1989.

[BRPP14] Nicolas Bonneel, Julien Rabin, Gabriel Peyre, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 2014.

[CHU] Chu poitiers : techniques-histologiques. https://www.chu-poitiers.fr/specialites/plateforme-recherche-morphologique/techniques-histologiques/.

[CHY+21] Jie Cao, Luanxuan Hou, Ming-Hsuan Yang, Ran He, and Zhenan Sun. Remix : Towards image-to-image translation with limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15018–15027, June 2021.

[CMK+14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4) :303–314, December 1989.

[CZM+19] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment : Learning augmentation strategies from data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019.

[CZSL20a] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment : Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3008–3017, 2020.

[CZSL20b] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment : Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[DAC19] Neofytos Dimitriou, Ognjen Arandjelović, and Peter D. Caie. Deep learning for whole slide image analysis : An overview. *Frontiers in Medicine*, 6 :264, 2019.

[dBHS+18] Thomas de Bel, Meyke Hermsen, Bart Smeets, Luuk Hilbrands, Jeroen van der Laak, and Geert Litjens. Automatic segmentation of histopathological slides of renal tissue using deep learning. In John E. Tomaszewski

and Metin N. Gurcan, editors, *Medical Imaging 2018 : Digital Pathology*, volume 10581, pages 285 – 290. International Society for Optics and Photonics, SPIE, 2018.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Det18] Nicki S. Detlefsen. libcpab. https://github.com/SkafteNicki/libcpab, 2018.

[DFH18] N. S. Detlefsen, O. Freifeld, and S. Hauberg. Deep diffeomorphic transformer networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4403–4412, 2018.

[DLX+21] Richard C. Davis, Xiang Li, Yuemei Xu, Zehan Wang, Nao Souma, Gina Sotolongo, Jonathan Bell, Matthew Ellis, David Howell, Xiling Shen, Kyle Lafata, and Laura Barisoni. Deep learning segmentation of glomeruli on kidney donor frozen sections. *medRxiv*, 2021.

[DN21] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[DQX+17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.

[DSK16] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

[EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM.

[EL99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, 1999.

[EMP+02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling : A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002.

[FB20] Nikolaus Fortelny and Christoph Bock. Knowledge-primed neural networks enable biologically interpretable deep learning on single-cell sequencing data. *Genome biology*, 21(1), 2020.

[FHBF15] O. Freifeld, S. Hauberg, K. Batmanghelich, and J. W. Fisher. Highly-expressive spaces of well-behaved transformations : Keeping it simple. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2911–2919, 2015.

[FHBF17] O. Freifeld, S. Hauberg, K. Batmanghelich, and J. W. Fisher. Transformations based on continuous piecewise-affine velocity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12) :2496–2509, 2017.

[FvdLL21] Khrystyna Faryna, Jeroen van der Laak, and Geert Litjens. Tailoring automated data augmentation to H&E-stained histopathology. In *Proc. Conference on Medical Imaging with Deep Learning*, volume 143 of *Proc. Machine Learning Research*, pages 168–178, 2021.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[gau] gaugan. https://keras.io/examples/generative/gaugan/.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[GDK+19] Michael Gadermayr, Ann-Kathrin Dombrowski, Barbara Mara Klinkhammer, Peter Boor, and Dorit Merhof. Cnn cascades for segmenting sparse objects in gigapixel whole slide images. *Computerized Medical Imaging and Graphics*, 71 :40–48, Jan 2019.

[GEB15] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[GEB16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[GEB+17] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3730–3738, 2017.

[GGM11] B. Galerne, Y. Gousseau, and J. M. Morel. Random phase textures : Theory and synthesis. *IEEE Transactions on Image Processing*, 20(1) :257–267, Jan 2011.

[Gir15] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[Glo] Glomérule, wikipédia. https://commons.wikimedia.org/wiki/File:Renal_corpuscle-en.svg.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[GPL+18] Jaime Gallego, Anibal Pedraza, Samuel Lopez, Georg Steiner, Lucia Gonzalez, Arvydas Laurinavicius, and Gloria Bueno. Glomerulus classification and detection based on convolutional neural networks. *Journal of Imaging*, 4(1), 2018.

[GSDC17] Geoffrey Guingo, Basile Sauvage, Jean-Michel Dischler, and Marie-Paule Cani. Bi-Layer textures : a Model for Synthesis and Deformation of Composite Textures. *Computer Graphics Forum*, 36(4) :111–122, 2017.

[GSV+14] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Local random-phase noise for procedural texturing. *ACM Trans. Graph.*, 33(6) :195 :1–195 :11, November 2014.

[GZM21] Yunhe Gao, Mu Zhou, and Dimitris N Metaxas. Utnet : a hybrid transformer architecture for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 61–71. Springer, 2021.

[HAS] Haute autorité de santé : greffe rénale, assurer un accès équitable à la liste d'attente. https://www.has-sante.fr/jcms/c_2576220/fr/greffe-renale-assurer-un-acces-equitable-a-la-liste-d-attente.

[HAS+17] Le Hou, Ayush Agarwal, Dimitris Samaras, Tahsin M. Kurç, Rajarsi R. Gupta, and Joel H. Saltz. Unsupervised histopathology image synthesis. *ArXiv*, abs/1712.05021, 2017.

[HB95] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings., International Conference on Image Processing*, volume 3, pages 648–651 vol.3, Oct 1995.

[HB17] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[HdBdB+19] Meyke Hermsen, Thomas de Bel, Marjolijn den Boer, Eric J. Steenbergen, Jesper Kers, Sandrine Florquin, Joris J. T. H. Roelofs, Mark D. Stegall, Mariam P. Alexander, Byron H. Smith, Bart Smeets, Luuk B. Hilbrands, and Jeroen A. W. M. van der Laak. Deep Learning–Based Histopathologic Assessment of Kidney Tissue. *Journal of the American Society of Nephrology*, 30(10) :1968–1979, 2019.

[HGZ21] Dongjin Huang, Hao Guo, and Yue Zhang. Add-net :attention u-net with dilated skip connection and dense connected decoder for retinal vessel segmentation. In Nadia Magnenat-Thalmann, Victoria Interrante, Daniel Thalmann, George Papagiannakis, Bin Sheng, Jinman Kim, and Marina Gavrilova, editors, *Advances in Computer Graphics*, pages 327–338, Cham, 2021. Springer International Publishing.

[His] Histopathologie, wikipédia. https://fr.wikipedia.org/wiki/Histopathologie.

[Hul20] Tim Hulsen. Sharing is caring-data sharing initiatives in healthcare. *Int J Environ Res Public Health*, 2020.

[HVCB21] Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9412–9420, June 2021.

[HWW+20] Robin Heckenauer, Jonathan Weber, Cedric Wemmert, Friedrich Feuerhake, Michel Hassenforder, Pierre-Alain Muller, and Germain Forestier. Real-time detection of glomeruli in renal pathology. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 350–355, 2020.

[HZRS16] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[JAFF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

[JBV16] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *CoRR*, abs/1611.08207, 2016.

[JMP+19] Qiangguo Jin, Zhaopeng Meng, Tuan D. Pham, Qi Chen, Leyi Wei, and Ran Su. DUNet : A deformable network for retinal vessel segmentation. *Knowledge-Based Systems*, 2019.

[JSZK15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2017–2025, Cambridge, MA, USA, 2015. MIT Press.

[Jun] Kim Junho. Spade-tensorflow. https://github.com/taki0112/SPADE-Tensorflow.

[KAH+20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. Advances in Neural Information Processing Systems*, volume 33, pages 12104–12114, 2020.

[KALL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[KDC17] M. Kolar, K. Debattista, and A. Chalmers. A subjective evaluation of texture synthesis methods. *Computer Graphics Forum*, 36(2) :189–198, 2017.

[KEBK05] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

[KFN+21] Jogile Kuklyte, Jenny Fitzgerald, Sophie Nelissen, Haolin Wei, Aoife Whelan, Adam Power, Ajaz Ahmad, Martyna Miarka, Mark Gregson, Michael Maxwell, Ruka Raji, Joseph Lenihan, Eve Finn-Moloney, Mairin Rafferty, Maurice Cary, Erio Barale-Thomas, and Donal O'Shea. Evaluation of the Use of Single- and Multi-Magnification Convolutional Neural Networks for the Determination and Quantitation of Lesions in Nonclinical Pathology Studies. *Toxicologic Pathology*, 49(4) :815–842, 2021. PMID : 33618634.

[KI18] Daisuke Komura and Shumpei Ishikawa. Machine learning methods for histopathological image analysis. *Computational and Structural Biotechnology Journal*, 16 :34–42, 2018.

[KJC16] Angjoo Kanazawa, David W. Jacobs, and Manmohan Chandraker. Warpnet : Weakly supervised matching for single-view reconstruction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3253–3261, 2016.

[KLA+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[KML+19] Shruti Kannan, Laura A. Morgan, Benjamin Liang, McKenzie G. Cheung, Christopher Q. Lin, Dan Mun, Ralph G. Nader, Mostafa E. Belghasem, Joel M. Henderson, Jean M. Francis, Vipul C. Chitalia, and Vijaya B. Kolachalama. Segmentation of glomeruli within trichrome images using deep learning. *Kidney International Reports*, 4(7) :955–962, 2019.

[Kri12] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[KRP+14] Anna Korzynska, Lukasz Roszkowiak, Dorota Pijanowska, Wojciech Kozlowski, and Tomasz Markiewicz. The influence of the microscope lamp filament colour temperature on the process of digital images of histological slides acquisition standardization. *Diagnostic Pathology*, 9(1) :S13, Dec 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[KW14] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http ://yann.lecun.com/exdb/mnist/, 2010.

[LCC+19] Jun Li, Yongjun Chen, Lei Cai, Ian Davidson, and Shuiwang Ji. Dense transformer networks for brain electron microscopy image segmentation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2894–2900. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[LDG+17] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[LDX+21] Xiang Li, Richard C. Davis, Yuemei Xu, Zehan Wang, Nao Souma, Gina Sotolongo, Jonathan Bell, Matthew Ellis, David Howell, Xiling Shen, Kyle J. Lafata, and Laura Barisoni. Deep learning segmentation of glomeruli on kidney donor frozen sections. *Journal of Medical Imaging*, 8(6) :1 – 18, 2021.

[Lef14] Sylvain Lefebvre. *Synhtèse de textures par l'exemple pour les applications interactives*. Habilitation à diriger des recherches, Université de Lorraine, Juin 2014. 100 pages.

