Artificial intelligence methods for object recognition : applications in biomedical imaging

Damien Blanc

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Biostatistiques

École doctorale Information Structure Système

Unité de recherche UMR 5149

Présentée par

Damien BLANC

le 9 Décembre 2022

Artificial intelligence methods for object recognition: Applications in biomedical imaging

Sous la direction de **Joseph SALMON**
Coencadré par **Benjamin CHARLIER**

Devant le jury composé de

Renata Basto,	Directrice de recherche, Institut Curie	Examinatrice
Aurélie Bugeau,	Professeure, Université de Bordeaux	Rapporteure
Benjamin Charlier,	Enseignant-chercheur, Université de Montpellier	Co-encadrant
Thérèse Commes,	Professeure, Université de Montpellier	Présidente du jury
Oumou Goundiam,	Ingénieure de recherche, Institut Curie	Invitée
Charles Kervann,	Directeur de recherche, Inria Rennes	Rapporteur
Victor Racine,	CEO/CSO, QuantaCell	Examinateur
Joseph Salmon,	Professeur, Université de Montpellier	Directeur de thèse

UNIVERSITÉ DE MONTPELLIER

Remerciements

Enfin ! C'est le premier mot qui me vient en tête à l'idée de terminer ce manuscrit. J'ai encore beaucoup de mal à réaliser que ce soit enfin à mon tour d'écrire ces lignes. Cette thèse a représenté un défi incroyable que je n'espérais pas pouvoir relever un jour. Ainsi, je tenais à exprimer toute ma gratitude aux personnes qui ont participé, de quelque façon que ce soit, à m'épauler pour venir à bout de ces trois années de recherche.

Mes encadrants Je tenais tout d'abord à remercier mes encadrants pour m'avoir accompagné tout au long de cette thèse. Merci à Joseph et Benjamin pour leur rigueur et leurs conseils scientifiques. Merci Victor pour ta bienveillance et la confiance que tu m'as accordé pour mener ce projet à son terme. J'espère que les prochains seront tout aussi passionnants !

Le jury Je voudrais remercier Aurélie Bugeau et Charles Kervrann pour avoir accepté de rapporter les travaux présentés dans cette thèse. Merci à Thérèse Commes pour sa parcipation en tant qu'examinatrice. J'adresse également mes remerciements les plus sincères à Renata Basto et Goundiam Oumou pour leur disponibilité et leurs retours sur les résultats de cette thèse.

QuantaCell Évidemment, je remercie toute l'équipe de QuantaCell que j'ai eu plaisir à voir s'agrandir au fil de ma thèse, et qui je l'espère grandira encore après celle-ci. Merci à Jocelyne pour ta bonne humeur, et merci Paul-Axel pour le partage de tes connaissances sur à peu près n'importe quel sujet. Merci à Titouan d'être l'arbitre ~~impartial~~ de toutes les blagues du bureau. Merci à Karthik de faire progresser mon anglais, merci à Dimitri pour ta spontanéïté et merci à Élise de mettre quotidiennement à l'épreuve son auto-dérision, même quand c'pô très nice ! Enfin, un grand merci à Gaëtan (et Sophie) pour toutes les discussions, scientifiques ou non, qu'on a pu avoir ensemble. J'ai grandement apprecié découvrir ta sincérité et ton humilité dans chaque conseil que tu as pu me donner.

L'Institut des Neurosciences de Montpellier Évidemment, une petite pensée particulière envers l'INM, qui a été pour moi comme un second labo pendant une partie de cette thèse. Merci à Chloé, Donovan, Marjorie et Alicia pour toutes les sorties, soirées, ou week-ends qu'on a pu faire ensemble. Merci également à Mélissa et Quentin d'être les plus grands cordons bleus de Montpellier ! Bien sûr, un grand merci à Adrien, qui a également toute sa place dans les paragraphes suivants, et qui a toujours été d'un soutien sans faille déjà bien avant cette thèse. Merci de m'avoir initié vers la sagesse de "l'esprit concours", et merci pour tous ces potins qu'on a partagé et qui font partis de mes plus gros fous rires de ces dernières années. Merci aussi à toi et Gaëtan pour tous les restos qu'on a fait ensemble, j'attends les prochains avec impatience !

Mes ami(e)s Merci à Vincent, Emmeline, Maëva, Lola, Faustine et Sören d'avoir grandi avec moi et d'être à mes côtés depuis tant d'années. Merci d'être là malgré mes absences ces trois dernières années, et merci d'avoir été si indulgent avec moi. Je suis si heureux de savoir que nous pouvons toujours continuer à partager tous nos plus grands moments de vie ensemble.

Ma famille Je tenais aussi à remercier toute ma famille pour le soutien et l'amour que vous me portez. Maman, merci d'avoir vécu les hauts et les bas de toutes mes études avec moi, tu mérites amplement tous les diplômes que j'ai passés ! Pierre, merci d'avoir su nous gérer dans les moments où le stress prenait le dessus. Merci à Anaïs, Benjamin, Valentin, Sarah, Moé, et Nathanaël: ma longue liste de frères et soeurs pour leur présence malgré la distance qui nous sépare parfois. Merci à mes grands-parents pour leurs encouragements et leurs nombreux allers-retours entre Avignon et Montpellier. Merci Papa d'avoir été présent à ta façon, même lorsque l'intérêt de faire une thèse t'échappait. Merci Bruno pour toutes tes questions et l'intéret que tu as porté sur mes recherches. Je tenais aussi à remercier la famille Coyat/Louboutin pour tous les bons moments qu'on a pu passer ensemble. Sachez néanmoins que je n'aurai aucune pitié pour nos prochaines parties de molkky !

Enfin, pas une seule ligne de ce manuscrit n'aurait de sens sans *toi*, Carolanne. Merci d'avoir traversé ta "seconde thèse" avec moi et merci de m'avoir relevé chaque fois où j'ai été pris de doutes. Merci pour ta patience, ton réconfort et ton amour. Merci d'avoir fait de cette thèse une simple parenthèse dans la vie que nous nous construisons à deux, et surtout merci de m'avoir fait découvert tous les bienfaits de la ronronthérapie ! Simplement, mille fois merci pour tout.

"*La théorie, c'est quand on sait tout et que rien ne fonctionne.*
La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi.
Ici, nous avons réuni théorie et pratique : Rien ne fonctionne... et personne ne sait pourquoi ! "

Albert Einstein

Abstract

This work deals with the use of automatic methods based on artificial intelligence in different contexts of biomedical imaging. The main objective is to investigate in what extend deep learning models (DL) can be used in solving concrete clinical problems. In particular, we highlight the models called *Convolutional Neural Networks* (CNNs), whose architecture is specifically adapted to the recognition and segmentation of objects in visual data such as images or videos. In the first part, we address the need to develop diagnostic support systems in the context of a tomodensitometric examination (also called CT -scan) for the detection of lung tumors. To this end, we are investigating the performance of state-of-the-art object detection models in a three-dimensional context on a dataset of more than 1000 publicly available patients. The results of this initial study have then allowed us to develop a complete ool capable of distinguishing healthy patients from those with tumors. This tool includes three major methods (lung segmentation, nodule detection and patient classification) and was selected as the winner of the 2019 Data Challenge organized by the French Society of Radiology (FSR).

In the second part of the manuscript, we put ourselves in a context where data, an essential element for the construction of a robust supervised model, is unavailable or scarce. We often find this configuration in Microscopy imaging, where it is necessary to correctly identify each cellular component, such as the nucleus, to determine a biological outcome. As an alternative to tedious and costly labeling of the pixels of each nucleus, we propose a modeling of all the mechanisms that enable the acquisition of images of biological samples. In this way, we obtain a tool capable of rapidly generating, from a set of predefined parameters, a large amount of data consisting of simulated images and an associated ground truth for each of these images. Finally, we investigate the impact of using simulated datasets on the performance of models designed for nuclei segmentation. Thus, we show that simulation can be used with a relatively small set of real images as a tool for data augmentation and can provide significant performance gains.

The final part of this manuscript is devoted to the development of a methodology for quantifying cancerous ovarian tissue in 3D confocal imaging. Starting from a cohort of 119 patients, the main objective is to automate the quantification of the centrosome-nucleus index (CNI), allowing a better understanding of the role played by the major organs of the cell (*i.e.* nuclei and centrosomes) in the development of the disease. The study is therefore divided into two counting tasks : first the centrosomes, then the nuclei. The visual counting performed for the centrosomes is based on the search for colocalized regions in the image, i.e., regions whose signals are overlapped by different channels. Thus, we implement this approach and compare the performances with methods using machine learning or deep learning models. Since we do not have voxel-level annotations, we extend our simulator to a third dimension and implement a reinforcement learning approach to automatically find the simulation parameters that provide the highest possible counting accuracy.

Keywords— Artificial intelligence, Deep Learning, Computer vision, Object detection, Image segmentation, Biomedical imaging

Résumé

Cette thèse porte sur l'utilisation de méthodes automatiques basées sur l'intelligence artificielle dans différents contextes d'imagerie biomédicale. L'objectif principal est d'explorer dans quelle mesure des modèles d'apprentissage profond, ou *Deep Learning* (DL), peuvent s'implanter dans la résolution de problématiques cliniques concrètes. En particulier, nous mettons l'accent sur les modèles nommés *Convolutional Neural Networks* (CNNs) dont l'architecture est spécialement adaptée à la détection et à la segmentation d'objets sur des données visuelles telles que les images ou les vidéos. Dans une première partie, nous abordons ainsi la nécessité de développer des systèmes d'aides au diagnostique dans le cadre d'examen tomodensitométrique (également appelé CT-scan) pour la détection de tumeurs pulmonaires. Nous étudions ainsi les performances de l'état de l'art des modèles de détection d'objet dans un contexte en trois dimensions sur un jeu de données de plus de 1000 patients disponibles publiquement. Les résultats obtenus sur cette première étude nous ont ensuite permis de construire un outil complet d'aide au diagnostique, capable de distinguer les patients sains de ceux disposant de tumeurs. Cet outil comprends trois procédures majeures (segmentation des poumoons, détections des nodules, et classification du patient) qui a permis de remporté l'édition 2019 du Data Challenge organisé par la Société Française de Radiologie (SFR).

Dans la seconde partie du manuscrit, nous nous plaçons cette fois dans un cadre où la donnée, élément essentiel pour construire un modèle supervisée robuste, n'est pas ou peu disponible. Nous retrouvons fréquemment cette configuration en imagerie microscopique, où il est nécessaire d'identifier correctement chaque composant cellulaire, comme le noyau, afin de pouvoir dégager un résultat biologique d'intérêt. En alternative à un étiquetage fastidieux et couteux des pixels de chaque noyau, nous proposons une modélisation de l'ensemble des mécanismes permettant l'acquisition d'images d'échantillons biologiques. Nous obtenons ainsi un outil capable de générer rapidement, à partir d'un ensemble de paramètres prédéfinis, une grande quantité de données composées d'images simulées et d'une vérité terrain associée pour chacune d'entre elles. Enfin, nous étudions l'impact de l'utilisation de jeu de données simulées sur les performances de modèles conçus pour la segmentation de noyaux. Nous montrons ainsi qu'avec une quantité relativement faible d'image réelle, la simulation peut être utilisé comme outil d'augmentation de données et permet un gain important de performances.

Enfin, la dernière partie de ce manuscrit est consacrée à l'élaboration d'une méthodologie pour la quantification de tissus ovariens cancéreux en imagerie microscopique confocale 3D. A partir d'une cohorte de 119 patientes, l'objectif principal est d'automatiser la quantification du Centrosome-Nucleus Index (CNI), et permettre ainsi de mieux comprendre le rôle que jouent les principaux organes de la cellule (dans notre contexte, le noyau et les centrosomes) dans l'évolution de la maladire. L'étude est ainsi divisée en deux tâches de comptages : d'abord des centrosomes, puis des noyaux. Le comptage visuel qui a été fait pour les centrosomes se base sur la recherche de régions colocalisées dans l'image, c'est-à-dire, des régions dont les signaux issus de différents canaux se superposent. Nous implémentons ainsi cette approche, et comparons ces performances avec des méthodes impliquant des modèles de Machine Learning ou Deep Learning. Enfin, ne disposant pas d'annotations au niveau des voxels, nous étendons notre simulateur

à une troisième dimensions, et implémentons un approche d'apprentissage par renforcement afin de trouver automatiquement les paramètres de simulations permettant de donner la plus grande précision possible de comptage.

Mots clés— Intelligence artificielle, Apprentissage profond, vision par ordinateur, segmentation d'images, détection d'objets, imagerie biomédicale

Résumé substantiel

Cette thèse porte sur l'utilisation de méthodes automatiques basées sur l'intelligence artificielle dans différents contextes d'imagerie biomédicale. L'objectif principal était d'explorer dans quelle mesure des modèles de *machine learning* (ML) ou *Deep Learning* (DL) peuvent s'implanter dans la résolution de problématiques cliniques concrètes. Les paragraphes suivants constituent ainsi un bref résumé des chapitres présentant les travaux réalisés.

Chapitre 2 : Fondamentaux de l'apprentissage profond pour la vision par ordinateur

Ce chapitre donne les bases théoriques et méthodologiques nécessaires afin d'appréhender les problématiques et les challenges à relever dans le domaine de la vision par ordinateur. Nous commençons par établir un premier constat : depuis le début des années 2010, les modèles prédictifs réalisent des progrès fulgurants dans le domaine de la reconnaissance d'image. À l'origine de ces avancées, les réseaux de neurones ont ainsi grandement gagné en notoriété au sein de la communauté scientifique. Appartenant à la grande famille des méthodes basées sur l'apprentissage profond (ou *Deep Learning*), notre démarche consiste dans un premier temps à revenir aux prémisses des modélisations mathématiques qui ont par la suite permis d'aboutir à des modèles capable de défier, voir de surpasser, les performances humaines.

L'inspiration première des réseaux de neurones artificiels provient d'une modèlisation du fonctionnement des neurones biologiques. L'idée est alors de mimer les deux propriétés fondamentales de ces derniers : la capacité à réagir à des stimulations extérieurs (excitabilité) mais aussi celle à transmettre des impulsions nerveuses (conductivité). Mathématiquement, ce mécanisme se traduit par un modèle, appelé *neurone formel* composé de 4 éléments : un vecteur d'entrée d'éléments réels, des poids et biais constituant les paramètres, une fonction d'activation non-linéaire, et une sortie. Dans sa forme la plus simple, ce neurone artificiel va pondérer ses entrées, les sommer, puis comparer le résultat à une valeur seuil déterminant la sortie finale. En munissant ce modèle d'un algorithme permettant la mise à jour automatique de l'ensemble de ces paramètres, nous créons ainsi un *perceptron* capable de réaliser efficacement une tâche de classification binaire. Afin de résoudre des tâches plus complexes, cet algorithme a été étendu au perceptron multicouche, qui consiste simplement à un empilement de "couches" de perceptron, dont chaque élément est entièrement connecté aux autres. Il s'agit ainsi de la modélisation la plus basique des réseaux de neurones, qui inspirera ensuite le développement d'architectures plus complexes.

Afin d'être plus axé autour des problèmatiques rencontrés lors de traitement de données visuelles, nous centrons le reste du chapitre autour des *Convolutional Neural Networks* (CNNs) et de leur fonctionnement. En particulier, comme leur nom l'indique, nous étudions le principe des opérations de convolutions et leur utilité dans le domaine du traitement de l'image. Aussi, nous fournissons une description détaillée des architectures classiquement recontrées pour réaliser des tâches de classification ou de segmentation d'images.

Enfin, un aperçu de l'état de l'art des modèles de détection d'objets est présenté. Les premières contributions majeures dans cette tâche sont attribuées au *Region-based CNN* (R-CNN), dont le principe repose sur une extraction de certaines régions dans l'image,

puis une classification de celles-ci. Nous présentons ensuite quelles ont été les améliorations apportées dans la littérature pour détecter des objets de différentes tailles, et être capable de les séparer dans un contexte où ceux-ci sont extrêmement proches les uns des autres.

Chapitre 3 : Utilisation du Deep Learning pour la détection de nodule pulmonaire sur imagerie CT-scan 3D

Le cancer du poumon est classé parmi les tumeurs malignes les plus meurtrières dans le monde. En France, cette maladie est un problème de santé publique majeur du fait de son épidémiologie mais aussi de son caractère particulièrement aggressif. Ainsi, le taux de survie à 5 ans est estimé à seulement 18%. Néanmoins, le diagnostic précis de nodule pulmonaires à stade précoce est une des clés principale pour réduire la mortalité de cette maladie. Ainsi, l'approche traditionnelle pour la détection de nodules pulmonaire passe par un suivi médical du patient avec un ou plusieurs scanners à tomodensitométrie basse dose (communément appelé CT-scan). Bien qu'efficace, cette méthode est très couteuse à la fois sur le plan matériel, mais aussi sur le plan médical et humain. En effet, l'imagerie des scanners est délivré sous la forme d'un grand volume de données en trois dimensions, qui est à la fois difficile à manipuler et particulièrement fastidieux à inspecter. Au sein de la communauté des radiologues, il y a ainsi un réel besoin d'automatisation et/ou d'accélération de cette tâche. Nous proposons ainsi d'explorer les performances des modèles de détections dans ce contexte.

Nous présentons dans un premier temps le jeu de données *Lung Image Database Consortium* (LIDC) qui nous servira de base de données de référence pour nos expériences. Celle-ci regroupe un total de 1035 CT scan, annotées en trois dimensions par différentes équipes hospitalières. Les informations extraîtes grâce à ce processus d'annotations nous permettent de transformer le jeu de données de sorte à obtenir, pour chaque patient, un volume ainsi qu'un masque de segmentation 3D où chaque pixel sera étiqueté selon son appartenance à une classe spécifique (0 : non-nodule, 1 : nodule bénins, 2 : nodules malins).

Nous présentons dans un second temps un framework particulièrement adapté à notre problématique et regroupant différentes implémentations (2D et 3D) de modèles de détection d'objets. En particulier, nous choisissons de porter notre analyse sur un modèle de segmentation (appelée U-FPN), un modèle de détection à deux-étapes (Mask R-CNN) et deux modèles de détection à une étape (Retina Net, Retina U-net). Une attention particulière sera portée à la difficulté de traiter des données aussi volumineuse, nous obligeant ainsi à passer par des méthodes de tuilage (ou *patching*) consistant à donner des sous-sections de l'image originale en guise de lots d'entraînement. Également, nous présentons une méthode simple basée sur un pré-tri des partients permettant l'équilibrage des classes au sein des lots d'entraînement. Après une brève présentation des différents modèles utilsés, nous procédons finalement à une comparaison complète de chacun d'entre eux en incluant à chaque fois les architectures 2D et leur équivalent en 3D. Cette comparaison se fait sur plusieurs critères en écho avec une utilisation clinique : nombre de détections par patient, temps d'analyse, métrique Average Precision (AP), score Competition Performance Metric (CPM). Les résultats tendent à montrer plusieurs faits. Le plus évident, est celui de la superiorité des modèles 3D sur leur équivalent 2D en terme de précision de localisation (meilleur score AP et CPM) mais aussi en terme de rapiditité de temps de réponse : le temps moyen d'analyse d'un patient est de l'odre de 15 à 25 secon-

des en 3D quand il dépasse les 40 voir les 50 secondes en 2D. En s'appuyant sur ces résultats, nous avons poursuivi notre étude exclusivement sur les modèles 3D et avons implémenté par la suite une stratégie de réduction des détections faux positifs. Celle-ci se base simplement sur une exclusion des boites de détection qui ne chevauche pas suffisament les poumons. Une procédure de segmentation des poumons a ainsi été réalisé par une combinaison de méthodes de seuillage et d'opération mathématiques morphologiques. Nous avons ainsi pu observer que cette simple approche a améliorer la précision de tous les modèles de détection.

Enfin, nous présentons la méthodologie que nous avons employée lors de notre participation au Data Challenge organisé lors des *Journées Francophones de la Radiologie* (JFR) par la Société Française de Radiologie (SFR). L'objectif de cet événement était de regrouper divers communautés scientifiques autour de différentes problématiques de santé publique. Parmi les challenges proposés, l'un d'eux portait sur le diagnostic de CT scan 3D autour de la problématique du cancer du poumon. EN se basant sur l'idée que le volume d'un nodule est un facteur prédictif important dans l'évolution de la maladire, l'objectif était de créer un algorithme capable de classer chaque scanner de patient en "normale - sans signes de nodules malins (volume $< 100\,\mathrm{mm}^3$)" vs. "anormale - présence probable de régions suspectes (volume $\geq 100\,\mathrm{mm}^3$)". En quatre semaines, nous avons ainsi pu mettre en place un pipeline complet de diagnostique du cancer du poumon. Ce pipeline se décompose en 3 trois étapes majeurs et reflétent les tâches réalisées visuellement par les radiologistes : une segmentation des poumons, une détection des nodules, puis une classification de ces derniers pour indiquer si leur volume est supérieur à $100\,\mathrm{mm}^3$. Les deux première étapes sont assez simmilaires à celle présentées précédement, et la troisième implique l'extraction de paramètre (de taille, d'intensité moyenne, ...) afin d'entraîner un modèle Support Vector Machine (SVM) pour donner une réponse finale au niveau du patient. Ainsi, cette méthodologie nous a permi de remporter la compétition avec un score AUC (Area Under the Curve) à 0.899.