[LH15] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015.

[LKB+17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42 :60–88, 2017.

[LKK+19] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[LLC+10] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S. Ebert, John P. Lewis, Ken Perlin, and Matthias Zwicker. State of the art in procedural noise functions. In Helwig Hauser and Erik Reinhard, editors, *EG 2010 - State of the Art Reports*, Norrkoping, Sweden, May 2010. Eurographics, Eurographics Association.

[LLY+21] Chaohui Li, Yingjian Liu, Haoyu Yin, Yue Li, Qingxiang Guo, Limin Zhang, and Pengting Du. Attention residual u-net for building segmentation in aerial images. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 4047–4050, 2021.

[LMS+19] Thomas Lampert, Odyssée Merveille, Jessica Schmitz, Germain Forestier, Friedrich Feuerhake, and Cédric Wemmert. Strategies for training stain invariant cnns. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 905–909, 2019.

[LN89] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1) :503–528, Aug 1989.

[LPF+20] Adrian B Levine, Jason Peng, David Farnell, Mitchell Nursey, Yiping Wang, Julia R Naso, Hezhen Ren, Hossein Farahani, Colin Chen, Derek Chiu, Aline Talhouk, Brandon Sheffield, Maziar Riazy, Philip P Ip, Carlos Parra-Herran, Anne Mills, Naveena Singh, Basile Tessier-Cloutier, Taylor Salisbury, Jonathan Lee, Tim Salcudean, Steven JM Jones, David G Huntsman, C Blake Gilks, Stephen Yip, and Ali Bashashati. Synthesis of diagnostic quality cancer pathology images by generative adversarial networks. *The Journal of Pathology*, 252(2) :178–188, 2020.

[LPP+19] M. C. H. Lee, K. Petersen, N. Pawlowski, B. Glocker, and M. Schaap. Tetris : Template transformer networks for image segmentation with shape priors. *IEEE Transactions on Medical Imaging*, 38(11) :2596–2606, 2019.

[LPZQ20] Ziqiang Li, Hong Pan, Yaping Zhu, and A. K. Qin. Pgd-unet : A position-guided deformable network for simultaneous segmentation of organs and tumors. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[LSA+16] Y. D. Lockerman, B Sauvage, R Allègre, JM Dischler, Julie Dorsey, and Holly Rushmeier. Multi-scale label-map extraction for texture synthesis. *ACM Transactions on Graphics*, 07/2016 2016.

[LSD21] Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. Cyclostationary gaussian noise : theory and synthesis. *Computer Graphics Forum*, 40(2) :239–250, 2021.

[LST+16] G. Litjens, C. I. Sánchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen-van de Kaa, P. Bult, B. van Ginneken, and J. van der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, 6 :26286, 2016.

[LTC+20] Chen Li, Yusong Tan, Wei Chen, Xin Luo, Yuanming Gao, Xiaogang Jia, and Zhiying Wang. Attention unet++ : A nested attention-aware u-net for liver ct image segmentation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 345–349, 2020.

[LWHC19] Robin Liu, Lu Wang, Jim He, and Wenfang Chen. Towards staining independent segmentation of glomerulus from histopathological images of kidney. *bioRxiv*, 2019.

[MAdMB+20] Steve Tsham Mpinda Ataky, Jonathan de Matos, Alceu de S. Britto, Luiz E. S. Oliveira, and Alessandro L. Koerich. Data augmentation for histopathological images based on gaussian-laplacian pyramid blending. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[MAKM22] Abdulrahman Abbas Mukhlif, Belal Al-Khateeb, and Mazin Abed Mohammed. An extensive review of state-of-the-art transfer learning techniques used in medical imaging : Open issues and challenges. *Journal of Intelligent Systems*, 31(1) :1085–1111, 2022.

[MBC+20] Faisal Mahmood, Daniel Borders, Richard J. Chen, Gregory N. Mckay, Kevan J. Salimian, Alexander Baras, and Nicholas J. Durr. Deep adversarial training for multi-organ nuclei segmentation in histopathology images. *IEEE Transactions on Medical Imaging*, 39(11) :3257–3267, 2020.

[MJH+17] Joep Moritz, Stuart James, Tom S. F. Haines, Tobias Ritschel, and Tim Weyrich. Texture stationarization : Turning photos into tileable textures. *Computer Graphics Forum (Proc. Eurographics)*, 36(2) :1–13, 2017.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529–533, February 2015.

[MLG+20] Leema Krishna Murali, Brendon Lutnick, Brandon Ginley, John E. Tomaszewski, and Pinaki Sarder. Generative modeling for renal microanatomy. *Proc. SPIE–the International Society for Optical Engineering*, 11320 :113200F, 2020.

[MLS+21] Odyssee Merveille, Thomas Lampert, Jessica Schmitz, Germain Forestier, Friedrich Feuerhake, and Cédric Wemmert. An automatic framework for fusing information from differently stained consecutive digital whole slide images : A case study in renal histology. *Computer Methods and Programs in Biomedicine*, 208 :106157, 2021.

[MLT+19] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[MPPS16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016.

[MPS+19] Alfonso Medela, Artzai Picon, Cristina L. Saratxaga, Oihana Belar, Virginia Cabezón, Riccardo Cicchi, Roberto Bilbao, and Ben Glover. Few shot learning in histopathological images :reducing the need of labeled data on biological datasets. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1860–1864, 2019.

[MRS+16] Raphaël Marée, Loïc Rollus, Benjamin Stévens, Renaud Hoyoux, Gilles Louppe, Rémy Vandaele, Jean-Michel Begon, Philipp Kainz, Pierre Geurts, and Louis Wehenkel. Collaborative analysis of multi-gigapixel imaging data using cytomine. *Bioinformatics*, 32(9) :1395–1401, 01 2016.

[MW47] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1) :50 – 60, 1947.

[NMK+06] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4) :809–836, 2006.

[NMP+19] Jakub Nalepa, Grzegorz Mrukwa, Szymon Piechaczek, Pablo Ribalta Lorenzo, Michal Marcinkiewicz, Barbara Bobek-Billewicz, Pawel Wawrzyniak, Pawel Ulrych, Janusz Szymanek, Marcin Cwiek, Wojciech Dudzik, Michal Kawulok, and Michael P. Hayball. Data augmentation via image registration. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4250–4254, 2019.

[NS21] R. H. Naik and S. H. Shawar. *Renal Transplantation Rejection*. Treasure Island (FL) : StatPearls Publishing, Jan 2021. PMID : 31971715.

[OSF+18] Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net : Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018.

[PA20] Narinder Singh Punn and Sonali Agarwal. Inception u-net architecture for semantic segmentation to identify nuclei in microscopy cell images. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(1), feb 2020.

[PCR11] Tania Pouli, Douglas W. Cunningham, and Erik Reinhard. A survey of image statistics relevant to computer graphics. *Computer Graphics Forum*, 30(6) :1761–1788, 2011.

[PCSD20] D. H. Pak, A. Caballero, W. Sun, and J. S. Duncan. Efficient aortic valve multilabel segmentation using a spatial transformer network. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1738–1742, 2020.

[Per85] Ken Perlin. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 287–296, New York, NY, USA, 1985. Association for Computing Machinery.

[pix] pix2pix. https://www.tensorflow.org/tutorials/generative/pix2pix.

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016.

[PLWZ19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Pol] Pôle meca tech : Cytomine, de l'open source pour mieux s'intégrer. https://www.polemecatech.be/fr/news/cytomine-de-lopen-source-pour-mieux-sintegrer/.

[PS00] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1) :49–70, Oct 2000.

[PTD+22] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Brain imaging generation with latent diffusion models, 2022.

[PTR+21] Olivier Petit, Nicolas Thome, Clement Rambour, Loic Themyr, Toby Collins, and Luc Soler. U-Net Transformer : Self and Cross Attention for Medical Image Segmentation. In *MICCAI workshop MLMI*, Strasbourg (virtuel), France, September 2021.

[QMSY20] Adalberto Claudio Quiros, Roderick Murray-Smith, and Ke Yuan. Pathologygan : Learning deep representations of cancer tissue. In *Proc. Conference on Medical Imaging with Deep Learning*, volume 121 of *Proc. Machine Learning Research*, pages 669–695, 2020.

[RBL+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[RcDDM17] Lara Raad cisa, AXEL DAVY, Agnès Desolneux, and Jean-Michel Morel. A survey of exemplar-based texture synthesis. working paper or preprint, 2017.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[RHGS15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. International Conference on Learning Representations*, 2016.

[Sau18] Basile Sauvage. *Contributions à l'analyse et à la synthèse de l'apparence d'objets 3D numériques*. Habilitation à diriger des recherches, Université de Strasbourg, Juillet 2018. 83 pages.

[SBCH11] Darshan Siddalinga Swamy, Kellen J. Butler, Damon M. Chandler, and Sheila S. Hemami. Parametric quality assessment of synthesized textures. *Proc.SPIE*, 7865 :7865 – 7865 – 9, 2011.

[SCM21] Chetan L. Srinidhi, Ozan Ciga, and Anne L. Martel. Deep neural network models for computational histopathology : A survey. *Medical Image Analysis*, 67 :101813, 2021.

[SCO17] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Transactions on graphics (TOG)*, 2017.

[SK16] Tim Salimans and Durk P Kingma. Weight normalization : A simple reparameterization to accelerate training of deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1) :60, Jul 2019.

[SMG+21] Massimo Salvi, Alessandro Mogetta, Alessandro Gambella, Luca Molinaro, Antonella Barreca, Mauro Papotti, and Filippo Molinari. Automated assessment of glomerulosclerosis and tubular atrophy using deep learning. *Computerized Medical Imaging and Graphics*, 90 :101930, 2021.

[SMW06] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3) :533–540, July 2006.

[Sne] Xavier Snelgrove. multiscale-neural-synthesis. https://wxs.ca/research/multiscale-neural-synthesis/.

[Sne17] Xavier Snelgrove. High-resolution multi-scale neural texture synthesis. In *SIGGRAPH ASIA 2017 Technical Briefs*, SA '17, New York, NY, USA, 2017. ACM.

[SNS+18] Caglar Senaras, Muhammad Khalid Khan Niazi, Berkman Sahiner, Michael P. Pennell, Gary Tozbikian, Gerard Lozanski, and Metin N. Gurcan. Optimized generation of high-resolution phantom images using cgan : Application to quantification of ki67 breast cancer images. *PLOS ONE*, 13(5) :1–12, 05 2018.

[SSP03] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963, 2003.

[SWL+16] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4168–4176, 2016.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[SZG+21] Pourya Shamsolmoali, Masoumeh Zareapoor, Eric Granger, Huiyu Zhou, Ruili Wang, M. Emre Celebi, and Jie Yang. Image synthesis with adversarial networks : A comprehensive survey and case studies. *Information Fusion*, 72 :126–146, 2021.

[Tel04] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9, 01 2004.

[TJL+21] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7921–7931, 2021.

[TLB+19] David Tellez, Geert Litjens, Péter Bándi, Wouter Bulten, John-Melle Bokhorst, Francesco Ciompi, and Jeroen van der Laak. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical Image Analysis*, 58 :101544, 2019.

[TOFS+17] M. Temerinac-Ott, G. Forestier, J. Schmitz, M. Hermsen, J.H. Bräsen, F. Feuerhake, and C. Wemmert. Detection of glomeruli in renal pathology by mutual comparison of multiple staining modalities. In *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, pages 19–24, 2017.

[TOG20] Maximilian E. Tschuchnig, Gertie J. Oostingh, and Michael Gadermayr. Generative adversarial networks in digital pathology : A survey on trends and future potential. *Patterns*, 1(6) :100089, 2020.

[TPA+18] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data : Bridging the reality gap by domain randomization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1082–10828, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[ULVL16] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks : Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1349–1357. JMLR.org, 2016.

[UVL16] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization : The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[VEFD19] Yves-Rémi Van Eycke, Adrien Foucart, and Christine Decaestecker. Strategies to reduce the expert supervision required for deep learning-based segmentation of histopathological images. *Frontiers in Medicine*, 6 :222, 2019.

[VFWL21a] Jelica Vasiljević, Friedrich Feuerhake, Cédric Wemmert, and Thomas Lampert. Self adversarial attack as an augmentation method for immunohistochemical stainings. *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, 2021.

[VFWL21b] Jelica Vasiljević, Friedrich Feuerhake, Cédric Wemmert, and Thomas Lampert. Towards histopathological stain invariance by unsupervised domain augmentation using generative adversarial networks. *Neurocomputing*, 460 :277–291, 2021.

[VK20] Arash Vahdat and Jan Kautz. Nvae : A deep hierarchical variational autoencoder. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19667–19679. Curran Associates, Inc., 2020.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[Wan17] Zirui Wang. guided-neural-style. https://github.com/wzirui/guided-neural-style, 2017.

[WFE+17] Thomas S. A. Wallis, Christina M. Funke, Alexander S. Ecker, Leon A. Gatys, Felix A. Wichmann, and Matthias Bethge. A parametric texture model based on deep convolutional features closely matches texture appearance for humans. *Journal of Vision*, 17(12) :5, 2017.

[Wil45] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6) :80–83, 1945.

[WL19] Monan Wang and Pengcheng Li. A review of deformation models in medical image registration. *Journal of Medical and Biological Engineering*, 39(1) :1–17, Feb 2019.

[WLKT09] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009.

[WLZ+18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[WRB17] Pierre Wilmot, Eric Risser, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *CoRR*, abs/1701.08893, 2017.

[WSV+20] Jerry Wei, Arief Suriawinata, Louis Vaickus, Bing Ren, Xiaoying Liu, Jason Wei, and Saeed Hassanpour. Generative Image Translation for Data Augmentation in Colorectal Histopathology Images. In Adrian V. Dalca, Matthew B.A. McDermott, Emily Alsentzer, Samuel G. Finlayson, Michael Oberst, Fabian Falck, and Brett Beaulieu-Jones, editors, *Proceedings of the Machine Learning for Health NeurIPS Workshop*, volume 116 of *Proceedings of Machine Learning Research*, pages 10–24. PMLR, 13 Dec 2020.

[WWFF21] Cédric Wemmert, Jonathan Weber, Friedrich Feuerhake, and Germain Forestier. *Deep Learning for Histopathological Image Analysis*, pages 153–169. 03 2021.

[WYKN20] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples : A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), jun 2020.

[WYW+18] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan : Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[XDVF+19] Yang Xiao, Etienne Decencière, Santiago Velasco-Forero, Hélène Burdin, Thomas Bornschlögl, Françoise Bernerd, Emilie Warrick, and Thérèse Baldeweck. A new color augmentation method for deep learning segmentation of histological images. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 886–890, 2019.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks ? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[YSH18] Wei Yi, Yaoran Sun, and Sailing He. Data augmentation using conditional gans for facial emotion recognition. In *2018 Progress in Electromagnetics Research Symposium (PIERS-Toyama)*, pages 710–714, 2018.

[YTT19] M. Mutlu Yapici, Adem Tekerek, and Nurettin Topaloğlu. Literature review of deep learning research areas. *Gazi Journal of Engineering Sciences*, pages 188 – 215, 2019.

[YWB19] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging : A review. *Medical Image Analysis*, 58 :101552, 2019.

[ZLXL20] M. Zhang, X. Li, M. Xu, and Q. Li. Automated semantic segmentation of red blood cells for sickle cell disease. *IEEE Journal of Biomedical and Health Informatics*, 24(11) :3095–3102, 2020.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[ZRSTL18] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++ : A nested u-net architecture for medical image segmentation. In Danail Stoyanov, Zeike Taylor, Gustavo Carneiro, Tanveer Syeda-Mahmood, Anne Martel, Lena Maier-Hein, João Manuel R.S. Tavares, Andrew Bradley, João Paulo Papa, Vasileios Belagiannis, Jacinto C. Nascimento, Zhi Lu, Sailesh Conjeti, Mehdi Moradi, Hayit Greenspan, and Anant Madabhushi, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, Cham, 2018. Springer International Publishing.

[ZSTL20] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++ : Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*, 39(6) :1856–1867, 2020.

[ZWZZ19] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *Proc. International Conference on Learning Representations*, 2019.

[ZZB+18] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4), 2018.

[ZZP+17] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Annexe A

Annexes

Dans cette annexe :

- Nous donnons les graphiques additionnels montrant les propriétés de champs de déplacement générés aléatoirement pour les modèles de déformation considérés dans le manuscrit.
- Nous donnons les résultats de segmentation complets de chaque chapitre du manuscrit.
- Nous donnons les p-valeurs associées aux expérimentations décrites dans le manuscrit.

A.1 Graphiques additionnels

A.1.1 *Random Displacement Fields*

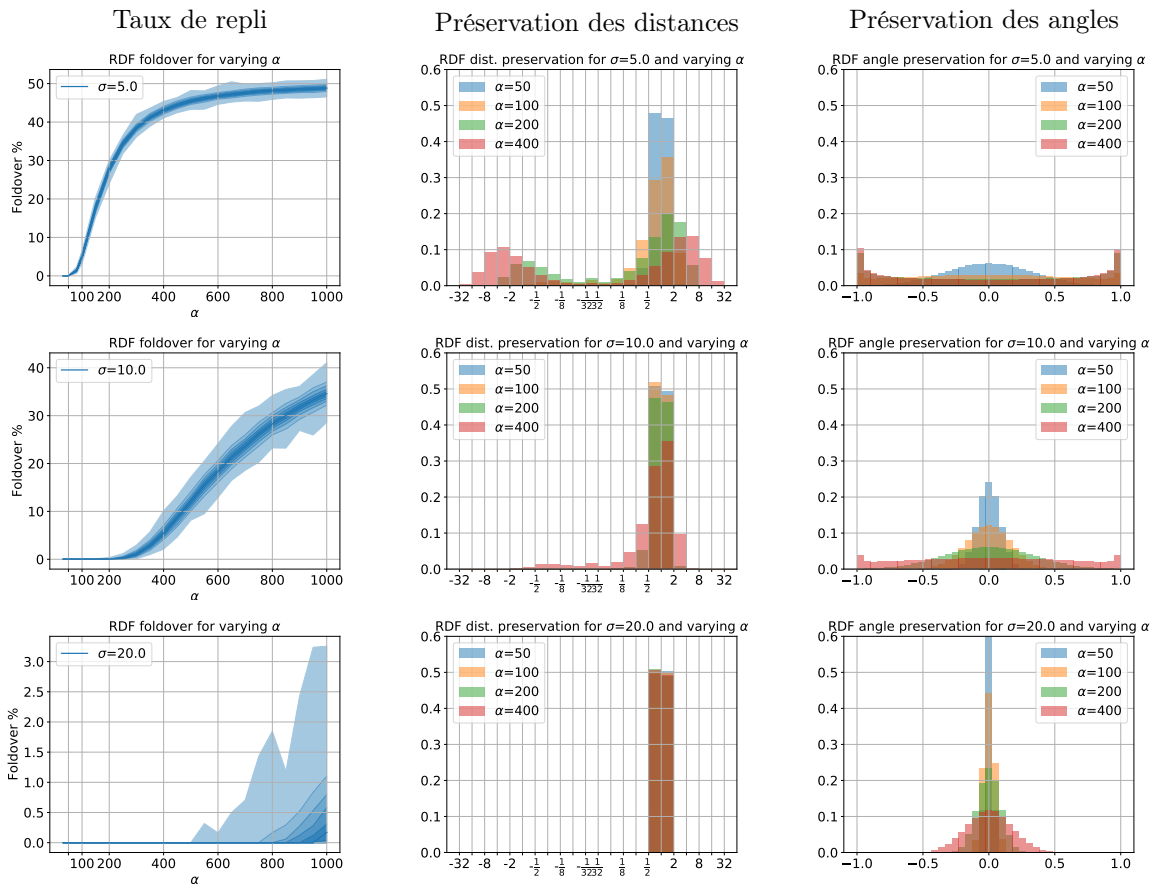


Figure A.1 : Taux de repli et préservation des distances/angles pour les *Random Displacement Fields*.