Chapitre 4 : Entraînement de réseaux de neurones avec des images synthétiques de microscopie

De par leur nature supervisée, la robustesse des modèles d'apprentissage est fortement dépendante de la quantité, mais aussi de la qualité, des données fournies pour l'entraînement de ces derniers. Afin de répondre à une problématique spécifique, il est alors nécessaire de construire une base de données la plus conséquentes possibles, et dont le processus d'étiquetage est rigoureux et consistant. Ceci nécessite souvent un processus très couteux, que ce soit en terme matériel ou en temps humain. En particulier dans ce chapitre, nous nous concentrons particulièrement sur l'imagerie de microscopie, dont les fondements sont intrésequement lié à la recherche en biologie médicale. Ainsi, afin de procéder à l'analyse d'un mécanisme biologique quelconque, il est souvent nécessaire de pouvoir extraire en amont un ensemble de caractéristiques morphologiques des noyaux cellulaires, ce qui passe donc par une tâche de segmentation de l'image.

Cette tâche peut être réalisée par un CNN entraîné avec suffisament de données. Ainsi, notre principale contribution consiste au développement d'un modèle génératif d'image de microscopie. Nous détaillons ainsi chaque étape modélisée qui nous mène à la formation d'une image et de sa vérité-terrain : la taille, forme et texture des noyaux, la disposition spatiale de la population dans l'image, les variation d'intensité et d'illumination, le flou optique, les variation du bruit, etc... Par simplicité et pour plus

d'accessibilité, nous avons également réalisé une interface graphique qui permet à un utilisateur (expert ou non-expert) de pouvoir faire des tests simples pour visualiser le rendu que donnent certaines valeurs de ces paramètres de simulation.

Finalement, afin d'évaluer la qualité de notre modèle, nous cherchons à évaluer l'impact de l'utilisation de données simulées comme ensemble d'apprentissage sur les performances de segmentation des noyaux cellulaires. Nous nous appuyons alors sur deux modèles de l'état de l'art pour la segmentation des noyaux et deux jeux de données publiques issus de divers échantillons biologiques montrés sous différentes modalités d'acquisition. Dans un premier temps, nous comparons les performances de chaque modèle en faisant varier la quantité d'image utilisée dans l'ensemble d'entraînement et en prenant à chaque fois soit des données exclusivement réelles, soit exclusivement simulées. Ceci nous ammène à la premiere conclusion plutôt attendue que les performances des modèles CNNs augmentent avec la quantité de données utilisées pour l'entraînement. Il est cependant à noter que pour une même quantité d'images, il sera en général préférable d'utiliser des données réelles plutôt que des simulées. Dans un second temps, nous cherchons à voir cette fois si l'ajout de données simulées, en supplémant d'une certaine quantité de données réelles, permet d'avoir un gain de performances. Nous réalisons de nouveau l'expérience précédente en définissant cette fois, pour chaque proportion de données réelles utilisées, un facteur d'augmentation de données. Ce facteur donne ainsi le nombre d'images simulées rajoutées pour chaque donnée réelle utilisée. Nous réalisons ces expériences à 3 niveaux d'augmentation : 1, 5, 10, et nous observons cette fois une augmentation nette des performances lorsque le facteur d'augmentation croit. Pour une comparaison plus complète, nous réalisons ces mêmes expériences en utilisant des procédés de data augmentation plus traditionnelles, impliquant des transformations géométriques des images. Finalement, nous observons que le mélange données simulées + données transformées est un mélange encore plus efficace pour la segmentation de noyaux cellulaires.

Chapitre 5 : Méthodes de quantifications de tissus cancéreux ovariens sur imagerie microscopique confocale 3D

Les cancers épithéliaux de l'ovaire constituent le 5ème cancer féminin le plus létal dans le monde occidental. Bien que leurs causes précises soient encore mal connues, la transformation maligne des cellules constituant le tissu de surface de l'ovaire, en contiguïté avec une membrane qui tapisse les parois intérieures de l'abdomen, semble jouer un rôle important dans l'apparition de la maladie. Le faible pronostic de la maladie est lié à une détection tardive, lorsque les cellules tumorales ont atteint le pelvis ou l'abdomen. En outre, le taux de survie à 5 ans des cancers de l'ovaire va de 30 à 40% en raison du manque de symptômes spécifiques ou de biomarqueurs validés. Un protocole expérimental a été mis en place pour imager une cohorte composée 19 tissues sains et 100 tissues cancéreux de l'ovaire avec les principaux organes d'intérêt (noyau, centrosomes, microtubules, Golgi). Ces données permettent d'analyser manuellement l'organisation du cytosquelette dans ces tumeurs et d'améliorer la compréhension du rôle des centrosomes et des microtubules dans l'établissement et la progression des carcinomes de l'ovaire. Néanmoins, l'exploration du protocole est grandement freinée par la quantification manuelle que requièrent ces images. En effet, ces dernières nécessitent un grand nombre de pointage manuel que seul un expert est capable de réaliser, rendant alors le processus de quantification extrêmement lent. L'objectif dans ce chapitre est d'utiliser

les technologies d'intelligence artificielle (notamment de Deep Learning) pour faire apprendre à une « machine-expert » les quantifications déjà existantes. La stratégie adoptée dans ce contexte se divisera en deux temps, et proposera d'abord des méthodes de quantifications du nombre de centrosomes, puis du nombre de noyaux.

Les méthodes de détections du centrosome est assez peu étudiée dans la littérature. De plus, la morphologie des centrosomes est très diverse (centrosome simple isolé, clusters, superclusters). Le comptage des centrosomes est donc une tâche très complexe, car en fonction de la morphologie des objets détectés, plusieurs centrosomes doivent être comptés. Aussi, nous proposons de comparer différentes méthodologies de quantifications que nous avons nous même développés. Ces méthodes se basent essentiellement sur une mesure de la colocalisation entre les canaux rouges et verts de chaque image. Les approches proposées commencent ainsi par une démarche par traitement d'image traditionnelle et se dirigent ensuite sur l'utilisation d'un modèle ResNet conçu pour réaliser des tâches de régression. Après utilisation de chacune des méthodes sur l'ensemble de la cohorte, il apparaît que les tissus sains peuvent être raisonnablement quantifiés à partir de simple méthodes de traitement d'images, tandis que les tissus cancéreux, plus complexes, ont été mieux quantifiés par les méthodes impliquant un processus d'apprentissage.

Concernant le comptage des noyaux, la difficulté de fournir des annotations de segmentations 3D nous a mené à utiliser le simulateur construit dans le chapitre précédent. Nous proposons alors une approche par *Reinforcement learning* (ou apprentissage par renforcement) pour déterminer quel serait le meilleur paramètrage de notre simulateur donnant les meilleures performances sur les données réelles. Cette méthode est assez coûteuse en temps de calcul et ne permet pas un grand nombre d'expérience en temps raisonnable. Aussi, nous validons dans un premier temps les choix stratégiques à opter sur un jeu de données en deux dimensions, avant de réaliser quelques expériences sur la cohorte complète.

Enfin, la dernière partie de ce chapitre expose les résultats obtenus sur notre quantification du Centrosome-Nucleus Index (CNI), correspondant au ratio des quantifications obtenus pour les centrosomes et les noyaux.

Chapitre 6 : Conclusions

Ce chapitre de conclusion établit une synthèse de nos démarches et de nos résultats obtenus au travers de chaque domaines d'applications que nous avons étudié. Nous y listons également les limites de nos travaux et discutons également des approches alternatives qui pourront être suivies dans de futurs travaux.

Contents

Remerciements i

Abstract ii

Résumé iii

Résumé substantiel iv

Contents ix

Liste of Figures xii

List of Tables xvii

1 Introduction 1
1.1 General context . . . 1
1.2 Objectives . . . 2
1.3 Thesis organization . . . 2

2 Deep Learning fundamentals for computer vision 3
2.1 Introduction . . . 4
2.2 Deep Learning basics . . . 5
2.2.1 The perceptron . . . 5
2.2.2 Multi-layer perceptron . . . 6
2.2.3 Backpropagation . . . 8
2.2.4 Gradient-based optimization . . . 9
2.3 Convolutional neural network (CNN) . . . 11
2.3.1 Convolutional layer . . . 11
2.3.2 Standard layers of CNNs . . . 14
2.3.3 ResNet architecture . . . 16
2.3.4 Fully convolutional network . . . 17
2.3.5 U-net architecture . . . 19
2.4 CNNs for object detection . . . 20
2.4.1 Performing an object detection task . . . 20
2.4.2 Traditional detectors . . . 21
2.4.3 Region-based CNN (R-CNN) . . . 22
2.4.4 Fast R-CNN and SSP-Net . . . 23
2.4.5 Faster R-CNN and Mask R-CNN . . . 23
2.4.6 You only look once . . . 25
2.4.7 Single shot detector . . . 27
2.4.8 Retina Net . . . 27

2.5 Conclusions . . . 29

3 Pulmonary nodule detection on 3D CT scans. 31
3.1 Medical context . . . 32
3.1.1 Motivations . . . 32
3.2 Training strategy for nodule detection . . . 33
3.2.1 Datasets . . . 34
3.2.2 Preprocessing . . . 35
3.2.3 Compared models and implementation details . . . 36
3.2.4 Loss functions . . . 38
3.2.5 Batch balancing . . . 40
3.2.6 Data augmentation . . . 41
3.3 Evaluating nodule detectors . . . 43
3.3.1 Intersection over union . . . 43
3.3.2 Filtering detections . . . 44
3.3.3 Labeling of detections . . . 44
3.3.4 Precision-Recall curve . . . 45
3.3.5 Free-Response ROC curve . . . 48
3.4 Experiments and results . . . 50
3.4.1 Comparative analysis: 2D vs 3D models . . . 50
3.4.2 Filtering false positive detection . . . 52
3.4.3 Influence of hyperparameters on nodule detection . . . 55
3.5 Building a Computer-Aided System for patient diagnosis . . . 55
3.5.1 Data challenge context . . . 55
3.5.2 A three-stage pipeline for patient diagnosis . . . 56
3.6 Conclusions and perspectives . . . 57

4 Training neural networks on synthetic microscopy images 59
4.1 Introduction . . . 60
4.1.1 Microscopy imaging . . . 60
4.1.2 The need for data . . . 61
4.2 Modeling microscopy image . . . 62
4.2.1 Nuclei shape . . . 62
4.2.2 Population dispersal . . . 64
4.2.3 Increasing population density . . . 64
4.2.4 Nuclei contouring . . . 66
4.2.5 Texture model . . . 67
4.2.6 Artifacts and noises . . . 67
4.2.7 Simulation scenario . . . 69
4.2.8 Extension for 3D simulation . . . 70
4.2.9 Graphical interface . . . 72
4.3 Models for nuclei segmentation . . . 73
4.3.1 Stardist . . . 73
4.3.2 Cellpose . . . 74
4.4 Segmentation metrics . . . 75
4.4.1 F1-score . . . 75
4.4.2 Jaccard Index (JI) . . . 76
4.4.3 Aggregated Jaccard Index (AJI) . . . 76
4.5 Experiments and results . . . 76
4.5.1 Biomedical image datasets . . . 77

4.5.2 Extraction of simulation parameters from real images 77
4.5.3 Training setup 78
4.5.4 Reducing of the amount of training data 79
4.5.5 Mixing real and simulated data as a training dataset 80
4.6 Conclusions 85

5 Methods for quantification of ovarian cancerous tissues on 3D confocal microscopy 87
5.1 Introduction 88
5.1.1 Ovarian cancer 88
5.1.2 The centrosome 89
5.1.3 Experimental protocol 91
5.1.4 Motivations 91
5.1.5 Strategy for Centrosome-Nuclei Index (CNI) quantification 92
5.2 Methods for centrosome counting 92
5.2.1 Image processing approach 93
5.2.2 Machine Learning approach 95
5.2.3 Deep Learning approach 96
5.2.4 Evaluating centrosome counting models 97
5.2.5 Results and Interpretations 97
5.3 Learning to simulate nuclei cells 98
5.3.1 Problem statement 98
5.3.2 Reinforcement learning 99
5.3.3 Policy gradient 99
5.3.4 A strategy to optimize nuclei cells simulations 100
5.3.5 Application on 2D dataset 100
5.3.6 Application on 3D dataset 102
5.4 Quantifications of Centrosome-Nuclei Index 102
5.5 Conclusions 104

6 Conclusions 105
6.1 Conclusions 105
6.2 Perspectives and future work 106

Bibliography 108

A Appendix I
A.1 Appendix on chapter 3 I
A.2 Appendix on Chapter 5 IV

Liste of Figures

2.1 Evolution of the best classification error rates in the ImageNet Large Scale Visual Recognition Challenge (ILSCVR) competition from 2010 to 2017. The year 2012 marks the first victory of a Deep Learning model. This is followed by steady progress from one year to the next. Extracted from kaggle website. 4

2.2 Formal neuron model introduced by McCulloch and Pitts [1943] with 3 inputs x_1, x_2, x_3. Each input is weighted by the appropriate parameter w_i before being summed. The result of this sum is placed as input of the activation function φ to obtain a final response $\hat{y}$. 5

2.3 A Multi-Layer Perceptron (MLP) composed of an input layer with 6 neurons, two hidden layers with 4 neurons each, and an output layer with 1 neuron. Each layer is fully connected to the previous one and to the next. 7

2.4 2D convolution operation between a 7 × 7 matrix I and a 3 × 3 kernel K. The kernel slides along the input and performs element-by-element multiplication before outputting the sum of all components at each step. Figure extracted from Mohamed [2017] 11

2.5 Application of Sobel filters consisting of a pair of 3 × 3 convolution kernels, giving images with emphasis on edges. 13

2.6 Architecture of the VGG-16 model developed by Simonyan and Zisserman [2014]. The convolutional layers are staged in five blocks, with a pooling layer between every two blocks. This is followed by three fully connected layers, whose role is to estimate a probability for each class on which the model has been trained. Figure generated with PlotNeuralNet repository. . . 14

2.7 Structure of a basic residual block, consisting of successive convolution, batch normalization and Rectified Linear Unit (ReLU) layers. The skip connection associates the input map x with the output of residual layers $\mathscr{F}(x)$ by element-wise addition. The result then passes through a ReLU layer to give the final output of the block. 16

2.8 A 1 × 1 convolution operation with a filter of size 1 × 1 × D and an input layer of size H × W × D, giving an output map of size H × W × 1. A 1 × 1 convolutional layer composed of K kernels yields an output layer of dimension H × W × K. Image credits to Bai's article in Towards Data Science website. 18

2.9 Schematic illustration of U-Net model as presented in Ronneberger et al. [2015]. The left half reduces the spatial resolution with convolution and max-pooling layers, while the right half gradually restores the resolution through convolution and upsampling layers. Skip connections (gray arrows) concatenate features from each side to propagate contextual information from early to deepest layers. 19

2.10 Object detection outputs with a model trained in *Tensorflow* from Huang et al. [2017]. Each result is a bounding box, encompassing an object of class according to a certain confidence score. 20
2.11 An example of the first Haar-like features used by Viola and Jones [2001]. Each rectangle defines a mask where the sum of pixels delimited by the dark area is subtracted from the sum of pixels delimited by the bright area. 22
2.12 Illustration of the Region-based Convolutional Neural Network (R-CNN) model. The R-CNN's workflow consists of four main stages: generating region proposals, resizing them to a fixed size, extracting features with Convolutional Neural Network (CNN) layers and classifying each of them. Image credit to Girshick et al. [2014]. 22
2.13 Framework of fast R-CNN. The input image directly feeds the CNN layers and provides feature maps to which the selective search algorithm is applied. The size of the proposed regions is then standardized by the Region of interest (ROI) pooling layer. The resulting maps are flattened and passed to the Fully Connected (FC) layers, whose final branch performs a classification and regression task. Image source: Girshick [2015] 23
2.14 Illustration of Region Proposal Network (RPN). The RPN uses a 3×3 window that slides over a deep feature map to associate k anchors of different sizes and aspect ratios at each position. It also uses a convolutional layer to convert the CNN output into a new 256-channel output. In this way, each unit is a new feature vector of length 256 that feeds into the box regression layer (reg), which computes the box offsets, and into the box classification layer (cls), which computes the confidence values. For each input pixel (blue dot), the reg layer returns $4k$ outputs, while the cls layer returns $2k$ outputs. Image souce: Ren et al. [2015] 24
2.15 Faster RCNN model from Ren et al. [2015] 24
2.16 YOLO workflow from Redmon et al. [2016]. Input image is divided into S × S cells, each producing B bounding boxes, confidence scores, and C class probabilities. 26
2.17 Architecture of Single Shot Detector (SSD) model. Single Shot detector (SSD) uses VGG-16 as a feature extractor and extends it with additional layers of downsampling convolutions. Each new layer is used for bounding box prediction and allows detection at different scales. Image source: Liu et al. [2016]. 27
2.18 Architecture of Feature Pyramid Network (FPN). FPN comprises a bottom-up and a top-down pathway. The bottom-up pathway refers to the feedforward operation of the backbone. The top-down pathway apply upsampling operation on feature maps of each pyramid stage and merge it with lateral connections. 28

3.1 Estimated number of incidence cases and deaths in 2018 worldwide, all cancers, both sexes, all ages (adapted from International Agency for Research on Cancer 2019). 32
3.2 A collection of pulmonary nodule extracted from the LIDC dataset [Armato III et al., 2011]. Lung nodules are lesions of a more or less spherical shape whose presence can be the direct consequence of lung cancer. 33

3.3 Left: a slice of three-dimensional (3D) Computed Tomography (CT) scan taken from Lung Image Database Consortium (LIDC) dataset. Right: 3D annotations by several radiologists depicting a nodule (blue). From 1 to 8, the scan slices are cropped and sorted in order of depth in the lung. 34
3.4 Architecture of Retina U-net proposed by Jaeger et al. [2018]. The model consists of a Features Pyramid Network (FPN) with a Resnet-50 backbone that enables feature extraction at different scales. A ROI regressor and a classifier are connected to each pyramid level from P_2 to P_5 and provide bounding box output. A segmentation output is also given by applying 1×1 convolution and softmax layer on P_0. 38
3.5 Description of the LIDC dataset after the preprocessing stage described in section 3.2.2. Left: countplot depending on the presence of nodules in each CT scan. Right: histogram of volumes (in voxels) according to nodule type. . 40
3.6 Intersection over Union (IoU) equation. The Intersection Over Union (IoU) metric reflects how close a prediction provided by an object detector is to the target object (the closer to 1, the better the match). 43
3.7 An illustration of true positive (blue), false positive (red), false negative (orange) in object detection task adapted for pulmonary nodule detection. . . 46
3.8 Workflow for computing precision and recall values in a context of nodule detection task, assuming IoU threshold $\tau = 0.5$ and the amount of nodule to detect is $G = 100$. 46
3.9 Precision-Recall curve with 11-points interpolation (left) and all points interpolation (right). The given average precision (AP) score depends on how many precision values are interpolated. 47
3.10 Illustration of ROC and FROC curve. In the medical context, the ROC curves are generally used to measure the quality of a global diagnosis, while the FROC curves show the quality of a detection system that can help in the diagnosis. 49
3.11 A comparison of the inference time for each tested model in 2D and 3D. Average processing time represents the average time required to process a full 3D scan of a patient. The bars represent the standard deviation of the measured times. 51
3.12 Comparison of precision-recall curves between 2D and 3D models on a 5 fold cross-validation applied to the LIDC dataset. 53
3.13 Average FROC curves for all 2D and 3D models tested on LIDC dataset. . . . 53
3.14 Visualisation of a lung scan CT under different views: axial (upper left), sagittal (lower left), coronal (lower right). The segmentation of lungs (brown) obtained with Algorithm 4 was overlaid in each view and the 3D representation of the segmentation is generated in the top right view. Any detection that does not overlay the segmentation is ignored when calculating score performances. This visualization was created using the open source software 3D Slicer. 54

4.1 Human osteosarcoma U2OS cell line visualized by fluorescence microscopy. The left image shows the cytoplasm (green staining) and nuclei (blue staining) of the cell line. The right image is the blue channel showing only the cell nuclei. Image publicly available at Broad Bioimage Benchmark Collection website. 61