A.1.2 Déformations basées sur une grille avec interpolation à l'ordre 2 (TPS)

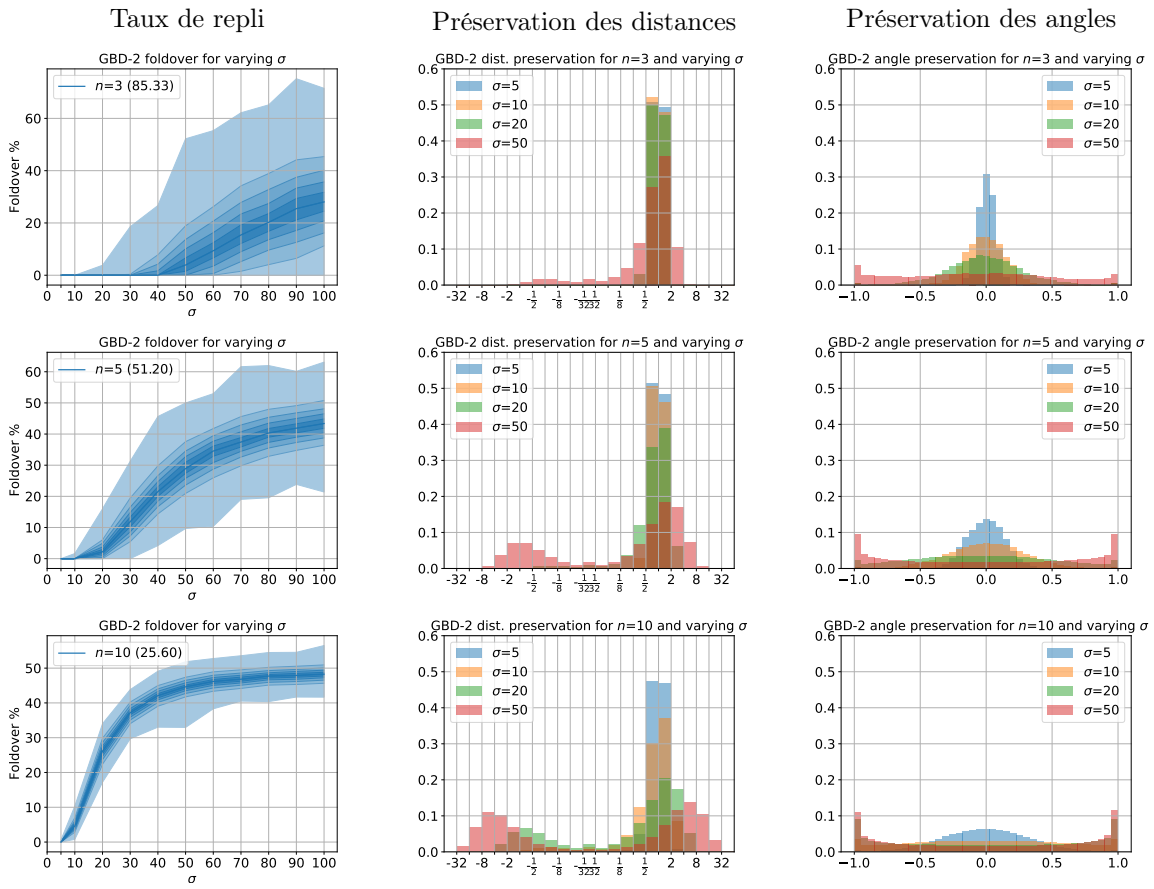


Figure A.2 : Taux de repli et préservation des distances/angles pour les déformations basées sur une grille avec interpolation à l'ordre 2 (TPS).

A.1.3 Déformations basées sur une grille avec interpolation à l'ordre 3

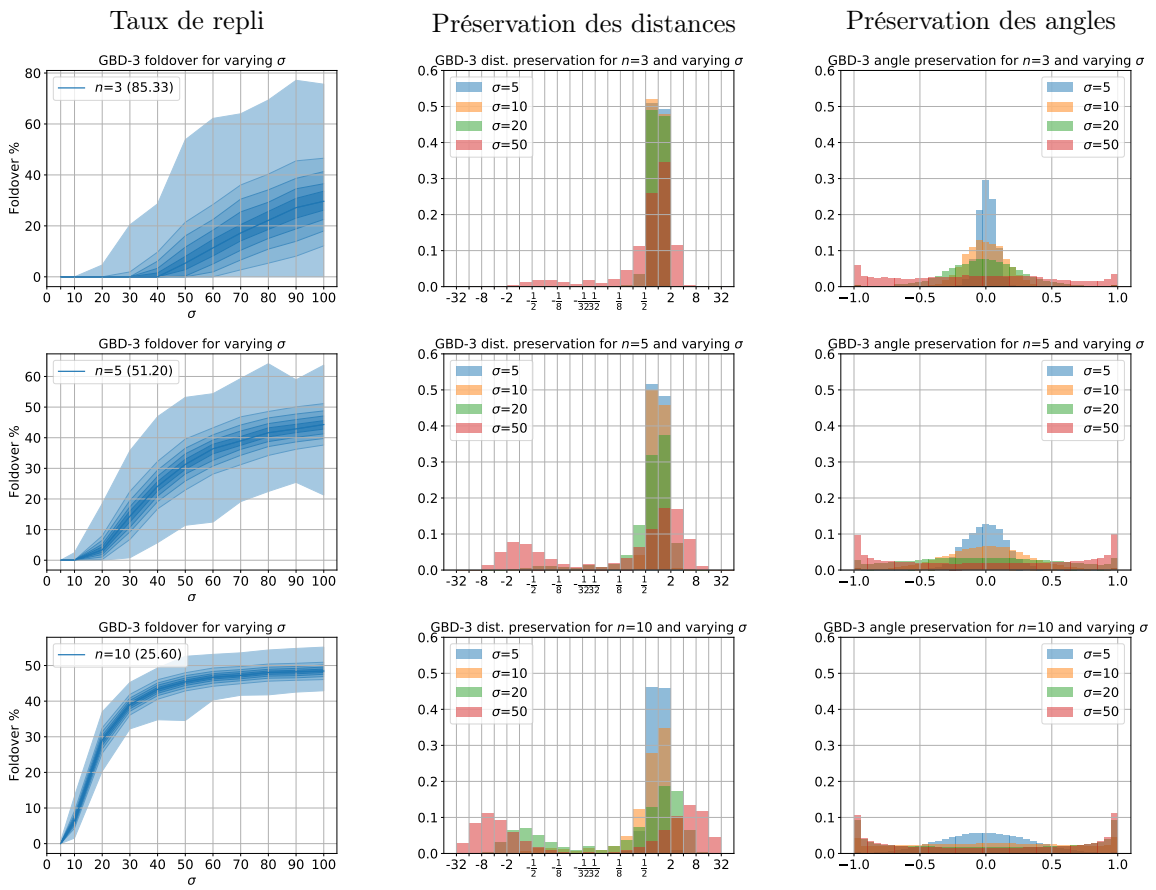
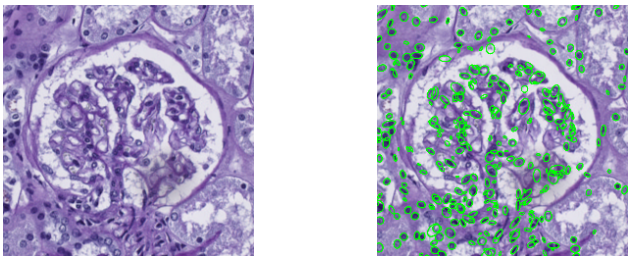


Figure A.3 : Taux de repli et préservation des distances/angles pour les déformations basées sur une grille avec interpolation à l'ordre 3.

A.1.4 MVN MLS-ARAP basé sur les noyaux de cellules



(a) (b)

Figure A.4 : Centres des noyaux de cellules détectés et utilisés pour le calcul des statistiques de la figure A.5. (a) Patch WSI d'entrée. (b) Noyaux détectés ($K = 310$).

Taux de repli Préservation des distances Préservation des angles

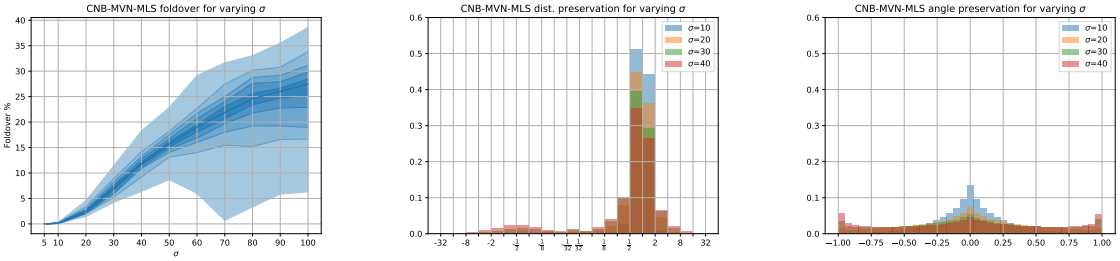
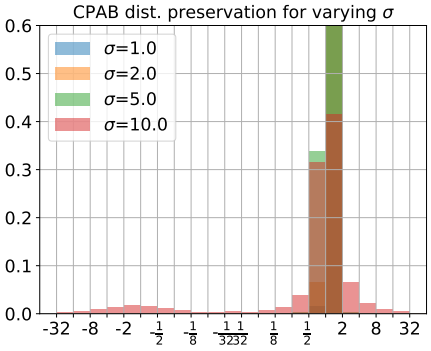


Figure A.5 : Taux de repli et préservation des distances/angles pour MVN MLS-ARAP basé sur les noyaux de cellules, calculé pour les noyaux de cellules détectés sur la figure A.4.

A.1.5 Déformations CPAB

Préservation des distances



Préservation des angles

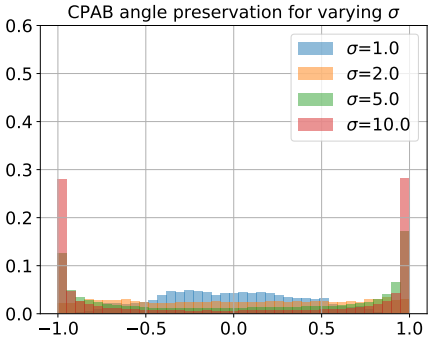


Figure A.6 : Préservation des distances/angles pour les déformations CPAB, avec $K = 100$.