4.2 Overview of possible deformation applied to a disc of radius 1. The parameter α controls the magnitude of the distortion, while the parameter σ is a smoothing parameter. For the generation of cell nuclei, forms with values of $\alpha \geq 0.5$ and $\sigma \leq 0.06$ are preferred. 63
4.3 Visualization of cell population dispersion as a function of distance constraint d_{min}. The dotted circles show the minimum distance between a point and its nearest neighbors. Each point then serves as a candidate for the centroid of a newly generated nucleus. 65
4.4 Operation to increase the population density of the nuclei generated by our simulation. At each iteration, the algorithm attempts to add one or more neighbors to each nucleus already present on the image, subject to the maximum overlap condition. 66
4.5 Visual comparison of segmentation masks created for a single nucleus. (a): corresponds to a (binarized) nucleus mask, (b) is obtained by eroding (a) and subtracting the result from (a), (c) is obtained by applying a subtraction to (a) based on a reduction of the initial polygon used for (a). 67
4.6 Left: Gaussian noise, right: Perlin noise. 68
4.7 Visualization of 3 examples of simulated data. Left: Image, right: Ground Truth. Image dimension is 256×256. The simulation parameters used to generate this data are listed in the table above. 71
4.8 A Star Convex shape (left), with local perturbations (right) 72
4.9 The graphical interface of our simulator. This allows to quickly see the effects of each parameter on the final rendering of the simulation. 73
4.10 A visual comparison between the real data (left) and the simulated data (right). Each row is constructed as follows: real image/real ground truth/simulated image/simulated ground truth. Each simulated image/ground truth was obtained by automatic extraction of input parameters as described in section 4.5.2. 78
4.11 A comparison of the evolution of the mAJI score measured on the S-BSST265 and DSB 2018 datasets as a function of the size of the training set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. The red lines are derived from a model trained on a dataset consisting only of real images and the blue lines are derived from one consisting only of simulated images. 80
4.12 Evolution of mAJI score measured on S-BSST265 for Stardist and Cellpose, depending on the composition of the training data set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. Different data augmentation factors are applied to each proportion, either by simulation (left) or by complex transformations of real images (right). The yellow lines are from a data augmentation procedure from both simulations and data transformations, with a factor of 5 for each. 83

4.13 Evolution of mAJI score measured on DSB 2018 for Stardist and Cellpose, depending on the composition of the training data set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. Different data augmentation factors are applied to each proportion, either by simulation (left) or by complex transformations of real images (right). The yellow lines are from a data augmentation procedure from both simulations and data transformations, with a factor of 5 for each. 84

5.1 Representative scheme of the different stages of ovarian cancer. Image credit to Katopodis et al. [2019]. 89
5.2 Consequences of centrosome amplification. A normal cell in interphase (top) contains two centrosomes with two centrioles (green barrels) each. After cytokinesis, two daughter cells with one centrosome each are generated. In the cells with extra centrosome (centriole duplication), two important scenarios have been described. Supernumerary centrosomes fail to cluster and form multipolar spindles that divide in a multipolar manner. This type of division is thought to have a poor outcome for cell survival. In cells where the extra centrosomes cluster to form a bipolar spindle, merotelic attachments (indicated by the red chromosome) can result in lagging chromosomes during anaphase, generating aneuploid cells. These aneuploid cells with unequal chromosome numbers can lead to several pathological conditions such as growth disorders or cancer. Figure adapted from Marthiens et al. [2012]. 90
5.3 A schematic outline of the experimental protocol designed by Institut Curie in Morretton et al. [2019]. 91
5.4 Description of the cohort quantified by Institut Curie. From left to right: counting distributions of nuclei, centrosomes and the resulting CNI for both healthy and tumoral tissues. 92
5.5 Overview of the task of quantifying centrosomes. Automated methods must capable of measuring a sufficient level of colocalization (a) and, for large, inseparable aggregates, providing an approximation of the count (b). Figures adapted from [Morretton et al., 2022]. 94
5.6 Interaction diagram between an agent and its environment in Reinforcement Learning (RL) paradigm. Image credit to Shweta Bhatt's article. 99
5.7 An overview of the reward curves for all experiments performed on the BBBC002-v1 dataset. Left: reward curves for different initialization of simulation parameters. Right: comparison of reward curves based on policy parameter optimization algorithm (SGD vs. Adam). 101
5.8 Comparison of reward curves using a pre-trained or untrained model. . . . 102
5.9 Overview of the results obtained for the CNI estimate for both tissue types. . 103

A.1 A visual comparison of false positive detections obtained after applying NMS algorithm (left) versus WBC algorithm (right). II
A.2 A visual comparison of true positive detections obtained after applying NMS algorithm (left) versus WBC algorithm (right). III
A.3 Performance overview of different approach for centrosome counting. . . . IV
A.4 Performance overview of different approach for centrosome counting. . . . V
A.5 Visualization of the relative error of the CNI estimate for both image and patient level. VI

List of Tables

3.1 Hounsefield Unit (HU) values for some components located in or around the lungs. Data extracted from HU scale wikipedia page. 36
3.2 Overview of AP metrics introduced by Common Object in Context (COCO) challenge [Lin et al., 2014]. 48
3.3 A comparative table of the average number of detections per patient for each model trained and then tested with the LIDC dataset as a function of the minimum confidence threshold s_{min}. 51
3.4 A comparative table of the different models tested with the LIDC dataset. Bold indicates the highest sensitivity for each level of average false positive per scan (FPs). 52
3.5 A comparative table of average CPM and AP scores for each 3D model. These values are obtained after filtering detections outside of lung segmentation. . 55
3.6 AUC score results calculated for each team participating at lung cancer classification challenge during the JFR. Bold indicates our score. N.S. indicates that the team left the challenge. Results extracted from Lassau et al. [2020]. 57

4.1 Summary table of the size of the training dataset used according to the proportion p_{train} of available data for datasets S-BSST265 and DSB 2018. . . . 79
4.2 Summary performance tables (at the object and pixel level) for the S-BSST265 and DSB 2018 datasets with Stardist and Cellpose models trained on real or simulated data only. The values in bold for a given model indicate the best value obtained depending on the type of dataset used. 81
4.3 Composition of the training dataset for S-BBST265 according to the proportion of real data p_{train} and the augmentation factor k_{aug}. 82

5.1 Summary table of counting results for each of the methods used to quantify centrosomes on healthy tissue and tumor tissue. 98

Chapter 1

Introduction

Contents

1.1 General context 1
1.2 Objectives 2
1.3 Thesis organization 2

1.1 General context

The term Artificial Intelligence (AI) in the sense in which it is most commonly used today, encompasses a wide range of methods whose main goal is to transfer knowledge and skills from humans to machines. Historically, the workshop organized by Marvin Minsky and John McCarthy at Dartmouth University in 1956 is the seminal event in this field of research. From that point on, the discipline experienced its first major boom in academia, before fluctuating between several waves of optimism and pessimism until the end of the 20th century. Since the 2010s, AI has seen a resurgence in popularity. This is mainly due to two factors. The first is the significant acceleration in computation times made possible by the use of Graphic Processing Unit (GPU) in modern computers, and their availability at relatively low cost. The second is the amount of data that is being generated at an ever-increasing rate in all application areas.

Machine Learning (ML) is one of the main branches of AI, and as the name suggests, its mechanism gives machines the ability to learn from experience, *i.e.* , by observing data. When that data is labeled, learning resembles an iterative process of adjusting the decisions made by the algorithm to get as many correct answers as possible. This corresponds to what is known as *supervised learning* and is one of the most common applications of ML. Among existing ML methods, Deep Learning (DL) is the class of algorithms that has the reputation of providing the best results on most problems. As a result, this category of models is becoming increasingly popular in industry. In particular, image analysis and computer vision are among the application areas that have developed at high speed thanks to DL.

This last point is particularly interesting for the field of biomedical imaging. Whether it is to diagnose a disease or to try to understand the mechanisms involved in the functioning of living organisms, there is a very wide variety of imaging systems that require extensive and complex analyses. In general, the observation obtained from the images can be quantified either by a class, a location, a count, or even directly by an analysis of the morphological features. Thus, the advantage of using DL methods in this context is to

automate the extraction of this information, overcoming the problems of subjectivity or reproducibility of a study.

1.2 Objectives

Work on this dissertation was carried out under a CIFRE agreement between QuantaCell, a company specializing in the analysis of biological and medical images, and the Institut Montpelliérain Alexander Grothendieck (IMAG). The main objective of this work is to evaluate to what extent methods based on AI, and in particular on DL, can prove to be powerful tools for solving recurrent issues that are encountered in a clinical context. To this end, we will attempt to answer the following research questions:

1. Which DL models are best suited for solving complex tasks such as object detection or image segmentation ? What are their functions and their limitations ?
2. In what configurations can DL be integrated into the solution of a biomedical imaging problem ?
3. How closely do DL models relate to the data fed to them ?
4. Is the use of DL still necessary and/or relevant ?

To answer each question, we present the different application contexts we have entered. Whenever possible, we will use publicly available datasets. However, for this work, a large dataset of ovarian cancer tissue was provided by Institut Curie. The study required complex procedures whose turnaround times were significantly extended by the 2020 health care context.

1.3 Thesis organization

This thesis is organized as follows:

- Chapter 2 reviews the existing literature on Deep Learning models generally used in computer vision. In detail, the individual building blocks that make up a CNN and the operations described in the literature for performing image segmentation or object recognition are explained.
- Chapter 3 describes the use of a framework that employs various modern implementations of object detection models in a context of early pulmonary nodule detection. In addition, this chapter reports on our participation in a Data Challenge that addressed the problem of diagnosing lung cancer.
- Chapter 4 provides a data simulation model to generate various microscopy images and examines the effect of using simulated data as a training set on the segmentation performance of cell nuclei.
- In Chapter 5, a methodology for quantifying ovarian cancerous tissue on 3D microscopy image is developed. The goal is then to perform a count of the major organs of a cell (*i.e.* nucleus and centrosomes in our context) for each image.
- Chapter 6 is a general conclusion that summarizes the entire work, the results obtained and the open research paths during these three years.

Chapter 2

Deep Learning fundamentals for computer vision

Abstract

This chapter is intended to provide the background knowledge necessary to understand this thesis. Section 2.1 is a brief introduction to computer vision and its challenges. We describe how Deep Learning represents an important turning point in the history of this research area. From biological modeling to learning algorithm, Section 2.2 introduces the fundamental concepts on which DL and Artificial Neural Network (ANN) are built. Section 2.3 describes the basic operations performed in Convolutional Neural Network (CNN) and how to move from an architecture that delivers one response per image to one response for each pixel. Section 2.4 provides details on how to perform an object detection task and traces the history of the models used in recent years.

Contents

2.1 Introduction 4
2.2 Deep Learning basics 5
2.2.1 The perceptron 5
2.2.2 Multi-layer perceptron 6
2.2.3 Backpropagation 8
2.2.4 Gradient-based optimization 9
2.3 Convolutional neural network (CNN) 11
2.3.1 Convolutional layer 11
2.3.2 Standard layers of CNNs 14
2.3.3 ResNet architecture 16
2.3.4 Fully convolutional network 17
2.3.5 U-net architecture 19
2.4 CNNs for object detection 20
2.4.1 Performing an object detection task 20
2.4.2 Traditional detectors 21
2.4.3 Region-based CNN (R-CNN) 22
2.4.4 Fast R-CNN and SSP-Net 23
2.4.5 Faster R-CNN and Mask R-CNN 23
2.4.6 You only look once 25

2.4.7 Single shot detector 27
2.4.8 Retina Net 27
2.5 Conclusions 29

2.1 Introduction

Computer vision is an active field of AI designed to enable computers to extract the most relevant information from visual data, be it images or videos. Depending on the application context, the machine uses this information to provide an interpretation of the scenes presented. Ultimately, this discipline has a dual goal: to give a computer the ability to see like a human, but also to understand what it sees. In humans, this process develops naturally in the course of life: our brain experiences different types of stimuli that lead to a strengthening or weakening of neuronal connections and enable to assimilate new information.

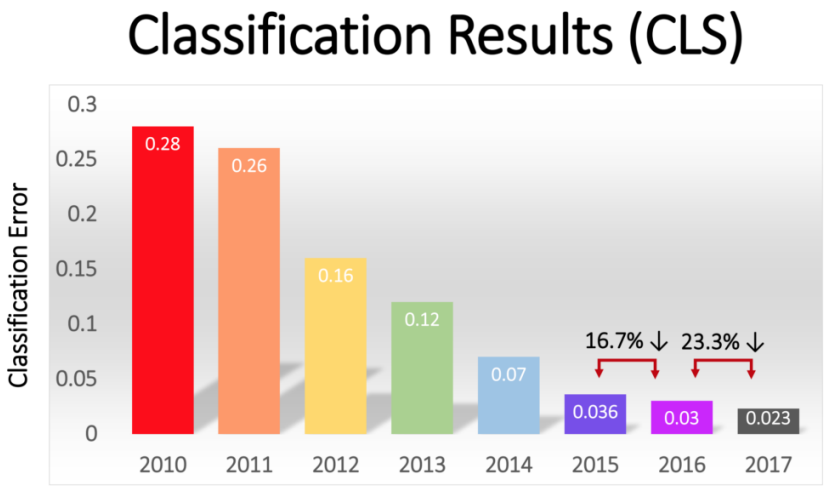


Figure 2.1: Evolution of the best classification error rates in the ILSCVR competition from 2010 to 2017. The year 2012 marks the first victory of a Deep Learning model. This is followed by steady progress from one year to the next. Extracted from kaggle website.

The corresponding procedure for machines consists first in the implementation of algorithms that allow the extraction of image descriptors. These descriptors, commonly referred to as *features*, are low-dimensional representations of data, such as the presence or absence of edges, shapes, or the quantification of pixel intensities. The relevance of a feature to solve a particular problem derives from its discriminative power between different categories of images. Consequently, finding features that are capable of representing the full heterogeneity of the real world is a major challenge in most concrete applications. In addition, there can be some computation time constraints for certain tasks such as object recognition in real time. This means that the development of a computer vision system is a very time-consuming task and could require a certain degree of expertise in the field of application. Without having completely disappeared, these technical difficulties have tended to diminish in recent years with the increasing popularity of methods based on Deep Learning. The 2012 edition of ILSCVR is often credited in the scientific community as the origin of this Deep Learning boom. This event actually corresponds to the great victory of the Alexnet model developed by Krizhevsky et al. [2012] with a classification

error of 15.3 % on the ImageNet database compared to 26.2 % for the second-best team. As shown in Figure 2.1, progress in recent years has resulted in falling below the level of human error.

The key to this success lies in Deep Learning's ability to find by itself the most appropriate features to solve each learning task. This principle is based on advanced mathematical modeling of the mechanisms involved in the function of biological neurons. This fact explains the composition of most Deep Learning models: a sequence of layers of artificial neurons that provide the connection between the input data and the predicted output. The next sections aim to introduce the fundamental concepts on which DL are based. In order to provide sufficient background for a better understanding of this document, the following sections detail the basic concepts upon which DL is built. In particular, the following sections examine the architecture of CNNs models, which are most commonly used in computer vision. We then review the state of the art in models that are particularly useful for visual data segmentation and object recognition.

2.2 Deep Learning basics

2.2.1 The perceptron

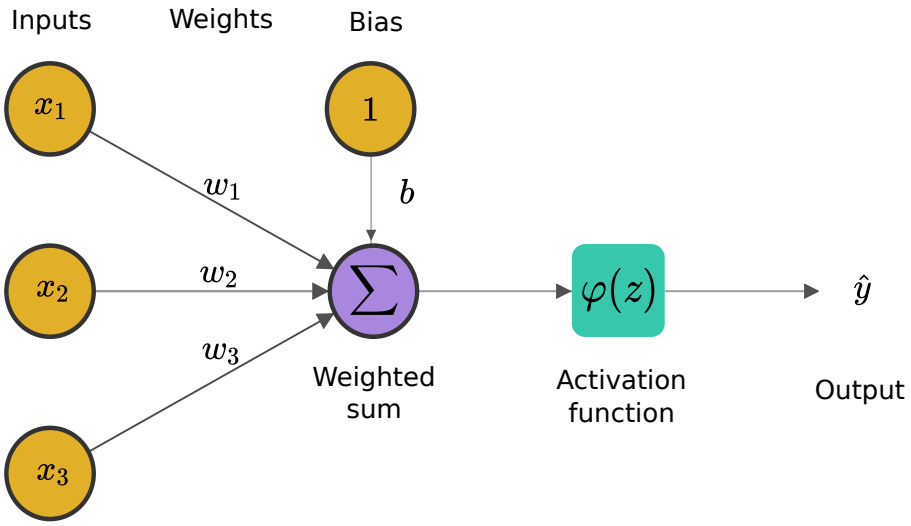


Figure 2.2: Formal neuron model introduced by McCulloch and Pitts [1943] with 3 inputs x_1, x_2, x_3. Each input is weighted by the appropriate parameter w_i before being summed. The result of this sum is placed as input of the activation function φ to obtain a final response $\hat{y}$.

The artificial neuron model, first proposed by McCulloch and Pitts [1943] and largely inspired by the biological model, is the basic unit of a neural network. As shown in Figure 2.2, a neuron consists in four main parts:

- **inputs**: generally represented in the form of a vector $x = (x_i)_{0<i\leq n}$. Each element can be either binary or real.
- **weights**: also represented with a vector $w = (w_i)_{0<i\leq n}$. It corresponds to the importance of each inputs in the resulting output of the neuron. If the x_i input tends to cause the neuron to fire (respectively to inhibit), the corresponding weight w_i will be positive (respectively negative).

- **activation function** (or *transfer function*): This component transforms the input values and introduces nonlinearity into the calculation process. A bias term b can be incorporated in order to delay the triggering of the activation function.
- **output**: it stands for the activation function result.

The operation of the artificial neuron goes through two phases. The first phase requires the use of a summation function. For simplicity, the bias parameter can be thought of as an additional entry x_0 whose value is always 1 and whose weight w_0 has value b. Thus, the linear component of the neuron is the weighted sum $z = \sum_{i=0}^{n} w_i x_i$. The resulting value is then passed to the transfer function, which provides the value of the state of the neuron. In the simplest cases, the activation function is the *Heaviside* function:

$$\varphi(z) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 & \text{else} \end{cases}$$

This function is used especially for binary classification problems. Sometimes it can be useful to have an output with multiple values. Note that unlike biological neurons, whose state is binary, most transfer functions are continuous and have infinitely many possible values in the range $[0,1]$ or in $[-1,+1]$. For example, by using a *sigmoid* activation function $\varphi(z) = 1/(1+\exp(-z))$, all values are between 0 and 1, allowing a probabilistic interpretation of the neuron output.

Learning rule of the perceptron

Let us assume a supervised classification problem with multiple instances of inputs x with targets y, forming a dataset $\mathscr{D} = \{(x,y), x \in \mathbb{R}^d \text{ and } y \in \{-1,1\}\}$. The challenge for any learning model, then, is to find parameters that are suitable for classifying the examples presented. The *perceptron* developed by Rosenblatt [1958] was the first learning implementation based on the neuron model of McCulloch and Pitts [1943]. The algorithm relies on a principle called *Hebb's law*. The rule is grounded on a scientific hypothesis in neuroscience and describes the neural adaptation changes in the brain during a learning process. In summary, this postulate states that when two neurons are excited together, a connection between them is formed or strengthened. The corollary of this law is also true: the less stimulated is a neuronal circuit, the less the neurons of this circuit can connect (or the connection between them is weakened). This leads to feed all the training inputs into the perceptron and to look at the outputs produced. At each time step t, when the perceptron's outputs $\hat{y}$ differs from the correct target y, every weight $w_i(t)$ are updated according to the following rule:

$$w_i(t+1) = w_i(t) + \alpha(y - \hat{y})x_i \ ,$$

where α is a strictly positive real, called the *learning rate*. Simplicity of implementation is the most important property of this algorithm. However, Minsky and Papert [1969] pointed out the main limitation of the algorithm: it is effective only for linear separations, which severely limits the number of real problems that can be solved.

2.2.2 Multi-layer perceptron

The Multi-Layer Perceptron (MLP) is an extensive extension of the perceptron. It consists of multiple neurons arranged in so-called neural networks. A network is defined by its

architecture: the number of neurons, the number of weights, and the arrangement of inputs and outputs. Formally, there is no theoretical rule for the arrangement of neurons. In practice, they are often arranged in an acyclic graph so that the input of a neuron does not depend on its output. Figure 2.3 shows an example of such an architecture. This structure allows information to be passed in the network in only one direction, forward. For this reason, neural networks organized with this topology are also referred to as Feed-Forward Neural Network (FFNN).

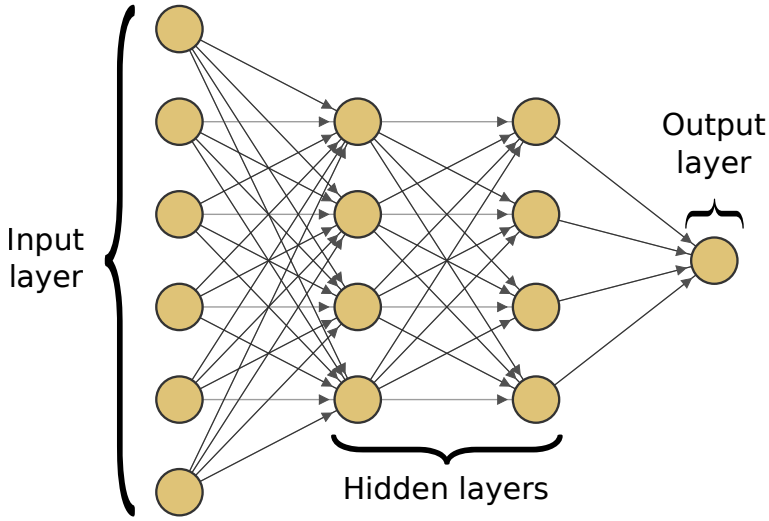


Figure 2.3: A Multi-Layer Perceptron (MLP) composed of an input layer with 6 neurons, two hidden layers with 4 neurons each, and an output layer with 1 neuron. Each layer is fully connected to the previous one and to the next.