A.1.6 Déformations basées FBM

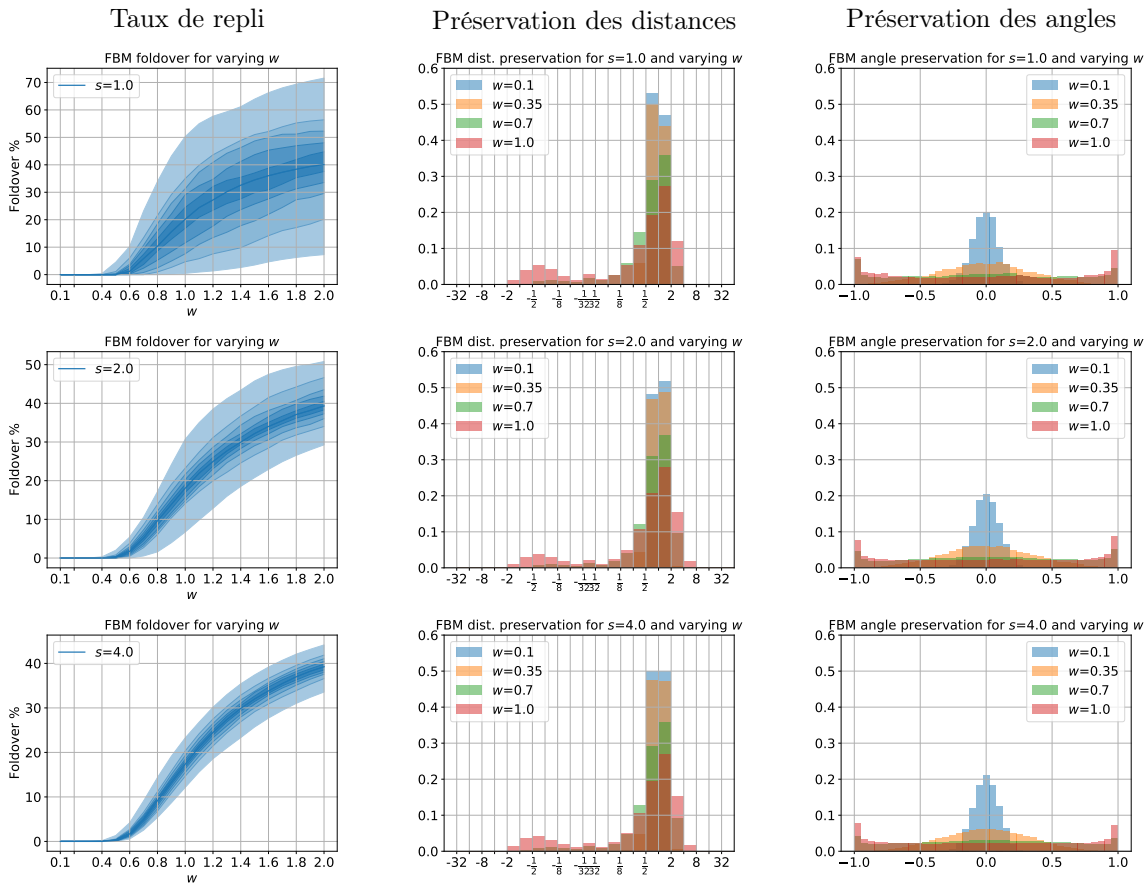


Figure A.7 : Taux de repli et préservation des distances/angles pour les déformations FBM, implémentées avec du bruit de Perlin.

A.2 Résultats de segmentation complets du chapitre 3

A.2.1 Baseline

- Indice de Dice : $62{,}31 \pm 4{,}02$
- Précision : $90{,}69 \pm 2{,}10$
- Recul : $47{,}71 \pm 5{,}63$
- Spécificité : $99{,}20 \pm 0{,}31$

A.2.2 *Random Displacement Fields*

σ \ α	100	200	400	800
5	$75{,}86 \pm 2{,}33$	$72{,}75 \pm 3{,}87$	$53{,}42 \pm 3{,}99$	$48{,}07 \pm 4{,}48$
10	$72{,}50 \pm 5{,}23$	$77{,}48 \pm 1{,}84$	$78{,}92 \pm 3{,}70$	$77{,}87 \pm 1{,}75$
20	$66{,}53 \pm 5{,}19$	$73{,}67 \pm 4{,}81$	$77{,}29 \pm 2{,}92$	$78{,}59 \pm 2{,}56$

σ \ α	2400	5000
40	$75{,}54 \pm 2{,}92$	$68{,}99 \pm 2{,}23$

(a) Indice de Dice en fonction des paramètres de la méthode RDF.

σ \ α	100	200	400	800
5	$92{,}46 \pm 1{,}93$	$92{,}82 \pm 2{,}54$	$89{,}54 \pm 8{,}60$	$87{,}45 \pm 1{,}88$
10	$90{,}04 \pm 1{,}86$	$91{,}85 \pm 2{,}09$	$91{,}41 \pm 0{,}77$	$86{,}18 \pm 2{,}88$
20	$92{,}18 \pm 2{,}59$	$91{,}87 \pm 1{,}48$	$92{,}11 \pm 1{,}69$	$91{,}42 \pm 1{,}96$

σ \ α	2400	5000
40	$91{,}01 \pm 2{,}27$	$86{,}77 \pm 1{,}70$

(b) Précision en fonction des paramètres de la méthode RDF.

σ \ α	100	200	400	800
5	$64{,}36 \pm 3{,}63$	$60{,}12 \pm 6{,}12$	$38{,}27 \pm 3{,}91$	$33{,}33 \pm 4{,}65$
10	$61{,}09 \pm 7{,}60$	$67{,}09 \pm 3{,}19$	$69{,}59 \pm 5{,}87$	$71{,}21 \pm 4{,}10$
20	$52{,}42 \pm 6{,}74$	$61{,}70 \pm 6{,}46$	$66{,}71 \pm 4{,}44$	$69{,}10 \pm 4{,}66$

σ \ α	2400	5000
40	$64{,}81 \pm 5{,}48$	$57{,}38 \pm 3{,}62$

(c) Recul en fonction des paramètres de la méthode RDF.

σ \ α	100	200	400	800
5	$99{,}17 \pm 0{,}20$	$99{,}24 \pm 0{,}37$	$99{,}23 \pm 0{,}73$	$99{,}23 \pm 0{,}25$
10	$98{,}91 \pm 0{,}33$	$99{,}05 \pm 0{,}30$	$98{,}97 \pm 0{,}17$	$98{,}17 \pm 0{,}54$
20	$99{,}27 \pm 0{,}33$	$99{,}14 \pm 0{,}19$	$99{,}09 \pm 0{,}24$	$98{,}96 \pm 0{,}31$

σ \ α	2400	5000
40	$98{,}97 \pm 0{,}39$	$98{,}60 \pm 0{,}28$

(d) Spécificité en fonction des paramètres de la méthode RDF.

Table A.5 : Résultats en fonction des paramètres de la méthode RDF.

A.2.3 Déformations basées sur une grille avec interpolation à l'ordre 2 (TPS)

n \ σ	5	10	20	50
3	78,84 ± 2,24	81,76 ± 3,83	84,34 ± 1,81	85,02 ± 1,58
5	80,14 ± 1,16	83,20 ± 2,43	85,41 ± 1,25	78,88 ± 3,51
10	80,28 ± 2,02	84,02 ± 3,17	76,48 ± 3,21	71,71 ± 4,73

(a) Indice de Dice en fonction des paramètres de la méthode TPS.

n \ σ	5	10	20	50
3	93,15 ± 2,45	92,95 ± 1,45	91,72 ± 1,38	89,43 ± 1,62
5	93,43 ± 2,00	93,05 ± 1,92	91,50 ± 1,76	88,81 ± 1,67
10	91,80 ± 2,48	93,43 ± 1,12	90,64 ± 3,10	82,48 ± 3,65

(b) Précision en fonction des paramètres de la méthode TPS.

n \ σ	5	10	20	50
3	68,46 ± 3,80	73,15 ± 5,94	78,16 ± 3,72	81,14 ± 3,81
5	70,19 ± 1,51	75,44 ± 5,11	80,16 ± 2,75	71,18 ± 5,88
10	71,44 ± 3,53	76,52 ± 5,50	66,53 ± 6,49	64,23 ± 9,70

(c) Recul en fonction des paramètres de la méthode TPS.

n \ σ	5	10	20	50
3	99,19 ± 0,32	99,14 ± 0,18	98,88 ± 0,24	98,47 ± 0,30
5	99,22 ± 0,25	99,11 ± 0,30	98,87 ± 0,52	98,57 ± 0,31
10	98,98 ± 0,36	99,16 ± 0,17	98,93 ± 0,56	97,76 ± 0,93

(d) Spécificité en fonction des paramètres de la méthode TPS.

Table A.10 : Résultats en fonction des paramètres de la méthode TPS.

A.2.4 Déformations basées sur une grille avec interpolation à l'ordre 3

n \ σ	5	10	20	50	70
3	76,88 ± 1,35	81,77 ± 1,84	85,93 ± 1,62	85,10 ± 1,39	80,25 ± 1,47
5	79,31 ± 2,07	84,07 ± 3,44	81,14 ± 3,19	73,81 ± 5,42	
10	81,13 ± 3,11	84,48 ± 1,65	75,06 ± 3,28	70,11 ± 1,97	

(a) Indice de Dice en fonction des paramètres de la méthode GBD-3.

n \ σ	5	10	20	50	70
3	92,13 ± 1,24	93,06 ± 1,48	91,92 ± 2,60	89,07 ± 1,16	84,61 ± 2,94
5	91,93 ± 2,08	92,16 ± 1,45	92,23 ± 1,78	87,54 ± 3,56	
10	93,86 ± 1,18	92,42 ± 1,05	87,71 ± 2,94	82,03 ± 1,41	

(b) Précision en fonction des paramètres de la méthode GBD-3.

n \ σ	5	10	20	50	70
3	66,02 ± 2,55	72,95 ± 2,32	80,90 ± 4,70	81,51 ± 2,47	76,54 ± 4,44
5	69,85 ± 3,73	77,53 ± 6,14	72,61 ± 5,58	64,49 ± 9,27	
10	71,50 ± 4,31	77,88 ± 3,41	65,98 ± 6,56	61,31 ± 3,29	

(c) Recul en fonction des paramètres de la méthode GBD-3.

n \ σ	5	10	20	50	70
3	99,10 ± 0,19	99,14 ± 0,18	98,85 ± 0,45	98,42 ± 0,20	97,77 ± 0,66
5	99,02 ± 0,31	98,95 ± 0,26	99,02 ± 0,27	98,49 ± 0,69	
10	99,26 ± 0,12	98,98 ± 0,19	98,35 ± 0,33	97,87 ± 0,28	

(d) Spécificité en fonction des paramètres de la méthode GBD-3.

Table A.15 : Résultats en fonction des paramètres de la méthode GBD-3.

A.2.5 MVN MLS-ARAP basé sur les noyaux de cellules

σ	5	10	15	30	100
	82,12 ± 1,16	85,01 ± 1,21	85,25 ± 1,82	83,37 ± 1,61	70,417 ± 2,98

(a) Indice de Dice en fonction des paramètres de la méthode CNB-MVN-MLS.