MLPs consist of several successive layers of neurons. The first layer is called the input layer, and the last layer is the output layer. All intermediate layers represent the *hidden layers* of the network. The term "hidden" comes from the fact that the outputs of these layers are not directly visible. In a network of N layers, each neuron of layer $l < \mathrm{N}$ is connected only to all other neurons of the following layer $l+1$. Thus, any output from one layer of neurons is only "fed" as input to each neuron in the next layer. This means that each neuron has a number of weights equal to the number of neurons in the previous layer. In addition, the input layer processes the original data, while the subsequent layers process data transformed according to the activation function of the neurons in the previous layer. Depending on the problem, the output layer has a specific activation function, which then transforms the output of the hidden layer into a final output that represents the response of the network.

Forward pass of MLPs

Let consider a MLP composed of L layers. Let n_l be the total number of neurons in the layer $l \in [1, \mathrm{L}]$. The matrix weight W^l groups all the weight values of the layer l and is defined as follows:

$$\mathrm{W}^l = \begin{pmatrix} w^l_{1,1} & w^l_{1,2} & \cdots & \cdots & \cdots & w^l_{1,n_{l-1}} \\ w^l_{2,1} & & & & & \vdots \\ \vdots & & & & & \vdots \\ \vdots & & & & & \vdots \\ w^l_{n_l,1} & \cdots & \cdots & \cdots & \cdots & w^l_{n_l,n_{l-1}} \end{pmatrix},$$

where $w^l_{j,k}$ denotes the weight value for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. We also introduce a vector b^l representing all the biases (intercepts) of the neurons of the layer l. The vector σ^l represents the outputs of all the neurons of the layer l. Since each layer is connected to the previous one, σ^l is also the input of the layer $l+1$. By extension, we denote $\sigma^0 = x$ the input of the network. Let suppose that all neurons in the same layer have the same activation function φ_l, then:

$$\varphi_l(z^l) = \left(\varphi_l(z^l_1), \varphi_l(z^l_2), \ldots, \varphi_l(z^l_{n_l})\right)^\top,$$

is the activation function and z^l is the weighted input of the layer l:

$$z^l = \mathrm{W}^l \cdot \sigma^{l-1} + b^l \ .$$

The output of a layer according to its input can be written as follows:

$$\sigma^l = \varphi_l(z^l) = \varphi_l(\mathrm{W}^l \cdot \sigma^{l-1} + b^l) \ . \tag{2.1}$$

The final output $\hat{y} = \sigma^{\mathrm{L}}$ of a MLP is therefore given by iteratively applying (2.1).

2.2.3 Backpropagation

In practice, it is common to test and compare several network configurations to find a high-performance architecture. To this end, training a model consists of finding a set of parameters θ (combining weights and bias) that, given any input x, make a prediction $\hat{y}$ the closest to the target y. Formally, the training process, regardless of the task to be performed, can be formulated as follows:

$$\min_\theta \mathscr{L}(\hat{y}, y; \theta)$$

where $\mathscr{L}(.)$ refers to the *loss* function (also called *error* or *cost* function) and must reflect how well the task is being performed. In general, the minimization of this function cannot be done analytically, so numerical optimization methods such as *gradient descent* are used. This means that the parameters are updated in small steps in the direction of the negative gradient of the loss function:

$$\theta_{k+1} = \theta_k - \alpha \frac{\partial \mathscr{L}}{\partial \theta_k}(x, \theta_k)$$

where k stands for the iteration number, $\alpha > 0$ is the learning rate. This parameter specifies how far the weights should be shifted in the direction of the gradient. If the learning rate is low, the steps to the minimum of the loss function are small, resulting in being stuck at a local minimum. On the contrary, if the learning rate is high, the training may not converge or may even diverge.

In summary, the learning process involves calculating the gradient of the loss function in order to update the model parameters. Popularized by Rumelhart et al. [1985], the *backpropagation* algorithm expresses the partial derivatives $\frac{\partial \mathscr{L}}{\partial w}$ and $\frac{\partial \mathscr{L}}{\partial b}$ with respect to any weights and bias in the network. These quantities provide information about how the loss function fluctuates when the parameters of the model change. Since $\mathscr{L}$ depends explicitly on the final output $\hat{y}$, it is natural to start by calculating $\frac{\partial \mathscr{L}}{\partial \hat{y}}$. Applying the *chain rule*, this result is then reused in each derivative calculation for each variable preceding $\hat{y}$ in the forward pass. In this way, the gradient of the loss function is propagated backward from the last to the first layer of the network, hence the name "backpropagation".

Let denote δ^l the error vector of the layer l. The error of the j^{th} neuron in the l^{th} layer, δ^l_j, is defined as follows:

$$\delta^l_j = \frac{\partial \mathscr{L}}{\partial z^l_j} \ .$$

For every layer, backpropagation allows computing δ^l and relating those errors to $\frac{\partial \mathscr{L}}{\partial w^l_{j,k}}$ and $\frac{\partial \mathscr{L}}{\partial b^l_j}$. More concretely, the algorithm starts by computing the error in the output layer:

$$\delta^{\mathrm{L}}_j = \frac{\partial \mathscr{L}}{\partial z^{\mathrm{L}}_j} = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{\mathrm{L}}_j} \ .$$

Using (2.1), the second term is simplified:

$$\delta^{\mathrm{L}}_j = \frac{\partial \mathscr{L}}{\partial \hat{y}} \frac{\partial \varphi_{\mathrm{L}}}{\partial z^{\mathrm{L}}_j}(z^{\mathrm{L}}_j) = \frac{\partial \mathscr{L}}{\partial \hat{y}} \varphi'_{\mathrm{L}}(z^{\mathrm{L}}_j) \ .$$

Similarly, we can generalize δ^l for any earlier layer in the following form:

$$\delta^l = \varphi'_l(z^l)(\mathrm{W}^{l+1})^\top \delta^{l+1}. \tag{2.2}$$

Combining (2.2) with the chain rule, we obtain a simple expression for the contribution of any network parameter to the variation in the loss function:

$$\frac{\partial \mathscr{L}}{\partial w^l_{j,k}} = \frac{\partial \mathscr{L}}{\partial z^l_j} \frac{\partial z^l_j}{\partial w^l_{j,k}} = \delta^l_j \frac{\partial}{\partial w^l_{j,k}} \left(\sum_{i=1}^{n_{l-1}} w^l_{j,i} \sigma^{l-1}_i + b^l_j \right) = \delta^l_j \sigma^{l-1}_k \tag{2.3}$$

$$\frac{\partial \mathscr{L}}{\partial b^l_j} = \frac{\partial \mathscr{L}}{\partial z^l_j} \underbrace{\frac{\partial z^l_j}{\partial b^l_j}}_{=1} = \delta^l_j \ . \tag{2.4}$$

2.2.4 Gradient-based optimization

To perform a full training of a model, the backpropagation algorithm must be coupled with an optimization algorithm that provides the "rule" to update the parameters. The vanilla gradient descent approach presented in the previous section, while theoretically valid, might be computationally expensive in practice, especially for large datasets. This is because, in this context, parameters updating occurs only once, when the entire dataset has been used in the forward pass. An alternative method for more frequent updating is the gradient descent algorithm called *stochastic gradient descent* (SGD). This algorithm

evaluates the gradient and updates the parameters using a random subset of the entire data set. This subset is referred to as a *mini-batch*. The gradient is determined sequentially using different mini-batches. The complete run of the algorithm over the entire data set using mini-batches is called *epoch*. Given a learning rate α, the algorithm uses a mini-batch $\mathscr{B}$ to update the parameters at each iteration k as follows:

$$\theta_{k+1} = \theta_k - \alpha \frac{\partial \mathscr{L}}{\partial \theta}(\mathscr{B}; \theta) \quad .$$

Since SGD does not use the entire dataset, but only a portion of it at each iteration, the path took by the algorithm is noisier compared to the gradient descent algorithm. To reduce oscillations along the path towards the optimum, we can add a momentum term to the parameter update. This procedure describes the stochastic gradient descent with momentum (SGDM) algorithm, and the update is:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta \mathscr{L}(\mathscr{B}, \theta_k) + \gamma(\theta_k - \theta_{k-1}) \quad ,$$

where $\gamma \in [0, 1]$ determines the contribution of the previous gradient step to the current iteration.

When the training ends, the optimal parameters θ^*, such as $\frac{\partial \mathscr{L}}{\partial \theta}(x, \theta^*) \approx 0$, should be reached. However, the loss function being optimized is usually strongly non-convex, which implies that θ^* could be a local optimum. This means that SGD (or its variants) requires a suitable initialization of the learning rate, which is not a simple adjustment, as we have seen previously. So far, we have represented the learning rate as a fixed and constant value during training. In practice, this value should be scheduled over time depending on the variations of the loss function. Adam (for Adaptative Moment Estimation) Kingma and Ba [2014] is a popular and robust extension of SGD algorithm allowing to update the learning rate for each network parameter individually. Formally, this optimizer can be seen as a combination of the SGDM algorithm and the RMSP (Root Mean Square Propagation) algorithm described by Tieleman and Hinton [2012]. The learning rate is adjusted based on the estimates of first and second moments of the gradient. Actually, the algorithm computes an exponential moving average of the gradient and the squared gradient:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + \left(1 - \beta_1\right) \nabla_\theta \mathscr{L}(\theta_t) \\ v_t &= \beta_2 v_{t-1} + \left(1 - \beta_2\right) \nabla_\theta \mathscr{L}(\theta_t)^2 \quad , \end{aligned}$$

where m_t and v_t are moving averages at iteration t and the parameters β_1 and β_2 control the decay rates of these moving averages. By default, these values are set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$ in the original paper, and should only be changed in very rare cases. At first iteration, m_0 and v_0 are initialized to 0, which implies that m_t and v_t are biased estimators. Therefore, the algorithm includes a bias correction step and computes $\hat{m}_t$ and $\hat{v}_t$ as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad .$$

Then parameters are updated based on previous moving averages computation:

$$\theta_{t+1} = \theta_t + \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad ,$$

where ϵ is a small value for preventing to divide by 0, generally set at 10^{-6}.

2.3 Convolutional neural network (CNN)

In the previous section, we presented the basic principles on which most neural networks are based. So far, we have assumed that data is represented in terms of vectors, but image data is generally represented in terms of one or more two-dimensional (2D) pixel grids. Therefore, using MLP to deal with computer vision problems means that images must be flattened before showing them to the model. Although this approach works, it does not take into account the spatial relation that may exist between neighboring pixels. This section focuses on one particular family of network designed for this purpose named CNN and introduced by LeCun et al. [1989]. The basic idea of the CNN was inspired by a concept from biology: the receptive field. Receptive fields are a feature of the visual cortex of animals. They act as detectors that are sensitive to specific types of stimuli, such as edges. The specific architecture of the network allows the extraction of features of varying complexity, starting from the simplest to the most complex. The main strength of CNNs lies in their ability to automatically extract features and prioritize them according to the problem. This explains the great popularity of these models, making them an indispensable tool in the field of computer vision.

2.3.1 Convolutional layer

Convolution operation

From a mathematical standpoint, convolution is an operation which, from two functions f and g defined on the same domain, produces a third function s such as:

$$s(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)\,\mathrm{d}\tau = \int_{-\infty}^{+\infty} f(t-\tau)g(\tau)\,\mathrm{d}\tau \quad . \tag{2.5}$$

As the convolution is very connected to multiplication, we use the symbol $*$ to symbolize this operation:

$$s(t) = (f * g)(t) = (g * f)(t) \quad .$$

One way to interpret this operation intuitively is to view it as a generalization of the moving average concept. As τ changes, we measure the overlap between f and g when a function is flipped and shifted by t. Although we use the variable t in these equations, this operation can be used for both temporal and spatial data.

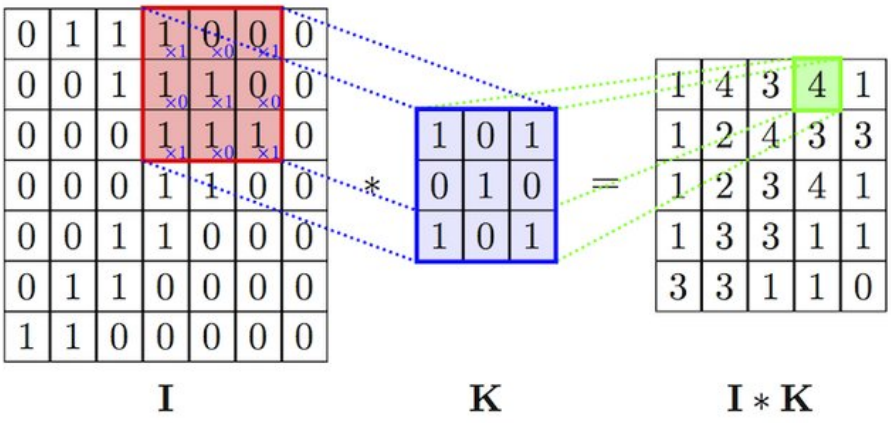


Figure 2.4: 2D convolution operation between a 7×7 matrix I and a 3×3 kernel K. The kernel slides along the input and performs element-by-element multiplication before outputting the sum of all components at each step. Figure extracted from Mohamed [2017]

Regarding images data, these are rather conceptualized as *tensors*. For instance, an image $\mathrm{I} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{C}}$ is a tensor of order 3 containing $\mathrm{H}\times\mathrm{W}\times\mathrm{C}$ elements (*i.e.* pixels) that we can index by a triplet (i, j, k) with $1 \leq i \leq \mathrm{H}$, $1 \leq j \leq \mathrm{W}$, and $1 \leq k \leq \mathrm{C}$. From a computer point of view, we can consider a tensor of order 3 as an array that includes C matrix channels of height H and width W for each of them. Tensors are particularly useful for representing images, whether grayscale images ($\mathrm{C} = 1$) or color images stored in a Red-Green-Blue (RGB) format ($\mathrm{C} = 3$). In this context, it seems more natural to introduce a discrete version of equation (2.5):

$$\mathrm{S}(i, j) = (\mathrm{I} * \mathrm{K})(i, j) = \sum_m \sum_n \mathrm{I}(i-m, j-n)\mathrm{K}(m, n) \ , \tag{2.6}$$

where I traditionally denotes the input image, K is a convolution *filter* (also called *kernel*) of size $m \times n$, and the output S is referred as a *feature map*. This operation, as described in equation (2.6), represents the main building block of a CNN. Concretely, it consists in sliding the kernel along each input dimension, from top to bottom and from left to right. At each step, the kernel is applied to the input by applying dot product, outputting a single scalar value. Figure 2.4 depicts an explicit visualization of such procedure, usually referred to as *conv* 3×3 with reference to the kernel size. Note that equation (2.6) corresponds to the operation in a 2D convolutional layer, but the principle for extending it to n dimensions is quite similar.

A convolutional layer usually contains a number of kernels of the same size, each outputting a different feature map. It is common to stack each of the outputs in a single block whose number of channels actually matches the number of feature maps. As for the size of the other dimensions (height and width) they depend on the values of three hyperparameters:

- **kernel size**: In 2D convolution, the kernel can slide in only two directions: horizontally or vertically. The process works only if the filter fits completely into the input, and it is completed when the kernel reaches the bottom right edge of the matrix. So the larger the filter is, the more the maximum number of operations to be performed is reduced.
- **padding**: When no other processes are involved, the successive application of multiple convolutional layers gradually reduces the size of the original maps. In this way, the information at the edges of the image becomes deteriorated layer after layer. The padding process then consists of increasing the size of the input image by adding extra pixels at the edges. In general, all these pixels are set to 0.
- **stride**: In contrast to padding, increasing the value of stride allows for a reduction in the outputs of each convolutional layer and thus more efficient computation. The parameter thus controls the step size that the filter takes during convolution.

Finally, for a fixed dimension d_{in} (height or width), if we note k the corresponding filter size, p the padding value and s the stride value, the dimension of the output feature map d_{out} can be calculated as follows:

$$d_{out} = \left\lfloor \frac{d_{in} + 2p - k}{s} \right\rfloor + 1 \ .$$

Convolution's benefits

In computer vision, convolution produces various visible effects on images. For example, Figure 2.5 shows how multiple convolution filters can detect edges in an image. Convolution filters can be used to detect the presence of a set of features in images received as

input. Following the concepts discussed in the previous sections, the resulting features map can be referred as an *activation* map, where the highest values correspond to the presence of the feature described by the filter. In general, the first convolutional layers of a CNN learn less complex features, such as those activated for contours with a certain angle. As we go deeper into the network, the learned features become more complex. In the case of edge detection, the deepest layers are activated in the presence of a group of edges that form a certain shape.

The main limitation of MLPs in image processing is that each neuron is fully connected to the neurons of the previous and the next layer, which can lead to difficulties in managing the image size. For example, an RGB image of size 200×200 means $200 \times 200 \times 3 = 120000$ connections for a single neuron. The number of connections therefore easily explodes for fully connected multilayer networks. A major advantage of CNNs over MLPs is the significant reduction in the number of connections between neurons. CNNs are also inspired by biological processes that occur in different regions of the brain. In this case, they were inspired by the functioning of sensory neurons in the visual cortex. The latter have the peculiarity that they are stimulated only by a limited area of the visual field, called the receptive field. The receptive fields of the different neurons partially overlap so that they cover the entire visual field.

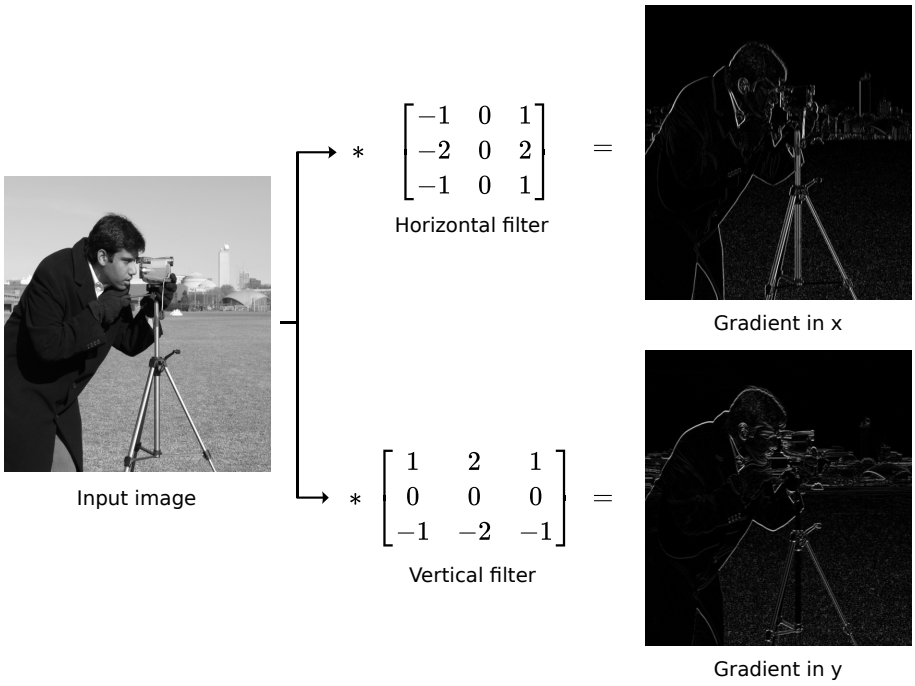


Figure 2.5: Application of Sobel filters consisting of a pair of 3×3 convolution kernels, giving images with emphasis on edges.

The principle is similar with CNNs and gives them fundamental properties that distinguish them from MLPs. Indeed, the size of the filters requires local connectivity of the neurons from one layer to the next. During the training phase, this allows the filter to provide the strongest responses to a spatially limited input region. Another advantage of using CNNs is that each filter is replicated across the entire input image. This means that the parameters of neurons located in the same feature maps are shared. This enables the detection of specific patterns regardless of their position in the image. Thus, CNNs also have a property of translational invariance. By combining these properties, the number

of parameters to be optimized can be significantly reduced. This implies better statistical estimation than MLPs for the same amount of data. In addition, the memory requirement is lower, which allows the construction of larger and generally more powerful networks.