σ	5	10	15	30	100
	92,73 ± 2,17	91,50 ± 2,25	91,44 ± 1,10	91,94 ± 1,13	90,19 ± 0,97

(b) Précision en fonction des paramètres de la méthode CNB-MVN-MLS.

σ	5	10	15	30	100
	73,77 ± 2,80	79,52 ± 3,42	79,93 ± 3,71	76,35 ± 3,28	57,87 ± 4,17

(c) Recul en fonction des paramètres de la méthode CNB-MVN-MLS.

σ	5	10	15	30	100
	99,07 ± 0,33	98,82 ± 0,37	98,81 ± 0,21	98,94 ± 0,20	98,97 ± 0,18

(d) Spécificité en fonction des paramètres de la méthode CNB-MVN-MLS.

Table A.20 : Résultats en fonction des paramètres de la méthode CNB-MVN-MLS.

A.2.6 Déformations basées FBM

s \ w	0,1	0,35	0,7	1
1	82,30 ± 2,86	85,21 ± 2,39	80,21 ± 2,98	75,79 ± 5,68
2	78,05 ± 4,06	85,18 ± 2,02	82,70 ± 3,45	80,91 ± 3,13
4	76,65 ± 1,66	83,91 ± 1,89	84,48 ± 2,73	80,27 ± 3,29

(a) Indice de Dice en fonction des paramètres de la méthode FBM.

s \ w	0,1	0,35	0,7	1
1	93,52 ± 2,03	92,00 ± 2,39	90,66 ± 2,38	88,96 ± 2,12
2	94,55 ± 2,64	89,55 ± 2,94	90,49 ± 2,72	88,89 ± 1,36
4	94,27 ± 1,55	93,49 ± 1,63	91,27 ± 2,57	91,54 ± 2,62

(b) Précision en fonction des paramètres de la méthode FBM.

s \ w	0,1	0,35	0,7	1
1	73,64 ± 4,87	79,62 ± 5,76	72,15 ± 5,81	66,71 ± 10,25
2	66,73 ± 6,17	81,48 ± 5,19	76,57 ± 7,33	74,41 ± 5,47
4	64,62 ± 2,53	76,56 ± 2,93	78,95 ± 6,17	71,57 ± 4,57

(c) Recul en fonction des paramètres de la méthode FBM.

s \ w	0,1	0,35	0,7	1
1	99,18 ± 0,29	98,88 ± 0,44	98,81 ± 0,41	98,65 ± 0,50
2	99,37 ± 0,36	98,47 ± 0,54	98,69 ± 0,51	98,53 ± 0,26
4	99,37 ± 0,20	99,07 ± 0,09	98,78 ± 0,48	98,95 ± 0,35

(d) Spécificité en fonction des paramètres de la méthode FBM.

Table A.25 : Résultats en fonction des paramètres de la méthode FBM.

A.2.7 Déformations CPAB

n \ σ	0,5	1	2	5	10
3	78,71 ± 1,93	83,34 ± 4,11	83,99 ± 2,98	81,54 ± 2,22	77,38 ± 3,62
5	81,92 ± 2,49	84,76 ± 1,96	84,64 ± 2,26	81,94 ± 1,36	73,97 ± 3,77
10	81,03 ± 3,61	84,83 ± 1,74	83,93 ± 2,56	77,90 ± 3,03	73,71 ± 1,71
15	82,19 ± 2,88	83,84 ± 1,83			

(a) Indice de Dice en fonction des paramètres de la méthode CPAB.

n \ σ	0,5	1	2	5	10
3	93,40 ± 1,59	91,99 ± 2,42	91,27 ± 2,08	87,41 ± 2,28	83,74 ± 3,74
5	93,94 ± 1,61	91,65 ± 2,03	90,60 ± 2,63	85,52 ± 2,42	83,94 ± 1,40
10	92,90 ± 1,73	91,42 ± 1,76	90,93 ± 1,48	84,67 ± 3,95	81,43 ± 3,14
15	91,20 ± 1,50	91,11 ± 1,57			

(b) Précision en fonction des paramètres de la méthode CPAB.

n \ σ	0,5	1	2	5	10
3	68,09 ± 3,16	76,64 ± 7,75	77,89 ± 4,78	76,49 ± 3,76	72,50 ± 8,13
5	72,80 ± 4,91	78,94 ± 3,69	79,60 ± 4,72	78,67 ± 1,10	66,28 ± 5,66
10	72,13 ± 6,62	79,18 ± 2,76	78,05 ± 4,52	72,48 ± 6,08	67,57 ± 4,34
15	74,86 ± 4,05	77,80 ± 4,32			

(c) Recul en fonction des paramètres de la méthode CPAB.

n \ σ	0,5	1	2	5	10
3	99,23 ± 0,22	98,91 ± 0,42	98,82 ± 0,30	98,25 ± 0,37	97,71 ± 0,86
5	99,24 ± 0,27	98,85 ± 0,33	98,67 ± 0,44	97,89 ± 0,41	98,00 ± 0,22
10	99,11 ± 0,32	98,84 ± 0,28	98,77 ± 0,24	97,88 ± 0,72	97,53 ± 0,69
15	98,86 ± 0,18	98,79 ± 0,30			

(d) Spécificité en fonction des paramètres de la méthode CPAB.

Table A.30 : Résultats en fonction des paramètres de la méthode CPAB.

A.2.8 Résultats en fonction de la taille du jeu d'entraînement et de la probabilité de déformation

p \ N_{train}	20	100	300	600
0	64,58 ± 2,54	70,98 ± 4,19	83,91 ± 1,98	88,13 ± 1,45
0,25	68,51 ± 3,69	76,02 ± 2,23	88,45 ± 0,59	90,36 ± 1,33
0,5	70,50 ± 2,28	76,31 ± 3,55	88,96 ± 0,55	89,65 ± 0,72
0,75	73,03 ± 2,48	78,02 ± 3,92	87,56 ± 2,25	90,75 ± 1,36
1	73,71 ± 1,96	80,05 ± 2,75	87,57 ± 1,64	88,05 ± 2,88

(a) Indice de Dice en fonction de la taille du jeu d'entraînement et de la probabilité de déformation.

p \ N_{train}	20	100	300	600
0	88,05 ± 1,38	93,21 ± 3,32	95,03 ± 0,76	95,33 ± 1,19
0,25	85,15 ± 1,95	92,42 ± 0,97	94,41 ± 0,48	95,46 ± 0,79
0,5	84,65 ± 1,31	91,65 ± 1,50	93,46 ± 1,14	95,51 ± 1,24
0,75	83,25 ± 3,00	90,84 ± 1,73	93,95 ± 1,81	94,50 ± 0,82
1	84,40 ± 1,76	87,35 ± 1,86	93,92 ± 1,27	95,20 ± 1,22

(b) Précision en fonction de la taille du jeu d'entraînement et de la probabilité de déformation.

p \ N_{train}	20	100	300	600
0	51,09 ± 3,54	57,87 ± 6,07	75,19 ± 3,57	82,02 ± 3,39
0,25	57,63 ± 6,20	64,65 ± 3,55	83,21 ± 1,24	85,84 ± 2,94
0,5	60,54 ± 4,07	65,60 ± 5,69	84,88 ± 1,01	84,49 ± 1,18
0,75	65,39 ± 5,76	68,68 ± 6,76	82,16 ± 5,00	87,32 ± 2,68
1	65,57 ± 3,96	74,14 ± 5,71	82,21 ± 3,68	82,09 ± 5,62

(c) Recul en fonction de la taille du jeu d'entraînement et de la probabilité de déformation.

p \ N_{train}	20	100	300	600
0	98,90 ± 0,21	99,23 ± 0,25	99,38 ± 0,13	99,36 ± 0,20
0,25	98,39 ± 0,42	99,16 ± 0,16	99,22 ± 0,08	99,35 ± 0,14
0,5	98,25 ± 0,30	99,04 ± 0,25	99,06 ± 0,18	99,37 ± 0,19
0,75	98,28 ± 0,39	98,89 ± 0,33	99,15 ± 0,31	99,20 ± 0,14
1	98,07 ± 0,36	98,28 ± 0,40	99,21 ± 0,35	99,32 ± 0,26

(d) Spécificité en fonction de la taille du jeu d'entraînement et de la probabilité de déformation.

Table A.35 : Résultats en fonction de la taille du jeu d'entraînement et de la probabilité de déformation.

A.2.9 Modèles de déformation : p-valeurs

A.3 p-valeurs des comparaisons d'indice de Dice

	GBD-2 (5,20)	GBD-3 (3,20)	RDF (10,400)	CNB-MVN-MLS (15)	FBM (0,35, 1)	CPAB (10,1)
Baseline	0,004	0,004	0,004	0,004	0,004	0,004

(a) p-valeurs pour le test de H0 : l'indice de Dice de la baseline est plus élevé que l'indice de Dice des méthodes testées (test unilatéral).

	GBD-2 (5,20)	GBD-3 (3,20)	RDF (10,400)	CNB-MVN-MLS (15)	FBM (0,35,1)	CPAB (10,1)
baseline	0,008	0,008	0,008	0,008	0,008	0,008
GBD-2 (5,20)	×	0,690	0,008	1,000	0,841	0.690
GBD-3 (3,20)	×	×	0,008	0,421	0,841	0,421
RDF (10,400)	×	×	×	0,008	0,008	0,008
CNB-MVN-MLS (15)	×	×	×	×	1,0	0,841
FBM (0,35, 1)	×	×	×	×	×	0,841

(b) p-valeurs pour le test de H0 : les deux distributions sont égales (test bilatéral).

(σ, α)	(10,100)	(20,100)	(5,200)	(10,200)	(20,200)	(5,400)	(10,400)	(20,400)	(5,800)	(10,800)	(20,800)	(40,2400)	(40,5000)
(5,100)	0,421	0,008	0,421	0,421	0,421	0,008	0,222	0,421	0,008	0,222	0,310	1,000	0,008
(10,100)	×	0,095	1,000	0,032	0,690	0,008	0,056	0,222	0,008	0,032	0,056	0,690	0,151
(20,100)	×	×	0,032	0,008	0,151	0,016	0,008	0,008	0,008	0,008	0,008	0,008	0,421
(5,200)	×	×	×	0,032	0,690	0,008	0,056	0,151	0,008	0,032	0,056	0,421	0,421
(10,200)	×	×	×	×	0,310	0,008	0,690	0,841	0,008	0,841	0,222	0,310	0,008
(20,200)	×	×	×	×	×	0,008	0,222	0,421	0,008	0,095	0,151	0,690	0,151
(5,400)	×	×	×	×	×	×	0,008	0,008	0,095	0,008	0,008	0,008	0,008
(10,400)	×	×	×	×	×	×	×	0,222	0,008	0,690	0,421	0,151	0,008
(20,400)	×	×	×	×	×	×	×	×	0,008	0,841	0,548	0,548	0,008
(5,800)	×	×	×	×	×	×	×	×	×	0,008	0,008	0,008	0,008
(10,800)	×	×	×	×	×	×	×	×	×	×	0,421	0,151	0,008
(20,800)	×	×	×	×	×	×	×	×	×	×	×	0,222	0,008
(40,2400)	×	×	×	×	×	×	×	×	×	×	×	×	0,008

(c) p-valeurs en fonction des paramètres des déformations RDF.