2.3.2 Standard layers of CNNs

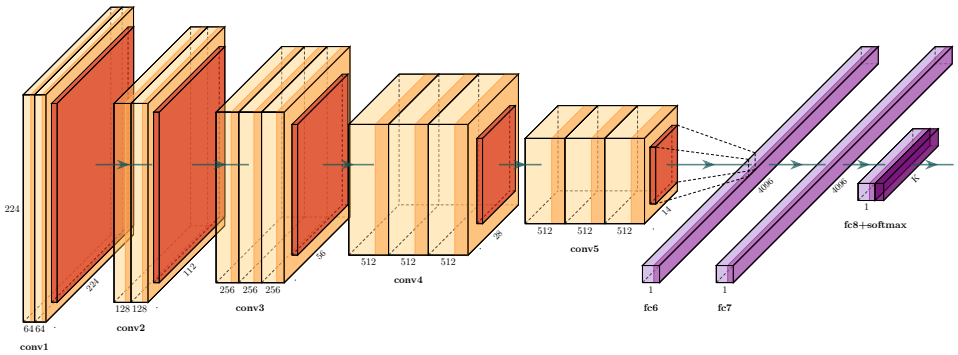


Figure 2.6: Architecture of the VGG-16 model developed by Simonyan and Zisserman [2014]. The convolutional layers are staged in five blocks, with a pooling layer between every two blocks. This is followed by three fully connected layers, whose role is to estimate a probability for each class on which the model has been trained. Figure generated with PlotNeuralNet repository.

A CNN consists of several successive convolutional layers, generally interspersed with layers of subsampling. This sequence is usually followed by one or more layers fully connected. Figure 2.6 illustrates a common architecture of CNN, with 16 layers. In this part, we will describe the operation performed by some non-convolutional layers.

Rectified linear unit (ReLU) layer

Most CNNs use ReLU as activation function for the hidden layers. The ReLU correction layer replaces all negative values received as inputs with zeros:

$$\mathrm{ReLU}(x) = \max(0, x) \quad .$$

Using the ReLU activation function rather than sigmoid function in the hidden layers of a CNN can be motivated for several reasons:

- **Fast and easy computation**: The result is really easy to calculate and inexpensive in terms of computation time. CNNs with ReLU layers have a faster forward pass.
- **Faster convergence**: Sigmoid activation functions have an extremely low gradient when input becomes large. This can cause the updating phase of the weights to slow down or even stop altogether during training. This problem is called the *vanishing gradient*. With a constant gradient, the ReLU function is less sensitive to this problem, thus showing faster convergence [Krizhevsky et al., 2012].
- **Sparsity activation**: By "turning off" the neurons with a negative outcome, it is more likely that the remaining neurons are associated with important aspects of the supervised task. The predictive power is thus improved.

Batch normalization layer

In the most general sense, normalization is a method of adjusting the value of numerical data to a typical scale while preserving differences in the range of values. Analogously, performing *batch normalization* (BN) [Ioffe and Szegedy, 2015] allows to train a CNN by standardizing inputs layer for each mini-batch. Thus, for a mini-batch $\mathscr{B}$ formed by m inputs $\{x_i\}_{i=1,\dots,m}$, the empirical mean $\mu_{\mathscr{B}}$ and variance $\sigma^2_{\mathscr{B}}$ are computed as follows:

$$\mu_{\mathscr{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \sigma^2_{\mathscr{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathscr{B}})^2 \ .$$

The standardization process centers and scales inputs individually:

$$\hat{x}_i = \frac{x_i - \mu_{\mathscr{B}}}{\sqrt{\sigma^2_{\mathscr{B}} + \epsilon}} \ ,$$

where $\epsilon > 0$ is a small constant added for numerical stability. Then, in order to preserve the expressive power of the model, the BN layer introduces extra trainable parameters β, γ to re-scale and shift inputs:

$$\hat{x}_i = \gamma \hat{x}_i + \beta \ .$$

The use of a BN layer allows to insert a regularization process into CNNs, as well as stabilization of the gradient values during backpropagation [Santurkar et al., 2018]. Consequently, sharing the same scale of values for each batch speeds up training time, increases the generalization of the model and reduces the risk of overfitting.

Pooling layer

Pooling operators are similar to convolution in the sense that they apply a sliding window to all regions of the input based on a given stride value and compute a single output at each position. Pooling layers, however, do not contain parameters to be optimized and always apply the same operation regardless of the input on which they are based. In general, this involves calculating either the average or the maximum of the elements covered by the sliding window (this is referred to as *average pooling* or *max pooling*, respectively). Intuitively, the purpose of a pooling layer is to aggregate the information available in the feature maps while achieving a more manageable dimensionality reduction. The relevant information with the highest activation values is then distributed across the layers, while information that is not needed is progressively ignored. In this way, the receptive field of the CNN increases layer by layer, as does the complexity and level of recognized features.

Fully connected layer

The FC layers are usually located at the end of the CNN architecture and form the classification block. These layers behave in exactly the same way as that of the MLPs introduced in Section 2.2.2. As a result, feature maps of convolutional layer are flattened in a one-dimensional form before entering the first FC layer. FC layers determine the link between the position of features in the image and a class. When a particular feature combination of a class is identified, significant weights are assigned to the corresponding neurons. The last FC layer classifies the input image of the network. The privileged activation function $\varphi : \mathbb{R}^K \longrightarrow [0,1]^K$ in this configuration is the *softmax* function defined for each component

$\varphi(z)_j$ as:

$$\varphi(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad , \tag{2.7}$$

where K is the number of classes in any classification problem, $z \in \mathbb{R}^K$ is the input vector of the last FC layer. After applying (2.7), each element will be in range $[0,1]$ and the normalization ensures $\sum_{k=1}^{K} \varphi(z)_k = 1$, providing a probabilistic interpretation to the output vector $\hat{y}$.

2.3.3 ResNet architecture

Since Alexnet's victory at ILSCVR 2012 [Krizhevsky et al., 2012], its architecture, consisting of 5 convolutional and 3 FC layers, has been considered a reference for the scientific community. While the ZFNet model [Zeiler and Fergus, 2014], winner of ILSCVR 2013, can rather be considered as an optimization of the Alexnet architecture, competitors of the subsequent editions have increased the number of layers to improve performance. Thus, Simonyan and Zisserman [2014] developed VGG16-Net, which outperforms AlexNet and ZFNet on ImageNet dataset, and, as its name suggests, has 16 layers. Also, the winner of the 2014 edition, Googlenet [Szegedy et al., 2015] goes through 22 layers. Nevertheless, He et al. [2016] has shown that increasing the depth, *i.e.* , stacking more and more layers is not enough to improve a model. On the contrary, above a certain depth, adding plenty of new layers can degrade performance. This is the direct consequence of the well known vanishing/exploiding gradient problem, Due to the backpropagation algorithm, the more layers a network has, the smaller the gradients of the parameters of the early layers become. This leads to slowing down or, in the worst cases, even stopping the training.

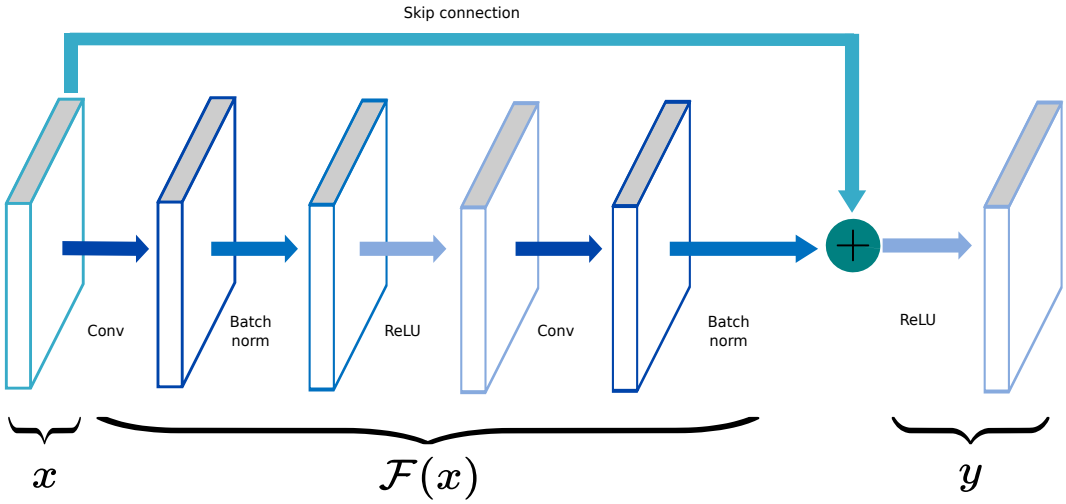


Figure 2.7: Structure of a basic residual block, consisting of successive convolution, batch normalization and ReLU layers. The skip connection associates the input map x with the output of residual layers $\mathcal{F}(x)$ by element-wise addition. The result then passes through a ReLU layer to give the final output of the block.

Winners of ILSCVR 2015, He et al. [2016] fix the problem with an architecture called ResNet. These models consist of very deep neural network whose main building blocks are *residual connections* (or residual blocks). A residual block is a group of layers configured to connect the output of one layer to another deeper layer within the block. This

connection is commonly called a *shortcut* (or *skip connection*) because it jumps across multiple layers, as shown in Figure 2.7. Mathematically, the forward pass of a residual block can be written as:

$$y = \varphi(\mathscr{F}(x) + \mathrm{Id}(x)) = \varphi(\mathscr{F}(x) + x) \ ,$$

where x and y are respectively the input and output features of the block, $\varphi(\cdot)$ is any activation layer, $\mathrm{Id}(\cdot)$ is the identity function, and $\mathscr{F}(\cdot)$ is a combination of nonlinear transformation operated by a sequence of any traditional layers. In this way, the identity function induced by the shortcut connection allows the gradient to be maintained during the backpropagation pass. Moreover, no additional parameters or computational overhead are added.

2.3.4 Fully convolutional network

All models presented in previous sections are particularly suitable for image classification tasks where only one response per image is expected. However, some computer vision applications, such as autonomous drive, require a more detailed understanding of the scenes under investigation. An obvious step after image classification would be a pixel-level classification. This is an approach called *semantic segmentation*. The goal now is to create a model whose response is multidimensional and the same size as the input image presented to it. This response map is commonly referred to as a *segmentation mask* and assigns each pixel a value from a predefined set corresponding to an object class. For CNNs such as those presented in Section 2.3.2, the solution would be to use a sliding window strategy. However, this type of approach is not really satisfactory, mainly because it requires inference of the CNN for each pixel of the input image, which drives up the execution time and also introduces low computational efficiency. Actually, architectures of CNNs consisting of convolutional layers and FC layers are incompatible with a segmentation aim. This is mainly because the FC layers perform flattening operation of feature maps that ignore the spatial relation between pixels in images. In addition, the use of FC layers requires the settings of a single input size for the input images. The Fully Convolutional Network (FCN) [Long et al., 2015] is the first striking attempt to restructure CNN's architecture to resolve semantic segmentation tasks. Basically, a FCN is equivalent to a CNN without FC layers.

1×1 convolution

First, the flattening operation and the FC layers are removed and replaced by a 1×1 convolutional layer. As the name implies, this type of layer is equivalent to applying convolution with a filter of size $1 \times 1 \times D$, where D denotes the input depth. This process is illustrated by Figure 2.8. In general, no padding is added, and the stride is set to 1. Unlike layers with a higher kernel size, using a 1×1 convolution layer preserves the spatial dimensions (*i.e.* height and width) of the input tensor while reducing the dimensionality in channel space.

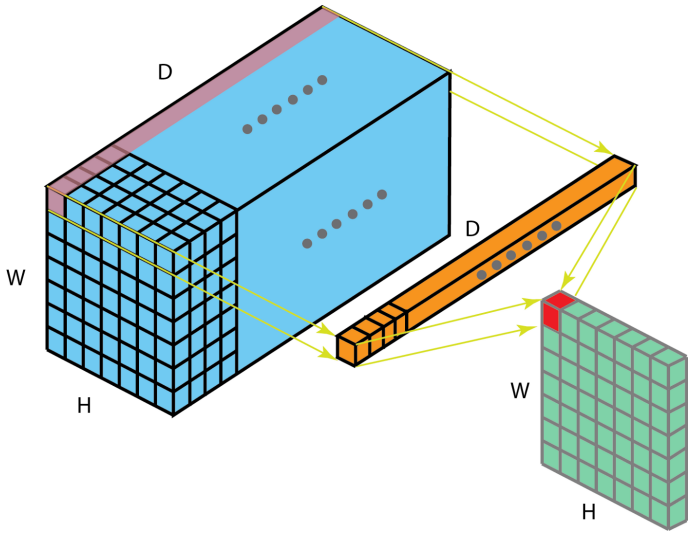


Figure 2.8: A 1 × 1 convolution operation with a filter of size 1 × 1 × D and an input layer of size H × W × D, giving an output map of size H × W × 1. A 1 × 1 convolutional layer composed of K kernels yields an output layer of dimension H × W × K. Image credits to Bai's article in Towards Data Science website.

At this stage, the final output of the model is a feature map whose size is independent of that of the input image. Nevertheless, the resolution was successively reduced with the multiple pooling layers (or any other subsampling operation). For example, the CNN shown in Figure 2.6 begins with a 224 × 224 input layer and decreases the resolution by a factor of 32, reaching 7 × 7 just before FC layers. These features map therefore contain a lot of general information about the image, but are too coarse to produce a high quality segmentation.

Upsampling convolution and skip connections

To complete the conversion of CNN classifications into a segmentation model, the coarse results must be restored to the original resolution. A classic way for solving this problem in image processing is to perform upsampling by interpolation. The simplest methods to use are nearest neighbor interpolation and bilinear interpolation. In the first method, an image is enlarged by a factor f by replacing each pixel with a block of size $f \times f$ with the same value. In the second method, each pixel values are obtained by linear interpolation in each direction. Although these approaches are acceptable, none of them have parameters that can be learned. To take a "fully convolutional" perspective, an upsampling operation with f-factor can also be viewed as a backward convolution (or *deconvolution*) with fractional stride $1/f$ [Long et al., 2015]. Note that the term "*transposed convolution*" is preferred in the DL community for this type of layer. This means that FCNs also learn to convert an image from low to full resolution. However, to achieve high precision, it makes more sense to proceed stepwise and incorporate multiple upsampling layers. Thus, it is also possible to use *skip connections* in the model architecture. Analogous to the connections presented in Section 2.3.3, the feature maps "jump" across multiple layers and are not simply passed on to the next. In this way, the features from the downsampling path, which are full of contextual information, can be connected to the features of the upsampling path. Generally, the connection can be done by simple summation or concatenation.

2.3.5 U-net architecture

The U-net model designed by Ronneberger et al. [2015] is one of the most popular architectures for semantic segmentation tasks. The model was developed in the context of a biomedical imaging application, but it also performs very well in other application domains [Iglovikov et al., 2017; Li et al., 2018; Tran and Le, 2019]. The authors have used this architecture on several segmentation challenges and have shown that the model requires only a few training images (< 40 labeled images) to significantly outperform the precision of the state of the art. This is mainly due to the quasi-symmetry form of the model, which allows the transfer of contextual information to the upsampling layers.

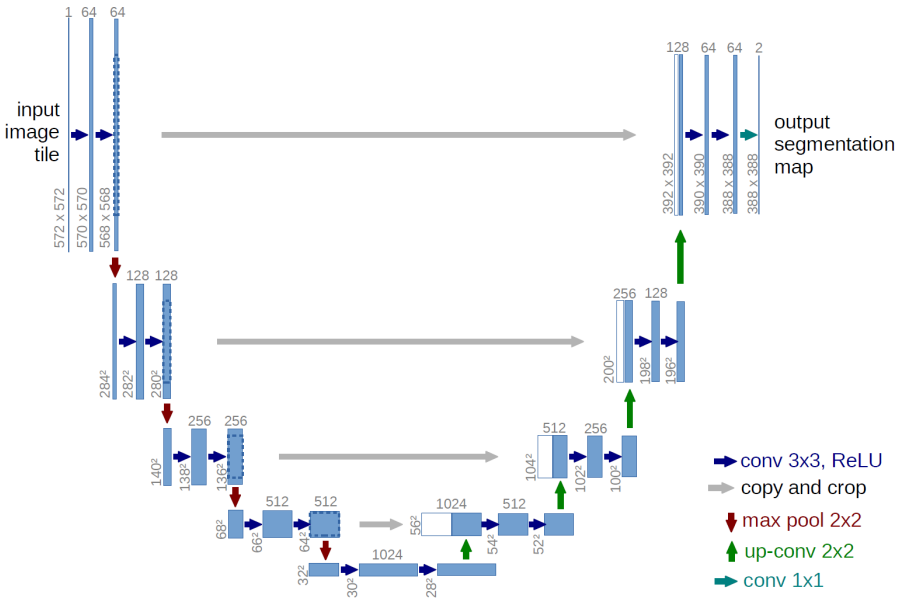


Figure 2.9: Schematic illustration of U-Net model as presented in Ronneberger et al. [2015]. The left half reduces the spatial resolution with convolution and max-pooling layers, while the right half gradually restores the resolution through convolution and upsampling layers. Skip connections (gray arrows) concatenate features from each side to propagate contextual information from early to deepest layers.

Based on FCN architecture, the original U-net implementation, illustrated by Figure 2.9, is mainly divided into 3 parts:

- **the contracting path** (also called *encoder*): Similar to a CNN for classification, this part of the model extracts features from the input image. This results in a sequence of four downsampling blocks, each consisting of two consecutive convolutions layer with ReLU activation, followed by a max-pooling layer. After passing through a block, the spatial dimensions are approximately halved, while the number of feature maps has doubled.
- **the bottleneck**: Between the downsampling and the upsampling part, two 3×3 convolutional layers followed by 2×2 transposed convolution are inserted. The intuition behind using this block is to transmit a compressed view of the input that contains only useful information to better learn the reconstruction in the following block.
- **the expanding path** (also called *decoder*): This part can be seen as a mirror of the contracting path and counteracts it. The max-pooling layers are replaced by transposed convolutional layers. In this way, the spatial dimensions gradually increase

again while the number of feature maps is halved. Each upsampled output is concatenated in the feature channel dimension with the corresponding cropped map from the contracting path. Finally, the last layer is a 1×1 convolution applied to the output of the expanding path. Then, a sigmoid function is applied for each pixel, providing some sort of confidence value for belonging to an object of interest.

2.4 CNNs for object detection

2.4.1 Performing an object detection task

Object detection is one of the classical problems of computer vision. It involves finding, individualizing, and categorizing all instances of an object present in a given image. This is a common task in the development of self-driving cars, but also in the context of object recognition in medical imaging. The complexity of the problem is twofold: on the one hand, the object in question must be recognized and classified, and on the other hand, it is also necessary to locate it accurately. For visualization, Figure 2.10 represents the results of an object recognition model.

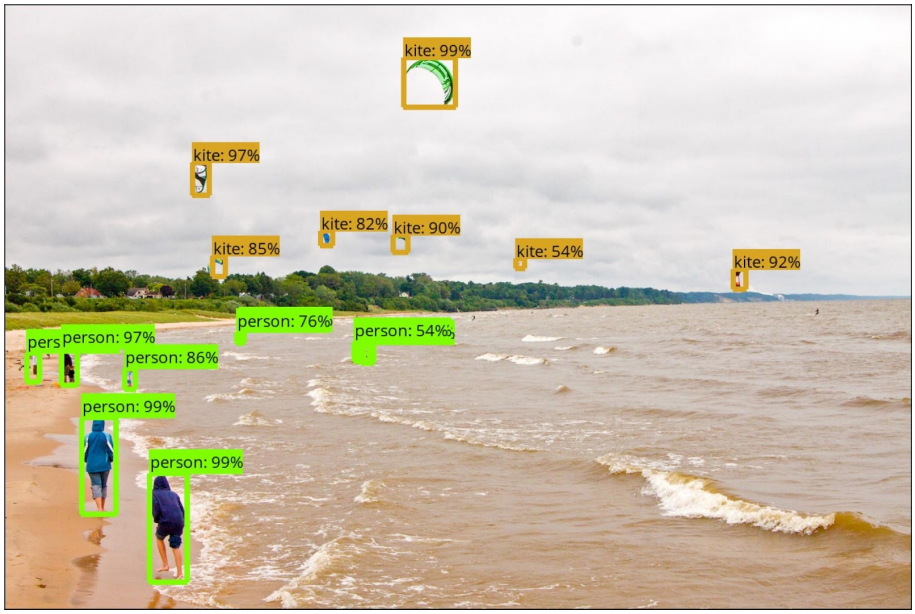


Figure 2.10: Object detection outputs with a model trained in *Tensorflow* from Huang et al. [2017]. Each result is a bounding box, encompassing an object of class according to a certain confidence score.

Given an object detection system f and a 2D image $\mathrm{I} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}}$, an object localization process results in a list B of size $\mathrm{N} \in \mathbb{N}$:

$$\begin{aligned} f\colon \quad \mathbb{R}^{\mathrm{H}\times\mathrm{W}} &\to B \\ \mathrm{I} &\mapsto \{b_1, b_2, ..., b_{\mathrm{N}}\} \ , \end{aligned}$$

where each element $b_i \in B$ is a tuple defining a *bounding box* (containing the targeted objects):

$$b_i = (b_x, b_y, b_w, b_h, c_i, s_i) \ ,$$

with:

- b_x, b_y define the center coordinates of the bounding box.
- b_w, b_h refers to the width and height of the bounding box.
- $c \in \mathscr{C}$ is the *object class*, where $\mathscr{C} = \{1, 2, \ldots, C\}$ is the set of bounding boxes classes, and $n_c \geq 1$ is the number of existing object class.
- $s \in [0, 1]$ is a *confidence score* associated with the predicted class c.

Using vanilla CNN is not appropriate for solving this type of problem. The first obstacle arises from the fact that the number of objects to be recognized is not known a priori, so it is impossible to specify a size of the output layer. One way around this difficulty would be to apply a CNN to different regions of interest, but the variety of sizes and locations of objects to be detected make this procedure hardly useful. In the next sections, we describe modern models of object detectors that represent two different approaches: single-level detectors or two-level detectors.

2.4.2 Traditional detectors

Before the rise in popularity of Deep Learning methods in the 2010s, most computer vision algorithms relied on handmade representations of images. This involved expertise for extracting quality feature vectors from input images and feeding classifier models to it. This section presents the most popular features extraction procedure before the emergence of deep learning-based models.

Scale-Invariant Feature Transform (SIFT)

Developed by Lowe [1999], the SIFT algorithm detects and identifies a set of similar elements between images. The process consists of collecting key points localization from a library of images and extracting local descriptors from it for building a reference database. The resulting features are assumed invariant to image translation, rotation, and scaling, which facilitates discrimination between each category of objects. Object detection is then performed on SIFT features and matching it with stored features by measuring Euclidean distance.

Viola-Jones Detector

The Viola-Jones (VJ) detector [Viola and Jones, 2001] is a complete framework proposing real-time face recognition. The approach consists of using sliding windows across space and scales and extracting Haar-like features at each step. As illustrated in Figure 2.11, Haar-like features are computed by considering adjacent rectangular regions in detection window and summing up the pixel intensities in each region. Computing the difference between sums provides useful information (edges, straight/diagonal lines, etc.) for object identification. For the training step, the authors designed a cascade of classifiers allowing to eliminate most negative bounding boxes and boosting the model speed.

Histogram of Oriented Gradient (HOG)

Dalal and Triggs [2005] proposed the Histogram of Oriented Gradients (HOG) feature descriptor in a context of pedestrian detection on 2D images. It leverages the description of intensity gradient or edge detection distributions to perform detection. The method starts by splitting the input image into small connected regions called *cells* and compute a histogram of gradient directions for all pixels in the corresponding cell. Adjacent cells

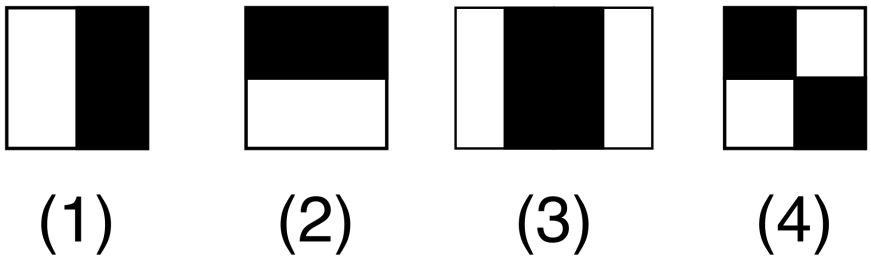


Figure 2.11: An example of the first Haar-like features used by Viola and Jones [2001]. Each rectangle defines a mask where the sum of pixels delimited by the dark area is subtracted from the sum of pixels delimited by the bright area.

(and associated histograms) are then concatenated and constitute a block, which will be normalized for being independent of lighting variations. The HOG descriptors can then be used to train a linear Support Vector Machine (SVM), though other supervised classifiers are suitable as well.

2.4.3 Region-based CNN (R-CNN)

In 2014, Girshick et al. [2014] pioneered the concept of Region-based Convolutional Neural Network (R-CNN), which is at the origin of the two-stage models family based on Deep Learning. The model can be seen as a pipeline made up of four disjoint stages, illustrated by Figure 2.12. The first stage of the method relies on candidate regions generation. Commonly, this step consists of hierarchically grouping pixels, a step often called *selective search* (SS) [Van de Sande et al., 2011] leading to about 2000 regions per input image. Proposals regions are then re-adapted to a fixed size, and given as inputs for fine-tuning a binary classification CNN (background vs. foreground region). The latter outputs a feature vector used for training a set of binary SVM and classify each region according the presence of each object class. Although this method is considered as a precursor in deep learning-based object detectors, it remains relatively pricey to use. Indeed, forwarding so many warped regions in a CNN considerably lengthens the training and testing time.

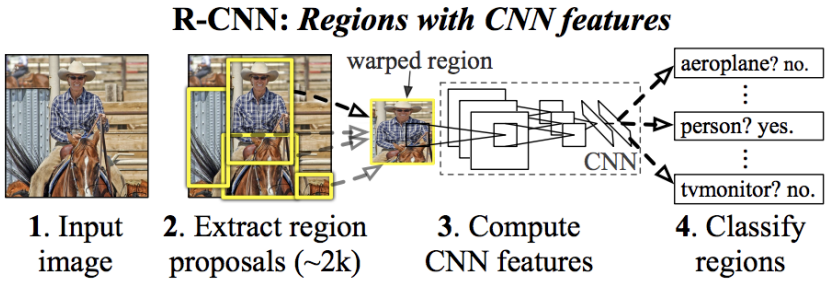


Figure 2.12: Illustration of the R-CNN model. The R-CNN's workflow consists of four main stages: generating region proposals, resizing them to a fixed size, extracting features with CNN layers and classifying each of them. Image credit to Girshick et al. [2014].

2.4.4 Fast R-CNN and SSP-Net

Numerous improvements have been made in terms of speed and performance throughout the years. He et al. [2015] and Girshick [2015] respectively introduced Spatial Pyramid Pooling Network (SPP-Net) and Fast R-CNN, which improved the initial workflow. Both models feed whole images into convolutional layers first. Next, the SS algorithm is applied to the resulting feature maps and yields Regions Of Interest (ROI), which are a set of sections of uneven size of the feature maps. In order to insert FC layers, it is then necessary to provide a fixed size representation of each region. Following this principle, SPP-Net uses a *spatial pooling* layer that consists in applying pooling operations with multiple windows at different scales. Fast R-CNN, on the other hand, uses a ROI pooling layer, which implies to apply max-pooling with a single window, but whose size depends on the desired output size. Thus, both models convert each region into a fixed-size vector that is transferable for the FC layers to perform a classification task. Note also that Fast R-CNN has an additional regression branch to learn the offset between each ROI and the corresponding bounding box of the detected object, as depicted in Figure 2.13. These adjustments allow unifying independent models in R-CNN (regressors and classifiers) and can lead to gains of up to one order of magnitude in terms of computational efficiency. However, region proposals generation is still based on external greedy algorithm, in the sense that SS algorithm takes local decision to solve the global problem. It starts by a graph-based sub-segmentation method [Felzenszwalb and Huttenlocher, 2004], and then aggregate recursively regions according to some similarity criteria (color, texture, size, etc.). The main issue for R-CNN lies in the fact that region proposal is time-consuming (from 1.8 to 3.7 sec per image).

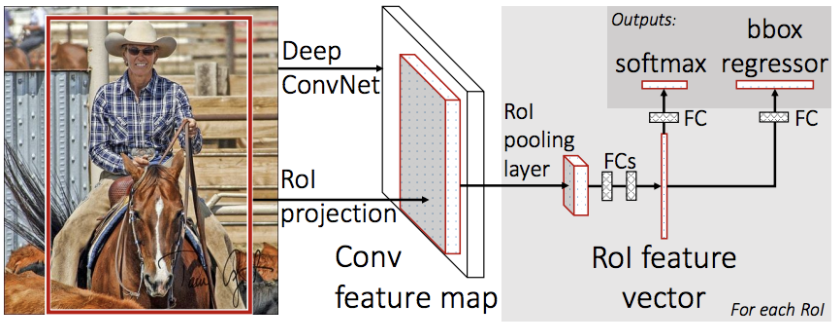


Figure 2.13: Framework of fast R-CNN. The input image directly feeds the CNN layers and provides feature maps to which the selective search algorithm is applied. The size of the proposed regions is then standardized by the ROI pooling layer. The resulting maps are flattened and passed to the FC layers, whose final branch performs a classification and regression task. Image source: Girshick [2015]

2.4.5 Faster R-CNN and Mask R-CNN

Faster R-CNN Ren et al. [2015] eventually replaced the bottleneck of the SS algorithm with Region Proposal Network (RPN). A RPN is basically a CNN trained to output rectangular region proposals associated with a score indicating how surely the proposal contains an object, regardless of its class. In practice, the regions are generated from some predefined reference boxes, which are called *anchor boxes*, and RPNs actually predict an offset from their coordinates.

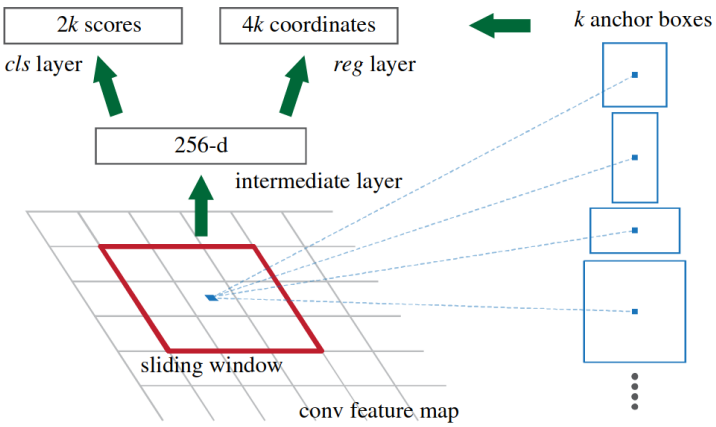


Figure 2.14: Illustration of Region Proposal Network (RPN). The RPN uses a 3×3 window that slides over a deep feature map to associate k anchors of different sizes and aspect ratios at each position. It also uses a convolutional layer to convert the CNN output into a new 256-channel output. In this way, each unit is a new feature vector of length 256 that feeds into the box regression layer (reg), which computes the box offsets, and into the box classification layer (cls), which computes the confidence values. For each input pixel (blue dot), the reg layer returns $4k$ outputs, while the cls layer returns $2k$ outputs. Image souce: Ren et al. [2015]

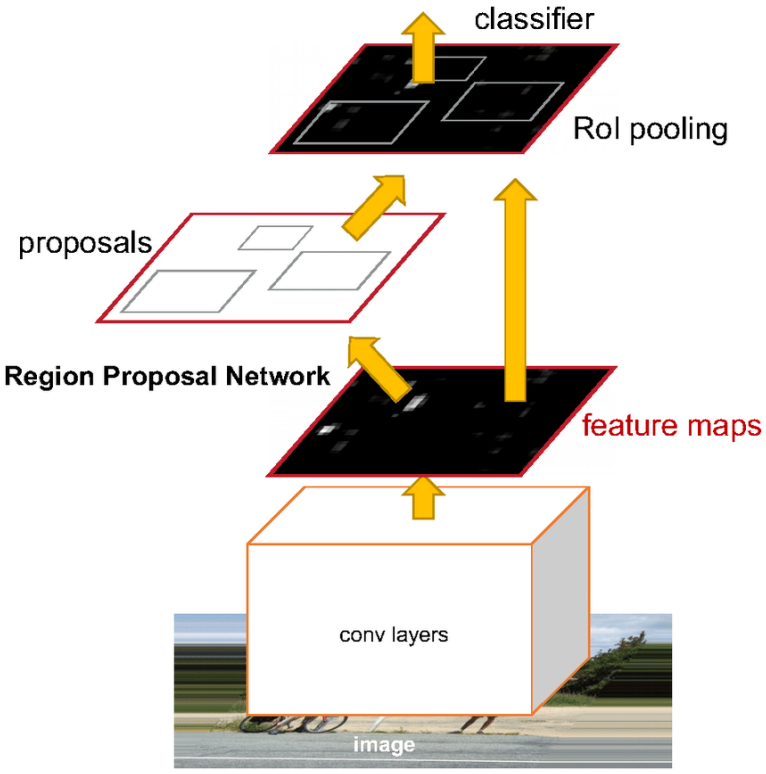


Figure 2.15: Faster RCNN model from Ren et al. [2015]

Anchors are generated from the latest feature maps of the *backbone* network which denotes a pre-trained CNN used as a feature extraction block. For example, the original paper used the VGG-16 model in their implementation. Multiple sizes and ratios can be preset for defining anchors, allowing different types of objects to be captured. By default, it is common to configure 9 anchor boxes per location on feature maps. All anchors are used to feed two FC layers, one for classification task, the other for coordinates regression. Each generated box is defined as positive (foreground class) or negative (background class) box depending on how much it overlaps with the ground-truth bounding boxes. The regressor is thus trained for giving boxes coordinates offset to adjust ground-truth boxes, and the classifier gives a confidence score for each box. Figure 2.14 depicts the workflow of the model on particular input pixel. Loss function for training a RPN is defined as follows:

$$\mathrm{L}\left(\{p_i, t_i\}\right) = \frac{1}{\mathrm{N}_{cls}} \sum_{i}^{\mathrm{N}_a} \mathrm{L}_{cls}(p_i, p_i^*) + \lambda \frac{1}{\mathrm{N}_{reg}} \sum_{i}^{\mathrm{N}_a} p_i^* \mathrm{L}_{reg}(t_i, t_i^*) \ ,$$

where:

- i is the anchor index inside the mini-batch, N_a is the total number of anchors.
- p_i is the predicted probability of anchors contains an object.
- p_i^* is the ground truth "probability" value of anchors. The value is 1 for positive anchors, 0 otherwise.
- t_i is the coordinates vector of predicted boxes.
- t_i^* is ground truth coordinate associated with bounding boxes.
- L_{cls} is a classifier loss (binary log loss over two classes: object vs. non object).
- $\mathrm{L}_{reg} = \mathrm{R}(t_i - t_i^*)$ is a regression loss, where R is a smooth ℓ_1 loss used in Girshick [2015].
- N_{cls} and N_{reg} are normalization parameters of mini-batch size. (In the original paper, $\mathrm{N}_{cls} = 256$ and $\mathrm{N}_{reg} \approx 2400$).
- λ is used in order to make loss function equally weighted for the classifier and the regressor ($\lambda = 10$ in the original paper).

To deploy the model on segmentation problems, He et al. [2017] have extended Faster R-CNN with Mask R-CNN model. Basically, Mask R-CNN presents high similarities with Faster RCNN and simply replace the ROI pooling layer with a more sophisticated ROI Align layer. The main interest of this operation is to avoid loss of information induced by rounding errors due to a floating point division. Bilinear interpolation is then applied to obtain the exact value of each feature at each position. After passing through a ROI align layer, all ROI are converted into feature maps of the same shape. For Mask R-CNN, these feature maps are used to predict a class and bounding box for each R-CNN (like Fast R-CNN), but the novelty is that a new branch consisting of a small FCN is added to determine the pixel-level location of each object.

2.4.6 You only look once

By virtue of their step of proposals regions, all methods examined so far correspond to a set of so-called two-stage detectors. This designation is in contrast to the You Only Look Once (YOLO) model which embodies the family of single stage detectors. Although they reach high accuracy, the main bottleneck of two-stage models is their execution time,

which prevent real-time use in certain practical cases, such as autonomous driving. To counterbalance this issue, Redmon et al. [2016] implemented the YOLO model whom the essential idea is depicted in Figure 2.16. Instead of going through a region classification step followed by refinement stage like Faster R-CNN/Mask R-CNN, this algorithm outputs class probabilities and bounding box coordinates from an entire image in one pass. The model divides the input image into S × S cells of equal size, and each cell is responsible for predicting the object centered in it with B anchors boxes. Each prediction yields 5 components: coordinates (b_x, b_y, b_w, b_h) and confidence value s. The bounding box center (b_x, b_y) is given relative to the bounds of the grid cell while the width and height (b_w, b_h) are predicted relative to the whole image. The confidence score is explicited by the authors as $\mathrm{Pr}(Object) * \mathrm{IoU}_{pred}^{truth}$ where $\mathrm{Pr}(Object)$ denotes to how likely the box contains an object, and $\mathrm{IoU}_{pred}^{truth}$ reflects how accurate the localization is [1]. For each grid cell, C conditional class probabilities $\mathrm{Pr}(Class_i \mid Object)$ are computed. At test time, the conditional class probabilities and the individual box confidence predictions are multiplied and give a class-specific confidence score for each box:

$$\mathrm{Pr}(Class_i|Object) * \mathrm{Pr}(Object) * \mathrm{IoU}_{pred}^{truth} = \mathrm{Pr}(Class_i) * \mathrm{IoU}_{pred}^{truth}$$

These scores reflect both the likelihood of that class appearing in the box and how well the box fits the object. The model only gives one set of class probabilities per cell, and the output tensor is of size S × S × (B × 5 + C).

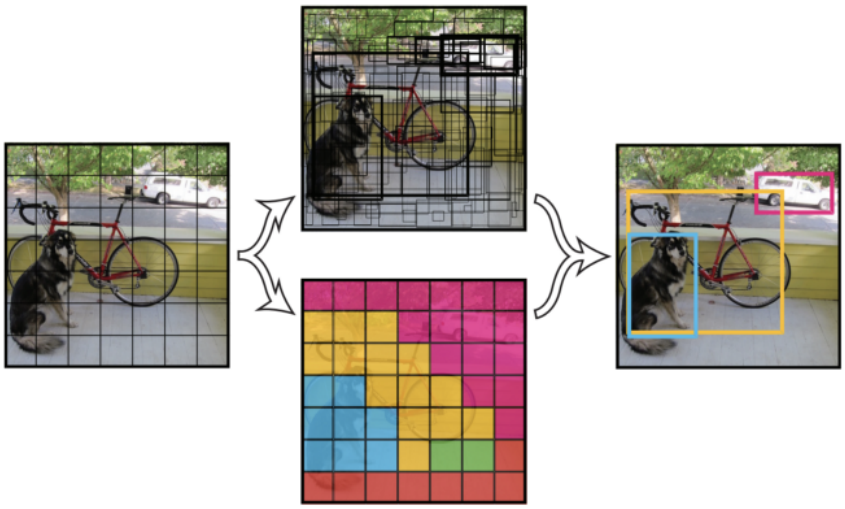


Figure 2.16: YOLO workflow from Redmon et al. [2016]. Input image is divided into S × S cells, each producing B bounding boxes, confidence scores, and C class probabilities.

YOLO is known to be particularly fast during runtime and suitable for real-time use. Its unified architecture allows capturing contextual information and avoid false positives. However, YOLO consists of several downsampling operations that lead to relatively coarse features and generalization problems for different object sizes. Furthermore, since only one object per cell is predicted, the main limitation is the performance of the detection of small objects near each other in the environment.

[1] (This score refers to the Intersection over Union metrics between prediction and ground-truth, see Section 3.3.1 for more details.)

2.4.7 Single shot detector

The SSD model proposed by Liu et al. [2016] finds a better trade off between speed and accuracy. Just like YOLO, this model belongs to the one-stage detector family and performs object detection in one pass. Feature extraction is originally performed by a VGG-16 network, which is truncated from its FC layers. This block is then augmented by successive convolutional layers that allow a gradual reduction in the resolution of the feature maps (represented by Conv8_2, Conv9_2, Conv10_2, and Conv11_2 in Figure 2.17). Intuitively, these supplementary layers can be viewed as a representation of the input image with multiple scaling that enables the detection of objects of different sizes.

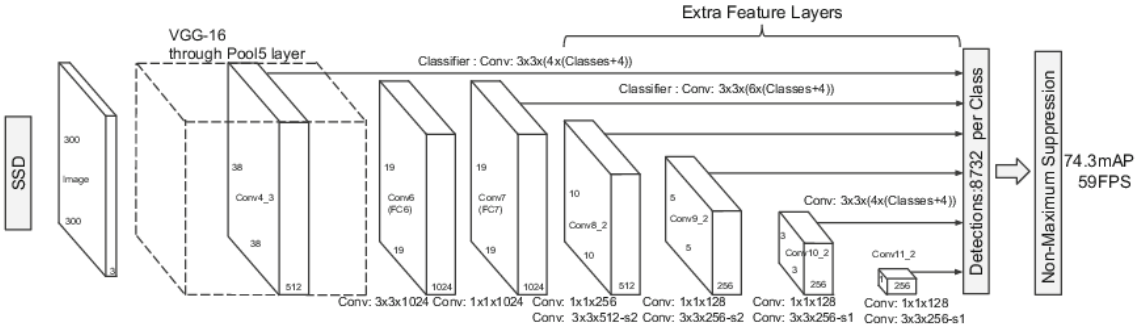


Figure 2.17: Architecture of Single Shot Detector (SSD) model. SSD uses VGG-16 as a feature extractor and extends it with additional layers of downsampling convolutions. Each new layer is used for bounding box prediction and allows detection at different scales. Image source: Liu et al. [2016].