	10	15	30	100
5	0,016	0,016	0,310	0,008
10	×	0.690	0,151	0,008
15	×	×	0,151	0,008
30	×	×	×	0,008

(d) p-valeurs en fonction des paramètres des déformations CNB-MVN-MLS.

(n, σ)	(3,10)	(3,20)	(3,50)	(5,5)	(5,10)	(5,20)	(5,50)	(10,5)	(10,10)	(10,20)	(10,50)
(3,5)	0,008	0,008	0,008	0,032	0,016	0,016	0,421	0,056	0,008	0,310	0,008
(3,10)	×	0,008	0,032	0,151	0,151	0,421	0,008	0,841	0,095	0,016	0,008
(3,20)	×	×	0,421	0,008	0,548	0,056	0,008	0,032	0,151	0,008	0,008
(3,50)	×	×	×	0,008	1,000	0,151	0,008	0,032	0,421	0,008	0,008
(5,5)	×	×	×	×	0,056	0,548	0,095	0,421	0,008	0,056	0,008
(5,10)	×	×	×	×	×	0,421	0,016	0,222	0,690	0,016	0,008
(5,20)	×	×	×	×	×	×	0,008	0,841	0,151	0,032	0,008
(5,50)	×	×	×	×	×	×	×	0,032	0,008	0,841	0,151
(10,5)	×	×	×	×	×	×	×	×	0,056	0,016	0,008
(10,10)	×	×	×	×	×	×	×	×	×	0,008	0,008
(10,20)	×	×	×	×	×	×	×	×	×	×	0,056

(e) p-valeurs en fonction des paramètres des déformations GBD-3.

(n, σ)	(3,10)	(3,20)	(3,50)	(5,5)	(5,10)	(5,20)	(5,50)	(10,5)	(10,10)	(10,20)	(10,50)
(3,5)	0,310	0,008	0,008	0,690	0,056	0,008	0,841	0,310	0,032	0,310	0,032
(3,10)	×	0,421	0,222	0,310	0,841	0,151	0,222	0,421	0,222	0,056	0,016
(3,20)	×	×	0,548	0,008	0,421	0,421	0,008	0,008	0,841	0,008	0,008
(3,50)	×	×	×	0,008	0,222	0,690	0,008	0,008	1,000	0,008	0,008
(5,5)	×	×	×	×	0,056	0,008	1,000	0,841	0,095	0,056	0,008
(5,10)	×	×	×	×	×	0,095	0,056	0,151	0,841	0,008	0,008
(5,20)	×	×	×	×	×	×	0,008	0,008	0,548	0,008	0,008
(5,50)	×	×	×	×	×	×	×	0.690	0,095	0,310	0,032
(10,5)	×	×	×	×	×	×	×	×	0,056	0,095	0,008
(10,10)	×	×	×	×	×	×	×	×	×	0,032	0,008
(10,20)	×	×	×	×	×	×	×	×	×	×	0,151

(f) p-valeurs en fonction des paramètres des déformations TPS.

Table A.42 : p-valeurs des comparaisons d'indice de Dice des méthodes de déformation, pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

(w, s)	(0,1, 1)	(0,1, 2)	(0,1, 4)	(0,35, 1)	(0,35, 2)	(0,35, 4)	(0,7, 1)	(0,7, 2)	(0,7, 4)	(1,1)	(1,2)	(1,4)
(0,1, 1)	×	0,151	0,008	0,151	0,151	0,421	0,222	0,841	0,222	0,056	0,421	0,421
(0,1, 2)	×	×	0,421	0,008	0,008	0,032	0,421	0,151	0,032	0,548	0,222	0,841
(0,1, 4)	×	×	×	0,008	0,008	0,008	0,056	0,008	0,008	0,548	0,095	0,095
(0,35, 1)	×	×	×	×	1,0	0,421	0,056	0,222	0.690	0,032	0,056	0,032
(0,35, 2)	×	×	×	×	×	0.310	0,016	0,222	0.690	0,016	0,056	0,032
(0,35, 4)	×	×	×	×	×	×	0,056	0,841	0,548	0,016	0,056	0,151
(0,7, 1)	×	×	×	×	×	×	×	0,222	0,056	0,222	0,841	1,0
(0,7, 2)	×	×	×	×	×	×	×	×	0.310	0,056	0,421	0.310
(0,7, 4)	×	×	×	×	×	×	×	×	×	0,032	0,095	0,095
(1,1)	×	×	×	×	×	×	×	×	×	×	0.310	0,222
(1,2)	×	×	×	×	×	×	×	×	×	×	×	0,841

Table A.43 : p-valeurs des comparaisons d'indice de Dice de la méthode FBM, pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

(n, σ)	(3, 0,5)	(3,1)	(3,2)	(3,5)	(3,10)	(5, 0,5)	(5,1)	(5,2)	(5,5)	(5,10)	(10, 0,5)	(10,1)	(10,2)	(10,5)	(10,10)	(15, 0,5)	(15,1)
(3, 0,5)	×	0,151	0,056	0,222	0.690	0,095	0,008	0,008	0,032	0,032	0,548	0,008	0,008	0,841	0,008	0,056	0,008
(3,1)	×	×	0,841	0,222	0,056	0,421	0,841	0,548	0,151	0,016	0,421	1,0	1,0	0,095	0,008	0,151	0.690
(3,2)	×	×	×	0,222	0,032	0,421	1,0	0,841	0,151	0,008	0.310	0,841	1,0	0,032	0,008	0,222	0.690
(3,5)	×	×	×	×	0,151	1,0	0,056	0,095	1,0	0,008	0,841	0,056	0,222	0,151	0,008	0,841	0,151
(3,10)	×	×	×	×	×	0,095	0,008	0,008	0,032	0.310	0,222	0,008	0,008	1,0	0,222	0,056	0,008
(5, 0,5)	×	×	×	×	×	×	0,151	0,151	1,0	0,008	0.690	0,095	0.310	0,095	0,008	0,548	0,222
(5,1)	×	×	×	×	×	×	×	1,0	0,032	0,008	0,222	1,0	0,548	0,008	0,008	0,222	0,421
(5,2)	×	×	×	×	×	×	×	×	0,095	0,008	0,151	1,0	0.690	0,008	0,008	0,421	0.690
(5,5)	×	×	×	×	×	×	×	×	×	0,008	0,841	0,032	0,421	0,032	0,008	0,421	0,095
(5,10)	×	×	×	×	×	×	×	×	×	×	0,016	0,008	0,008	0,151	1,0	0,016	0,008
(10, 0,5)	×	×	×	×	×	×	×	×	×	×	×	0,151	0,222	0,421	0,008	0.690	0,222
(10,1)	×	×	×	×	×	×	×	×	×	×	×	×	0,548	0,008	0,008	0,095	0.310
(10,2)	×	×	×	×	×	×	×	×	×	×	×	×	×	0,008	0,008	0,841	0,841
(10,5)	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,056	0,095	0,008
(10,10)	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,008	0,008
(15, 0,5)	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	0,421

Table A.44 : p-valeurs des comparaisons d'indice de Dice de la méthode CPAB, pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

A.4 Résultats de segmentation complets du chapitre 4

A.4.1 Masques d'entrée et de sortie identiques

N_r \ N_g	0	660	3300
0		$93{,}62 \pm 0{,}06$	$93{,}86 \pm 0{,}12$
660	$93{,}71 \pm 0{,}10$	$94{,}10 \pm 0{,}16$	$93{,}98 \pm 0{,}18$

(a) Indice de Dice en fonction du nombre d'images réelles et synthétiques (masques d'entrée et de sortie identiques).

N_r \ N_g	0	660	3300
0		$93{,}94 \pm 0{,}36$	$93{,}35 \pm 0{,}73$
660	$94{,}45 \pm 0{,}75$	$94{,}27 \pm 0{,}38$	$93{,}76 \pm 0{,}57$

(b) Précision en fonction du nombre d'images réelles et synthétiques (masques d'entrée et de sortie identiques).

N_r \ N_g	0	660	3300
0		$93{,}29 \pm 0{,}38$	$94{,}38 \pm 0{,}56$
660	$92{,}99 \pm 0{,}67$	$93{,}92 \pm 0{,}39$	$94{,}22 \pm 0{,}44$

(c) Recul en fonction du nombre d'images réelles et synthétiques (masques d'entrée et de sortie identiques).

N_r \ N_g	0	660	3300
0		$96{,}79 \pm 0{,}21$	$96{,}41 \pm 0{,}44$
660	$97{,}08 \pm 0{,}44$	$96{,}95 \pm 0{,}22$	$96{,}65 \pm 0{,}34$

(d) Spécificité en fonction du nombre d'images réelles et synthétiques (masques d'entrée et de sortie identiques).

Table A.49 : Résultats en fonction du nombre d'images réelles et synthétiques (masques d'entrée et de sortie identiques).

A.4.2 Masques de sortie aléatoires

N_r \ N_g	0	660	3300
0		$90{,}32 \pm 0{,}49$	$90{,}98 \pm 0{,}30$
660	$93{,}71 \pm 0{,}10$	$93{,}70 \pm 0{,}10$	$93{,}98 \pm 0{,}10$

(a) Indice de Dice en fonction du nombre d'images réelles et synthétiques (masque de sortie aléatoire).

N_r \ N_g	0	660	3300
0		$91{,}22 \pm 1{,}26$	$92{,}51 \pm 0{,}81$
660	$94{,}45 \pm 0{,}75$	$93{,}76 \pm 0{,}43$	$94{,}20 \pm 0{,}39$

(b) Précision en fonction du nombre d'images réelles et synthétiques (masque de sortie aléatoire).