The workflow of SSD has high similarities to those of YOLO and faster R-CNN. The algorithm divides feature maps to grid cells where *default boxes* are defined. These are equivalent to RPN's anchors boxes and instead of predicting boxes immediately, SSD predicts coordinates offsets. Each default boxes are predefined with several scales and aspect ratios values. One scale size is assigned to each layer and all scales are equally spaced between the layers. Assuming m feature maps are used for prediction, the default boxes scale for each feature map $k \in [1, m]$ is determined by:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1)$$

where $s_{min} = 0.2$ and $s_{max} = 0.9$ in the original paper. The aspect ratios are configured as $a_r \in \{1, 2, 3, 1/2, 1/3\}$ and allow to compute the width ($w_k^a = s_k\sqrt{a_r}$) and height ($h_k^a = s_k/\sqrt{a_r}$) for each default box. For the aspect ratio of 1, the authors also append a default box with a scale of $s'_k = \sqrt{s_k s_{k+1}}$, resulting in 6 default boxes per feature map location. Therefore, all predictions are associated with each default box coming from lower to higher features maps, allowing detection power to be maintained regardless of the size or shape of objects of interest.

2.4.8 Retina Net

Despite an elegant multiscale detection framework, the evaluation of the SSD model on COCO [Lin et al., 2014] or PASCAL Visual Object Classes (PASCAL-VOC) [Everingham et al., 2010] benchmarks tends to show lower accuracy for smaller objects compared to medium/large object [Liu et al., 2016]. This can be attributed to the fact that the predictions for small objects come from early feature maps, which provide a weaker representation of the input image. Lin et al. [2017b] addresses this issue by designing a one-stage

detector named RetinaNet. Its architecture consists of multiple connected blocks: a FPN for computing feature maps, and a classification subnet and a box regression subnet for outputting bounding boxes.

Feature Pyramid Network (FPN)

Introduced by Lin et al. [2017a], FPN's architecture shows similarities with SSD and take a multiscale approach for object detection. While most CNN consecutively produce decreasing height and width feature maps only, FPN is designed for generating growing feature maps dimensions as well. This allows to build a pyramid-shaped network combining low and high resolution maps through a top-down pathway and lateral connections (see Figure 2.18).

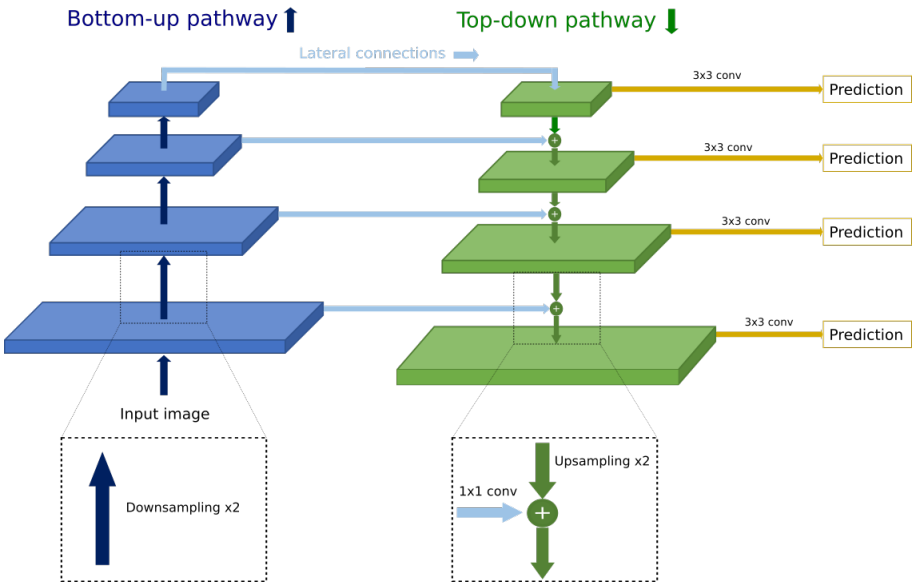


Figure 2.18: Architecture of Feature Pyramid Network (FPN). FPN comprises a bottom-up and a top-down pathway. The bottom-up pathway refers to the feedforward operation of the backbone. The top-down pathway apply upsampling operation on feature maps of each pyramid stage and merge it with lateral connections.

The architecture of FPN initiates a bottom-up pathway corresponding to the feedforward operation of the backbone network. This procedure results in a feature hierarchy where each level describes a declining scaling of feature maps. Once the highest stage is reached, the transition to the top-down pathway begins. Each new layer is generated from an upsampling process of the previous level, to which the lower layer of same size is merged via lateral connections.

More precisely, the top-down pathway goes as follows:

1. First, a nearest neighbor algorithm is used to upsample the highest feature map, resulting in maps twice larger. The authors assumed that another upsampling method could be applied for improving performances.
2. Feature maps from the backbone network undergoes a 1×1 convolution layer allowing to reduce the channel dimension and to combine high-resolution features with low-resolution features
3. Finally, both feature maps are merged by element-wise addition.

4. The process continues until reaching the largest feature maps, and predictions are generated by applying 3×3 convolution layer on every merged maps.

Classification and regression subnetworks

Two fully convolutional subnetworks supplement each pyramid level to perform object detection. The first subnetwork is a classification network which predicts the presence of each object class. Its architecture comprises four 3×3 convolutions layers, each with 256 channels and ReLU activations. Then, an additional 3×3 convolution layer is applied followed by a sigmoid activation. This results in an output grid with the same height H and width W as the input feature map, and with depth $(C \times k)$ where C denotes the number of class object and k the total amount of anchor boxes.

In parallel to the classification subnetwork, a second subnetwork with an almost identical design operates a regression task to adjust anchor box coordinates. The key difference lies in the last 3×3 convolutional layer which produces a feature map of shape $(H, W, 4 \times k)$ where k still refers to the number of anchor boxes. This output compiles all predicted offsets relative to the corresponding anchor boxes dimensions.

The final output is obtained by transforming detection proposals coming from each pyramid level into image coordinates. Detections are then refined with a Non-Maximum Suppression (NMS) algorithm which essentially involve to select the prediction with the highest confidence score among a group of overlapping boxes (see Section 3.3.2 for details).

2.5 Conclusions

This chapter provides a detailed overview of the basic principles required to process and use Deep Learning models for specific application purposes. Starting with a simple modeling of the biological neuron, we have presented all the basic operations required to train an algorithm to perform a particular task. As far as computer vision is concerned, it seems undeniable that the 2010s marked an impressive rise in the popularity of CNNs models in many application areas. This popularity is largely due to convolutional layers which are the main building block of these models. In conjunction with pooling, nonlinearity, or normalization operations, the superposition of convolutional layers enables the creation of an increasingly complex representation of features in the input image. During training, the backpropagation algorithm allows the adaptation of these representations so that the most relevant ones are automatically found to understand the given scene. This last property is clearly the main advantage of CNNs over more traditional methods, which often require a fairly good expertise for the feature extraction phase.

To achieve a better understanding of the presented scenes, the architecture of CNNs models has been further developed to perform segmentation or object recognition tasks. Up-sampling operations are particularly useful for transitioning from a lower-resolution to a high-resolution feature map. Similarly, skip connections are a good way to link contextual and spatial information. These components are key elements of the U-net model that forms the basis of many segmentation models for medical imaging [Kayalibay et al., 2017; Milletari et al., 2016; Zhou et al., 2018, 2019]. Therefore, in Chapter 4 and Chapter 5, we will use models whose architecture is derived from the architecture of U-net models for segmentation of cell nuclei in microscopic imaging.

As for the problems of object detection, we have given an overview of the main models that have made an important contribution to the field of computer vision. Two main fam-

ilies of models have been identified: two-stage and single-stage detectors. As the name suggests, the former divide the task into two subtasks (region proposal and subsequent region classification), while the latter perform the same task without the region proposal procedure. This difference generally results in higher accuracy for two-stage models, but this comes at a higher computational time cost than the single-stage detection models. In Chapter 3, we review this trend in the context of 3D recognition in medical imaging and compare different model implementations.

Chapter 3

Pulmonary nodule detection on 3D CT scans.

Abstract

In this chapter, we present different methods for object detection based on Deep Learning and show how they can be used in a medical context. After a brief presentation of lung cancer and its epidemiology, we expose the need to develop systems that can perform the same diagnosis as radiologists in Section 3.1. Section 3.2 summarizes all implementations used to train a DL model for pulmonary nodule detection. Then, Section 3.3 introduces the most common metrics used to measure models detection performance. In Section 3.4, we propose a comparative analysis of different models performances for detecting pulmonary nodules, as well as a method for reducing false positives by segmenting the lungs. Section 3.5, we extend this workflow in a data challenge context by building a decision support system that provides an estimate of the malignancy of lung nodules. Finally, Section 3.6 provides an assessment of the results obtained in each context and identifies some areas where diagnostic support could be improved.

Contents

3.1 Medical context 32
3.1.1 Motivations 32
3.2 Training strategy for nodule detection 33
3.2.1 Datasets 34
3.2.2 Preprocessing 35
3.2.3 Compared models and implementation details 36
3.2.4 Loss functions 38
3.2.5 Batch balancing 40
3.2.6 Data augmentation 41
3.3 Evaluating nodule detectors 43
3.3.1 Intersection over union 43
3.3.2 Filtering detections 44
3.3.3 Labeling of detections 44
3.3.4 Precision-Recall curve 45
3.3.5 Free-Response ROC curve 48
3.4 Experiments and results 50
3.4.1 Comparative analysis: 2D vs 3D models 50
3.4.2 Filtering false positive detection 52
3.4.3 Influence of hyperparameters on nodule detection 55

3.5 Building a Computer-Aided System for patient diagnosis 55
3.5.1 Data challenge context 55
3.5.2 A three-stage pipeline for patient diagnosis 56
3.6 Conclusions and perspectives 57

3.1 Medical context

3.1.1 Motivations

The World Health Organization (WHO) estimates that more than 1.7 million people died from lung cancer in 2018, making it one of the deadliest cancers worldwide (see Figure 3.1). These deaths can be partially explained by the fact that symptoms do not appear until the disease is at an advanced stage. The key to reducing lung cancer mortality rates, therefore, is early and accurate diagnosis of small tissue collections in the lungs called *nodules* [Winer-Muram, 2006].

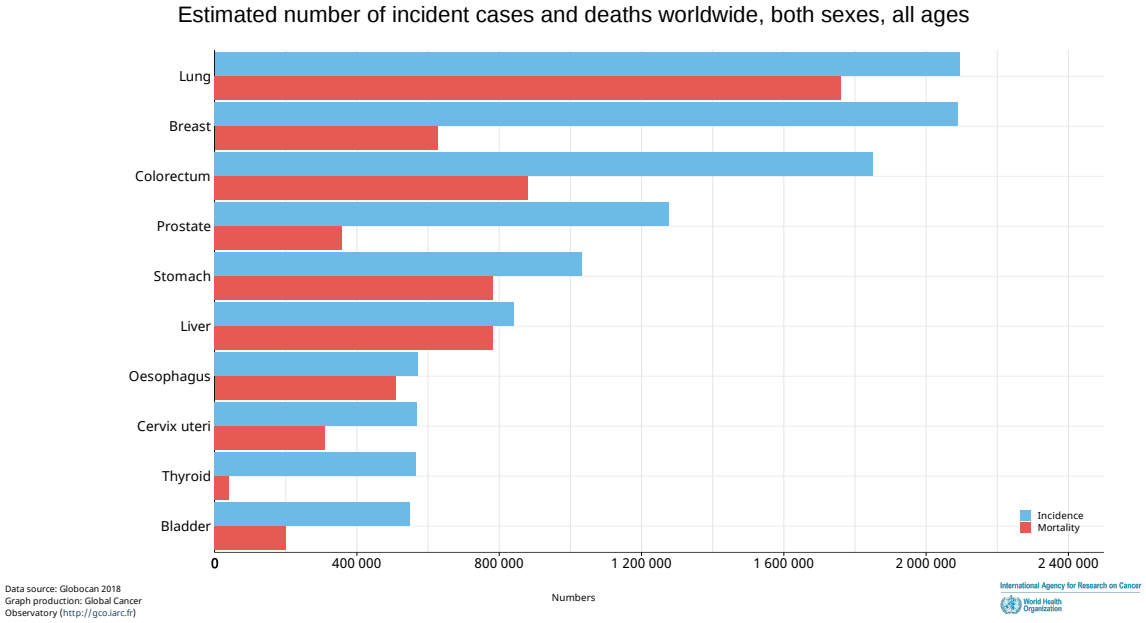


Figure 3.1: Estimated number of incidence cases and deaths in 2018 worldwide, all cancers, both sexes, all ages (adapted from International Agency for Research on Cancer 2019).

There are several medical definitions to describe a pulmonary nodule, but most tend to describe it as a localized lesion that is more or less round and may be up to 3 cm in diameter. A collection of representative nodules is shown in Figure 3.2. Patient care depends on early detection and diagnosis of the type of nodule (cancerous or benign). Nodule detection has been facilitated by the development of imaging techniques based on radiographic measurements that allow the interior of an object to be examined without cutting it. In particular, the CT protocol outlines early-stage cancer [Swensen et al., 2003] and is widely recommended in the patient diagnostic process. However, diagnosis remains a difficult analytic thinking even for the most experienced doctors because malignancy is not determined solely by the size or morphology of the nodule. The only way to make a true diagnosis is by biopsy or surgery, which carries additional risks for the patient. In this context, the challenge for computer vision is to support radiologists, who

are faced with a rather complex and time-consuming task on a daily basis. Computer-aided diagnosis (CADx) techniques therefore have a threefold goal: speeding up analyses, eliminating subjectivity in image interpretation, and reducing medical costs.

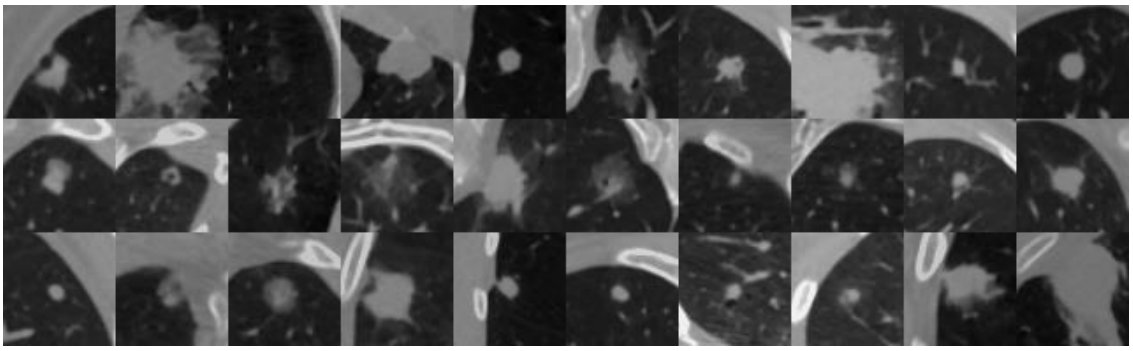


Figure 3.2: A collection of pulmonary nodule extracted from the LIDC dataset [Armato III et al., 2011]. Lung nodules are lesions of a more or less spherical shape whose presence can be the direct consequence of lung cancer.

The construction of an effective CADx system can rely on different approaches. Santos et al. [2014] developed a methodology based on classical image processing and ML techniques. The combination of texture features for detecting nodules regions associated with SVM classifiers to discriminate non-nodule from real nodule has shown satisfactory performance. Peña et al. [2016] implemented a four stage pipeline starting with lung detection and segmentation and performing connectivity algorithm to select nodules candidates. Camarlinghi et al. [2012] joined the output of three existing CADx systems and demonstrated the combination of different approaches offers better results than each single system.

The major drawback with these methods is the extracting process of features, which are totally handmade and depending on experts comprehensions, developers and/or parameters used during the image acquisition. On the contrary, DL methods automatically learn relevant features from the data, provided, thus enhancing the model's generalization. In particular, CNN-based methods are considered to improve on human experts for accurate diagnosis of early pulmonary nodules [Zhao et al., 2012]. Hence, more and more methods using DL have emerged to detect and classify nodules.

3.2 Training strategy for nodule detection

In this section, we describe all the procedures we use to train a nodule detection model. We mainly rely on a Python framework developed by Jaeger et al. [2018] called Medical Detection Toolkit (MDT)[1]. The latter offers implementations of powerful tools for training and testing state-of-the-art detection models on any customized datasets.

[1] https://github.com/MIC-DKFZ/medicaldetectiontoolkit

3.2.1 Datasets

Lung Image Database Consortium

Datasets play a key role in the development of efficient DL-based frameworks. Whether for classification, segmentation, or object recognition problems, high quality labelled data is key for training. In the medical field, the labelling procedure requires at least one expert, and in particularly ambiguous cases, possibly several. This process is very costly, both in terms of human and material resources, and is the main barrier to developing robust models to improve health outcomes in a hospital setup. To encourage the development of CADx solution for nodule detection, academic institutions and industrial companies collaborated to develop a public database, called LIDC [Armato III et al., 2011]. This dataset includes a total of 1018 cases from 1010 patients, consisting of 244,527 Digital Imaging and Communications in Medicine (DICOM) files and corresponding XML files with experts annotations. Each DICOM file corresponds to a two-dimensional (2D) slice image of a patient's scan. The combination of all slices from the same patient results in a 3D volume and represents the entire CT scan. Figure 3.3 shows examples montage of annotations superposed on CT scan. For data labeling, four radiologists have participated in a two-stage process. The first step was a blinded-reading phase involving to review independently all CT scans, then marking and categorizing all discoveries. After completed this phase, all results were compiled and sent back to the same radiologists to begin the unblinded phase. In this way, each expert reviewed his own evaluation and those of his colleagues and assigns a final mark category. This procedure enables to capture as many candidate nodules as possible without forcing a consensus among the experts. LIDC organizers decided to define three different mark categories, based on the nature and the size object. To build efficient CADx system, both nodules and non-nodules were annotated and categorized.

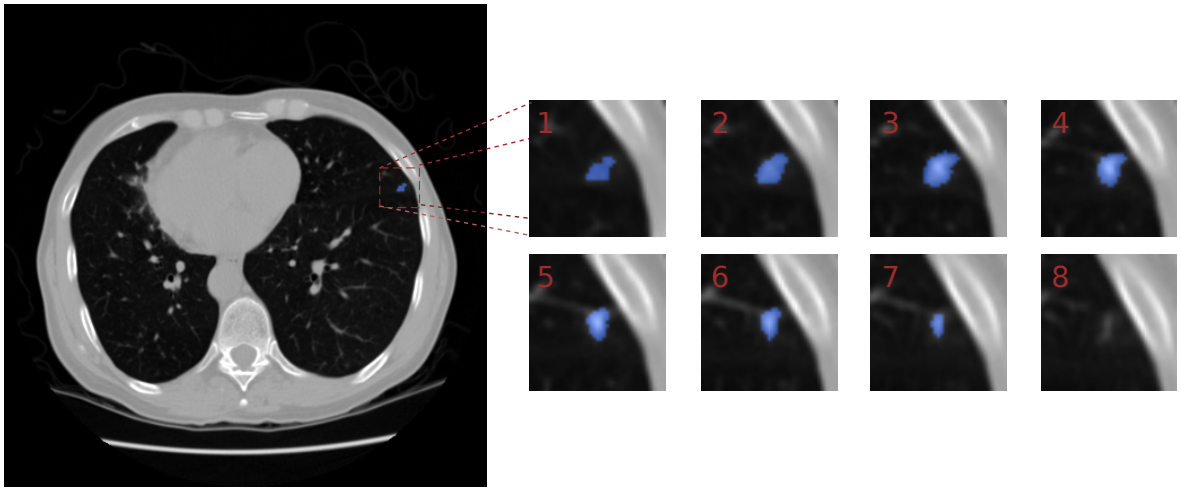


Figure 3.3: Left: a slice of 3D CT scan taken from LIDC dataset. Right: 3D annotations by several radiologists depicting a nodule (blue). From 1 to 8, the scan slices are cropped and sorted in order of depth in the lung.

Nodule with a diameter $\geq 3mm$ **but** $\leq 30mm$**.** This class was manually segmented and contours of objects have been defined in 3D. In addition, each expert has given a complete characterization by giving a 1-5 rating on different criteria :

- *Subtlety*: which refers to the difficulty in detection

- *Internal structure*: or expected internal composition of the nodule (soft tissue, fluid, fat, air)
- *Calcification*: pattern of calcification if present
- *Sphericity*: the three-dimensional shape of the nodule in terms of its roundness
- *Margin*: description of how well-defined the margins of the nodule is
- *Speculation*: amount of speculation (or spikes) present in nodule
- *Texture*: internal texture or composition of nodule in terms of solid and ground glass components
- *Malignancy*: reflecting how likely the annotation corresponds to a malignant nodule, (based on the assumption of a 60-year-old male smoker)

Nodule with a diameter $< 3mm$**.** This class was labelled with only an approximate three-dimensional center of mass, with no other characteristics. If the opacity clearly indicated a benign nodule, no marking was done.

Non-nodule with a diameter $\geq 3mm$**.** These markers were used to indicate anomalies in the scan but are not considered nodules. Explicitly indicating objects that do not represent potential cancer is very useful for developing an appropriate algorithm. Nevertheless, non-nodule objects with a diameter $< 3mm$ were simply ignored.