N_r \ N_g	0	660	3300
0		$89{,}48 \pm 1{,}92$	$89{,}51 \pm 1{,}22$
660	$92{,}99 \pm 0{,}67$	$93{,}65 \pm 0{,}57$	$93{,}75 \pm 0{,}50$

(c) Recul en fonction du nombre d'images réelles et synthétiques (masque de sortie aléatoire).

N_r \ N_g	0	660	3300
0		$95{,}39 \pm 0{,}80$	$95{,}53 \pm 1{,}03$
660	$97{,}08 \pm 0{,}44$	$96{,}67 \pm 0{,}26$	$96{,}92 \pm 0{,}24$

(d) Spécificité en fonction du nombre d'images réelles et synthétiques (masque de sortie aléatoire).

Table A.54 : Résultats en fonction du nombre d'images réelles et synthétiques (masque de sortie aléatoire).

A.4.3 Synthèse de texture : p-valeurs

(N_r, N_g)	(660,0)	(0,660)	(0,3300)	(660,660)	(660,3300)
(660,0)	×	0,095	0,095	0,008	0,016
(0,660)	×	×	0,008	0,008	0,008
(660,660)	×	×	×	0,032	0,421
(0,3300)	×	×	×	×	0,421

(a) p-valeurs des comparaisons d'indice de Dice pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

(N_r, N_g)	(0,660)	(0,3300)	(660,660)	(660,3300)
(660,0)	0,972	0,048	0,004	0,008

(b) p-valeurs des comparaisons d'indice de Dice pour l'hypothèse H0 : l'indice de Dice de la baseline (660,0) est plus élevé que les autres configurations (test unilatéral).

Table A.57 : p-valeurs des comparaisons d'indice de Dice de la synthèse de texture guidée multi-échelle (masques identiques).

(N_r, N_g)	(660,0)	(0,660)	(0,3300)	(660,660)	(660,3300)
(660,0)	×	0,008	0,008	0,841	0,008
(0,660)	×	×	0,056	0,008	0,008
(660,660)	×	×	×	0,008	0,008
(0,3300)	×	×	×	×	0,008

(a) p-valeurs des comparaisons d'indice de Dice pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

(N_r, N_g)	(0,660)	(0,3300)	(660,660)	(660,3300)
(660,0)	1,0	1,0	0,655	0,004

(b) p-valeurs des comparaisons d'indice de Dice pour l'hypothèse H0 : l'indice de Dice de la baseline (660,0) est plus élevé que les autres configurations (test unilatéral).

Table A.60 : p-valeurs des comparaisons d'indice de Dice de la synthèse de texture guidée multi-échelle (masques de sortie aléatoires).

A.5 Résultats de segmentation complets du chapitre 5

A.5.1 Simulation de nouvelles images

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	89,95 ± 0,94	79,58 ± 4,48	46,80 ± 14,45	83,19 ± 1,45
Carte de structure	80,88 ± 1,73	79,41 ± 1,30	84,80 ± 0,81	89,69 ± 1,35

(a) Indice de Dice en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	90,43 ± 0,99	95,31 ± 0,79	89,43 ± 4,38	76,60 ± 3,11
Carte de structure	92,71 ± 0,43	91,68 ± 1,05	81,85 ± 0,96	92,08 ± 0,67

(b) Précision en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	87,50 ± 4,65	68,57 ± 6,81	33,50 ± 16,13	91,13 ± 1,18
Carte de structure	71,76 ± 2,74	70,05 ± 1,53	87,99 ± 1,71	87,47 ± 2,72

(c) Recul en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	94,94 ± 0,63	98,18 ± 0,47	97,52 ± 2,26	85,04 ± 2,81
Carte de structure	96,99 ± 0,23	96,61 ± 0,42	89,58 ± 0,76	95,98 ± 0,43

(d) Spécificité en fonction de la version de SPADE et de l'entrée utilisée.

Table A.65 : Résultats en fonction de la version de SPADE et de l'entrée utilisée, images synthétiques uniquement.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	94,02 ± 0,19	93,68 ± 0,24	93,51 ± 0,19	93,10 ± 0,38
Carte de structure	93,90 ± 0,20	93,77 ± 0,11	93,07 ± 0,25	93,73 ± 0,11

(a) Indice de Dice en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	93,71 ± 0,36	94,17 ± 0,46	93,25 ± 0,79	92,40 ± 0,63
Carte de structure	94,43 ± 0,71	94,50 ± 0,93	91,91 ± 1,04	94,13 ± 0,67

(b) Précision en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	94,33 ± 0,52	93,21 ± 0,66	93,78 ± 0,90	94,14 ± 0,35
Carte de structure	93,38 ± 0,52	93,07 ± 0,86	94,27 ± 0,65	93,45 ± 0,79

(c) Recul en fonction de la version de SPADE et de l'entrée utilisée.

	SPADE-ADA	SPADE-ADA-REMIX	SPADE-ADA-MS	SPADE-ADA-ST
Masque binaire	96,62 ± 0,22	96,92 ± 0,27	96,37 ± 0,49	95,87 ± 0,37
Carte de structure	97,06 ± 0,41	97,10 ± 0,55	95,56 ± 0,66	96,89 ± 0,40

(d) Spécificité en fonction de la version de SPADE et de l'entrée utilisée.

Table A.70 : Résultats en fonction de la version de SPADE et de l'entrée utilisée, images réelles et synthétiques.

Entraînement avec images réelles, images générées avec masques binaires et images générées avec cartes de structure :

- Indice de Dice : $94{,}09 \pm 0{,}14$
- Précision : $94{,}20 \pm 0{,}43$
- Recul : $93{,}98 \pm 0{,}53$
- Spécificité : $96{,}91 \pm 0{,}26$

	Indice de Dice	Précision	Recul	Spécificité
Images réelles	$95{,}02 \pm 0{,}06$	$96{,}07 \pm 0{,}17$	$94{,}00 \pm 0{,}26$	$97{,}95 \pm 0{,}10$
Images synthétiques	$94{,}91 \pm 0{,}11$	$95{,}19 \pm 0{,}60$	$94{,}65 \pm 0{,}56$	$97{,}44 \pm 0{,}35$

Table A.71 : Résultats limites théoriques.

A.5.2 Reproduction de la base d'entraînement

	Indice de Dice	Précision	Recul	Spécificité
Masques binaires	$87{,}91 \pm 0{,}88$	$89{,}09 \pm 2{,}63$	$86{,}97 \pm 4{,}11$	$94{,}22 \pm 1{,}83$
Cartes de structure	$69{,}87 \pm 5{,}67$	$96{,}94 \pm 0{,}87$	$54{,}84 \pm 6{,}72$	$99{,}09 \pm 0{,}18$
Réelles + masques binaires	$93{,}82 \pm 0{,}18$	$93{,}57 \pm 1{,}07$	$94{,}08 \pm 0{,}96$	$96{,}54 \pm 0{,}66$
Réelles + cartes de structure	$93{,}56 \pm 0{,}24$	$95{,}07 \pm 0{,}60$	$92{,}10 \pm 0{,}80$	$97{,}45 \pm 0{,}34$
Toutes	$94{,}01 \pm 0{,}14$	$94{,}63 \pm 0{,}45$	$93{,}40 \pm 0{,}52$	$97{,}17 \pm 0{,}27$

Table A.72 : Résultats pour la reproduction par SPADE de la base d'entraînement.

A.5.3 Translation d'image à image : p-valeurs

	Baseline	ada-gts	ada-hsv	remix-gts	remix-hsv	gatys-gts	gatys-hav	ms-gts	ms-hsv
Baseline	×	0,008	0,008	0,008	0,008	0,008	0,008	0,008	0,008
ada-gts	×	×	0,008	0,008	0,008	0,008	0,841	0,008	0,008
ada-hsv	×	×	×	0,841	0,222	0,095	0,008	0,008	0,008
remix-gts	×	×	×	×	1,0	0,222	0,008	0,008	0,032
remix-hsv	×	×	×	×	×	0,008	0,008	0,008	0,008
gatys-gts	×	×	×	×	×	×	0,008	0,008	0,095
gatys-hsv	×	×	×	×	×	×	×	0,008	0,008
ms-gts	×	×	×	×	×	×	×	×	0,008

Table A.73 : p-valeurs des comparaisons d'indices de Dice en fonction des différentes configurations de SPADE (simulation d'images réelles, images synthétiques uniquement), pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

	Baseline	ada-gts	ada-hsv	remix-gts	remix-hsv	gatys-gts	gatys-hav	ms-gts	ms-hsv
Baseline	×	0,056	0,151	1,0	0.310	0,016	0,548	0,095	0,008
ada-gts	×	×	0,222	0,056	0,095	0,008	0,095	0,008	0,008
ada-hsv	×	×	×	0,222	0.310	0,016	0,421	0,056	0,008
remix-gts	×	×	×	×	0.310	0,032	0,548	0.310	0,008
remix-hsv	×	×	×	×	×	0,008	0,841	0,008	0,008
gatys-gts	×	×	×	×	×	×	0,008	0,095	0.690
gatys-hsv	×	×	×	×	×	×	×	0,056	0,008
ms-gts	×	×	×	×	×	×	×	×	0,032

Table A.74 : p-valeurs des comparaisons d'indices de Dice en fonction des différentes configurations de SPADE (simulation d'images réelles, mélange d'images réelles et d'images synthétiques), pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

	Baseline	gts	hsv	gts+réelles	hsv+réelles	gts+hsv+réelles
Baseline	×	0,008	0,008	0.310	0.310	0,016
gts	×	×	0,008	0,008	0,008	0,008
hsv	×	×	×	0,008	0,008	0,008
gts+réelles	×	×	×	×	1,0	0,032
hsv+réelles	×	×	×	×	×	0,032

Table A.75 : p-valeurs des comparaisons d'indices de Dice en fonction des différentes configurations de SPADE (reproduction de la base d'entraînement), pour l'hypothèse H0 : les distributions sont identiques (test bilatéral).

	Baseline	Réelles (1320)	ada-gts+réelles	ada-identité + réelles	ada-gts+ada-hsv+réelles
Baseline	×	0,004	0,028	0,004	0,004
Réelles (1320)	×	×	1,0	0,925	1,0
ada-gts+réelles	×	×	×	0,004	0,5
ada-identité + réelles	×	×	×	×	1,0

Table A.76 : p-valeurs des comparaisons d'indices de Dice en fonction de différentes configurations de SPADE, pour l'hypothèse H0 : la première distribution est supérieure à la deuxième (test unilatéral).