ANODE09 and LUNA16 challenges

To provide an objective comparison between CADx systems, several competitions were conducted using a common database and common evaluation criteria. Automatic Nodule Detection 2009 (ANODE09) Van Ginneken et al. [2010] was the first competition organized in this direction. However, the main limitation was the lack of diversity of available scanners: only about fifty scanners were made available to which the same acquisition protocol was applied.

Lung Nodule Analysis 2016 (LUNA16) focuses on a large-scale evaluation of nodule detection on the LIDC dataset. As the data description on the website states[2] CT scans with inconsistent slice thickness or with slice thickness greater than 2.5 mm were excluded, resulting in a final list of 888 scans merged into MetaImage (.mhd) files. In addition, the organizers decided to take as reference all nodules greater than 3 mm in diameter that were annotated by at least three radiologists. The other markers (nodules $\leq 3mm$, non-nodules $\geq 3mm$, nodules $\geq 3mm$ that were marked by only 1 or 2 radiologists) are simply ignored in the analysis. Therefore, all detections corresponding to these annotations are considered neither true positives nor false positives, since they represent abnormalities that may be important in the diagnosis of other pathologies.

3.2.2 Preprocessing

The nodule detection task is performed on LIDC dataset consisting of 1018 CT scans acquired from different hospitals and thus from different scanners and protocols. To manage the large variety of data, a unification process must be applied to all patients. Götz [2018] provided a first framework operating a preliminary data conversion step[3]. CT scan are converted from a whole DICOM (.dcm) Series to nearly raw raster data (.nrrd) files to

[2] https://luna16.grand-challenge.org/Data/

[3] See: https://github.com/MIC-DKFZ/LIDC-IDRI-processing/tree/v1.0.1

be read directly as 3D stacks. The contours of the nodules defined by the experts are extracted from XML files to define a nodule segmentation mask in Nifti format (.nii.gz). The features relating to the size of the nodule, its position or its malignancy is also extracted from XML files and converted to CSV format[4].

Material	HU
Air	-1000
Lung	-500
Fat	[-100,-50]
Water	0
Blood	30-45
Muscles	10-40
Bones	700
Foreign bodies	> 1000

Table 3.1: Hounsefield Unit (HU) values for some components located in or around the lungs. Data extracted from HU scale wikipedia page.

CT scanning uses X-rays to create lung slices whose intensity values correspond to an attenuation coefficient given in Hounsfield Unit (HU). As suggested in Table 3.1, HU values are directly associated with some fluids or body parts. To remove some unnecessary information, voxels intensities of CT were clipped between -1200 and 600 HU before applying a standardization process. To facilitate the detection of each nodule, the next step is to resampled the dataset (*i.e.* scans and masks) to a fixed pixel resolution. For each patient, the dimensions of the pixels (height and width) and the distance between each slice (depth) are stored in the meta-information that is written in each DICOM file. To account for the heterogeneity of the data, all scans were resampled to a resolution of $0.7 \times 0.7 \times 1.25 mm^3$, which is approximately the average resolution in the whole dataset.

The nodule features are also extracted from the CSV file to summarize the annotations of all the raters. In this way, a consensus contour for each lesion is determined by pixel-wise majority voting. In addition, averaging the malignancy score assigned by each expert allows the determination of the class of each region (benign or malignant). The result of this method is a segmentation of all nodules in the scan. Once pre-processing is complete, the images and corresponding masks are saved as numpy arrays (.npy) to dynamically load the data during training, and meta-information about the location and severity of the lesions is stored in a pickle file (.pck). With this configuration, the dataset occupies a total of more than 300 GB of storage space.

3.2.3 Compared models and implementation details

Nodule localization can be understood from two points of view: either it can be considered as a segmentation problem or as an object detection problem, and in both cases different state-of-art architectures are used in the following sections. Essentially, we reproduced the nodule detection experience described by Jaeger et al. [2018] using their MDT. All models described in this section are derived from this framework and are available both in a 2D and 3D implementation using *Pytorch* 0.4.1 library[5].

[4]This script was only tested with Windows. The authors assume that the script will also run on Linux, but no guarantee is given.

[5]A branch update with Pytorch >= 1.4.0 is available since April 2020, but not used here.

Backbone model

To ensure fairness between models, the feature extraction phase is performed with the same backbone FPN based on ResNet-50 [He et al., 2016]. The original ResNet-50 model consists of 5 blocks whose corresponding outputs are denoted by $\{C_i\}_{i\in[1,5]}$. The first block is a convolutional layer 7×7 with 64 filters and a stride of 2, followed by a layer of Max Pooling 3×3 with also a stride of 2. The following blocks of the model are organized in a series with a fixed length of residual blocks consisting of 3 convolutional layers (1×1, 3×3, then again 1×1). At each stage, the spatial resolution is divided by two compared to the input size, while the depth has doubled. Then, the top-down pathway rebuilt each level of the pyramid $\{P_i\}_{i\in[0,5]}$ by successively applying upsampling operations and using blocks C_i via lateral connections. Figure 3.4 provides a helpful representation of a model incorporating this architecture. Six levels are shown in the pyramid denoted as $\{P_0, P_1, P_2, P_3, P_4, P_5\}$, where P_i has resolution 2^i lower than the input image.

U-FPN

Following the U-net of Ronneberger et al. [2015], a model called U-FPN is used as a baseline for segmentation task. The difference lies in the lateral connections, which apply a 1×1 convolution on each block C_i before being merged by element-wise addition with pyramid level P_{i+1}. Moreover, a 1×1 layer is added after P_0, followed by a softmax layer applied to the pixels to obtain a final segmentation of the input image. The confidence value assigned to each bounding box is defined by the maximum probability among all pixels in the box.

Mask R-CNN

The implementation of the Mask R-CNN adds a RPN branch at each pyramid level except P_1 and P_0. The number of RPN feature maps is 512 in 2D, while in 3D it is lowered to 128 because of GPU memory limitations. The xy size of the anchors is chosen according to the standard size of the objects in the dataset and set to $\{4^2, 8^2, 16^2, 32^2\}$ for pyramid levels from P_5 to P_2. For 3D model, experience shows that medical images resolution in the z axis is generally lower than in x or y, so the depth of anchors cubes are set to $\{1, 2, 4, 8\}$.

Retina Net and Retina U-net

The models Retina net and Retina U-net [Jaeger et al., 2018] belong to the family of single-stage detectors and replace the RPN by two subnetworks: a classifier and a regressor. It consists of four 3×3 convolutional layers with 256 kernels (64 in 3D) and a final 3×3 convolutional layer with $n_{class} \times n_{anchor}$ filters for the classifier and $n_{anchor} \times d \times 2$ filters for the regressors, where d denotes the dimensionality of the input image (2D or 3D). The Retina U-net model, as shown in Figure 3.4, adds a segmentation output by simply applying convolution and softmax layer to the last layer of the top-down pathway P_0.

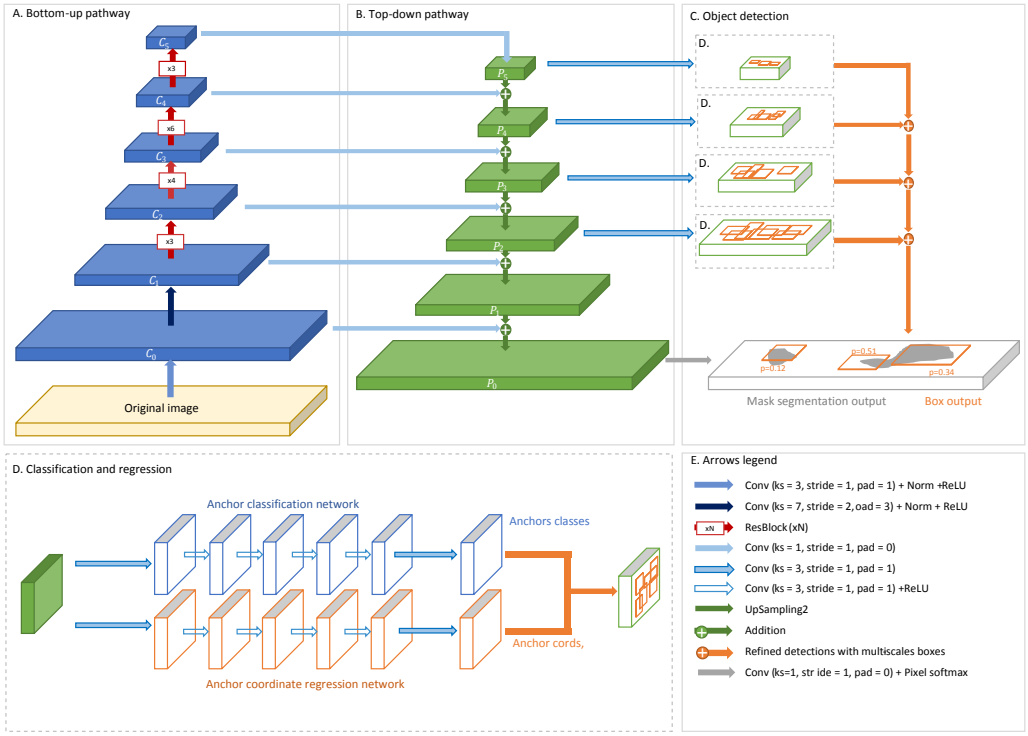


Figure 3.4: Architecture of Retina U-net proposed by Jaeger et al. [2018]. The model consists of a FPN with a Resnet-50 backbone that enables feature extraction at different scales. A ROI regressor and a classifier are connected to each pyramid level from P_2 to P_5 and provide bounding box output. A segmentation output is also given by applying 1×1 convolution and softmax layer on P_0.

3.2.4 Loss functions

Regardless of the given problem, training a model is fundamentally about optimizing a loss function in the context of the task being learned. This implies that the values of observed loss over epochs should indicate whether the predictions of the model are improving. As a consequence, the choice of a relevant loss function affects the robustness and efficiency of the final model. This section lists the loss functions used in our experiences.

Classification loss functions

Most classification models are constructed to provide an output for each class in the form of a value between 0 and 1, which is equated with a probability of belonging to the class. The most common and appropriate loss function in this configuration is the *cross-entropy* function, which is a measure of the distance between two distributions. In the case of a binary classification with N samples, all predicted probability s_i are compared to target class y_i (= 0 or 1) using the binary cross-entropy (BCE):

$$\mathscr{L}_{\text{BCE}} = -\frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \left(y_i \log(s_i) + (1 - y_i) \log(1 - s_i) \right) \quad . \tag{3.1}$$

Intuitively, this function strongly penalizes the large differences (close to 1) between probability and real class. For multi-class problem with $\text{C} > 2$ class, this function can be gen-

eralized to Categorical Cross Entropy (CCE):

$$\mathscr{L}_{\mathrm{CCE}} = -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{c=1}^{\mathrm{C}} y_{i,c} \log(s_{i,c}) \ , \tag{3.2}$$

where $y_{i,c}$ is a binary indicator encoding the class c of the i-th observation, and $s_{i,c}$ the corresponding probability.

Regression loss function

The regression task is concerned with the prediction of continuous quantities. The standard loss functions used in this context are generally the absolute error $\mathscr{L}_1 = |y - \hat{y}|$ or the quadratic error $\mathscr{L}_2 = \frac{1}{2}(y - \hat{y})^2$. Both have different mathematical properties that represent an advantage or disadvantage when compared to each other. For example, L_2 is differentiable everywhere, while L_1 is not differentiable at 0. However, L_1 is less sensitive to outliers than L_2. The Huber loss [Huber, 1992] is often presented as a compromise between L_1 and L_2 taking the behavior of one or the other from a fixed parameter $\delta > 0$:

$$\mathscr{L}_{reg} = \begin{cases} \frac{1}{2} \cdot \left(y - \hat{y}\right)^2 & \text{if } |y - \hat{y}| \leq \delta \\ \delta \left(|y - \hat{y}| - \frac{1}{2} \cdot \delta\right) & \text{otherwise.} \end{cases} \tag{3.3}$$

This function can be used for predicting bounding box coordinates and to measure the accuracy of the estimated bounding box position.

Segmentation loss functions

For segmentation tasks, (3.1) and (3.2) can easily be adapted pixel-wise. To counteract the large imbalance between the different pixel classes, it is also possible to extend the cross entropy to a weighted form called weighted cross entropy (WCE):

$$\mathscr{L}_{\mathrm{WCE}} = -\frac{1}{\mathrm{N}_{pix}} \sum_{i=1}^{\mathrm{N}_{pix}} \sum_{c=1}^{\mathrm{C}} \beta_c \log(p_{i,c}) \ , \tag{3.4}$$

with $p_{i,c}$ is a pixel element of the predicted probability map for class c, N_{pix} denotes the number of input pixels and β_c is a weight value related to class c. In practice, larger weight values are assigned to underrepresented class. Another popular loss function for evaluating the performance of a segmentation model is the Dice loss function [Milletari et al., 2016]. Ranging from 0 to 1, this function measures the overlapping between prediction and ground truth:

$$\mathscr{L}_{\mathrm{D}ice} = 1 - 2 \cdot \frac{\sum_{i}^{\mathrm{N}_{pix}} p_i g_i}{\sum_{i}^{\mathrm{N}_{pix}} p_i + \sum_{i}^{\mathrm{N}_{pix}} g_i + \epsilon} \ , \tag{3.5}$$

where p_i is a pixel of the predicted probability map with as many channels as classes, g_i a pixel of the one-hot ground-truth tensor, and $\epsilon > 0$ a small constant for numerical stability.

3.2.5 Batch balancing

The images from the LIDC dataset are particularly large because they are acquired as 3D volumes. This means that most computers, even those with the best GPU, have difficulty using a full scan as a single entity that forms a batch during the training phase. The most common solution to this problem is to train a model from subregions of fixed size in each image, called *tiles* or *patches*.

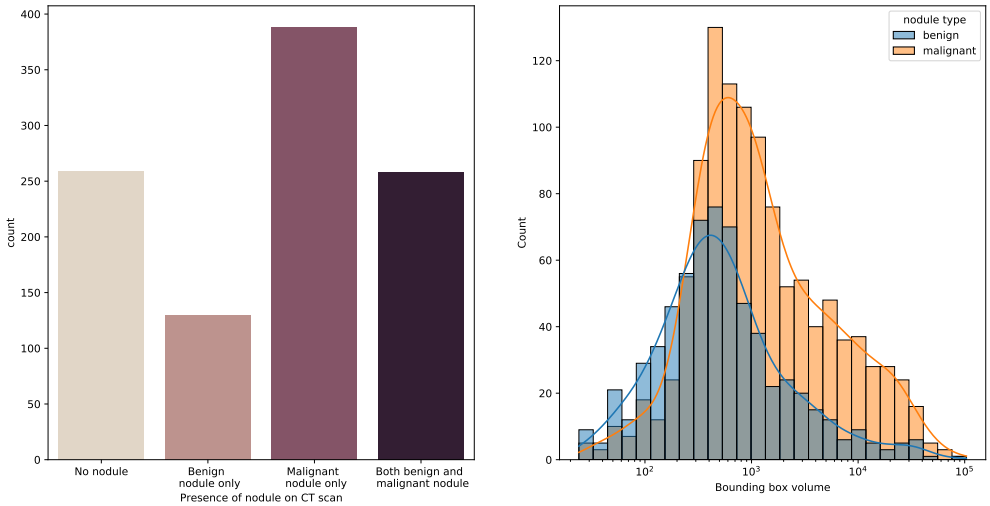


Figure 3.5: Description of the LIDC dataset after the preprocessing stage described in section 3.2.2. Left: countplot depending on the presence of nodules in each CT scan. Right: histogram of volumes (in voxels) according to nodule type.

Figure 3.5 gives an idea of the substantial imbalance of classes in the LIDC dataset. At patient level, scans without nodules make up only about 25 % of the total dataset. At annotation level, the average dimensions of bounding box are about $10 \times 10 \times 10$. This means that the proportion of voxels belonging to the nodule class is drastically smaller than the proportion of voxels belonging to "non-nodule" regions. With such a class imbalance within the training dataset, ML models tend to misclassify instances belonging to minority groups. To overcome this problem, the extent of imbalance is reduced by a batch balancing method at training.

The batch balancing method implemented in MDT is performed at two levels. First, a selection of patients is made with respect to the type of nodule present in each scan. Second, another selection is performed to determine the center of the tile to be extracted from the full scan.

Patient picking

When creating a training batch, it is necessary to pick a list of patients in advance to increase the chances of obtaining as many tiles with benign and malignant nodules as possible. The procedure is described by Algorithm 1 and consists of randomly selecting each patient and checking that the nodules present in the current scan do not increase the class imbalance in the batch. Thus, for each selected patient, each type of nodule is counted to determine the patient's *weak class*, denoted c_{weak}. If the weak class also matches the weak class of the batch to be created, the scan is rejected, otherwise it is accepted. In addition, a slack factor $s_f \in]0,1]$ is introduced corresponding to the fraction of patients

selected without any filtering. For the problem of nodule detection, the default value is set to 0.2, but this value can be increased for extremely imbalanced datasets.

Algorithm 1: Patient picking

Input : M: batch size, P^{list}: list of patients in the training set, $s_f \in]0,1]$: slack factor

Initialize $P^{list}_{batch} = \{\}$

for $i = 1, \ldots, M$ **do**

  Keep looking $\leftarrow$ True

  while *Keep looking* **do**

    $k \leftarrow$ Pick a random index uniformly among P^{list}

    $Counts \leftarrow$ List of number of nodules for each class

    $c_{weak,tmp} \leftarrow \text{argmin}(Counts)$ `// Check the least occurring nodule class for this patient.`

    if $i \leq |M \times s|$ **then**

      Keep looking $\leftarrow$ False

    else

      $Condition1 \leftarrow c_{weak,tmp} \neq c_{weak}$ `// Check if the weakest class in this patient is not the weakest in current batch.`

      $Condition2 \leftarrow Counts(c_{weak}) > 0$ `// Check if at least one region of this patient belongs to the weakest class.`

      if *Condition 1 & Condition 2* **then**

        Keep looking $\leftarrow$ False

      else

        Keep looking $\leftarrow$ True

  Add patient k to P^{list}_{batch}, and update the weakest class c_{weak}.

Output : P^{list}_{batch}

Patch picking

Most LIDC scans have xy dimensions of size 512×512, while their depth (z dimension) ranges from 200 to 800 slices. Since the size of nodules is particularly small compared to the dimensions of the individual scans, it is likely that a purely random extraction of tiles will result in numerous tiles without objects. To maintain the equilibrium previously established by Algorithm 1, MDT also includes a patch extraction procedure around randomly selected nodules. This extraction mechanism is triggered according to a Bernoulli process parameterized by a foreground probability p_{fg}, set to 0.5 by default. When the outcome of the Bernoulli process is "failure", the center pixel of the tile is randomly selected. In either case, extracting a tile of dimensions (t_H, t_W, t_D) is constrained by the dimensions of the full scan of dimensions (H, W, D). This means that the center of each tile is at a certain distance from the edges of the scan. The procedure for tile extraction is described by Algorithm 2.

3.2.6 Data augmentation

Data augmentation is a common method for training models. This method not only saves the cost of obtaining and labeling new data, but also has the main effect of avoiding overfitting to the training data and increasing the robustness of the models. The goal is to in-

Algorithm 2: Patch picking

Input : p_{fg} : foreground probability, (t_H, t_W, t_D): the tile size, a patient list P^{list}_{batch} of length M

for $i = 1, \ldots, M$ **do**

  Scan $\leftarrow$ i-th element of patient list

  H, W, D $\leftarrow$ Scan dimensions

  Pick $x \sim Bern(p_{fg})$

  if $x > 0$ & *Patient has at least one nodule* **then**

    Pick a random nodule among all nodules in current scan.

    Pick a random pixel coordinate (r, c, d) among all pixels belonging to the picked nodule.

    `// Looping over dimensions to pick patch center coordinates`

    for $dim_{tile} = t_H, t_W, t_D$; dim_{scan} = H, W, D ; $p_{coord} = r, c, d$ **do**

      Low $\leftarrow max\left(\frac{dim_{tile}}{2}, p_{coord} - \frac{\dim_{tile}}{2}\right)$

      High $\leftarrow min\left(dim_{scan} - \frac{\dim_{tile}}{2}, p_{coord} + \frac{\dim_{tile}}{2}\right)$

      Pick a random coordinate between Low and High.

  else

    Pick any coordinates away from the image border by at least $\frac{\dim_{tile}}{2}$ for each dimension.

  Extract a patch centered around the resulting coordinates and append it to batch.

Output : A batch with M elements consisting of tiles of size (t_H, t_W, t_D)

crease the diversity of the training dataset by applying different geometric or photometric transformations to each image fed to the model.

When sufficient storage space is available (say for relatively small datasets) *offline augmentation* is most common, and consists of enlarging the dataset by applying most combinations of transformations, and storing each resulting image. It has the advantage of controlling the augmentation factor of the dataset and making it easier for developers to verify the quality of the augmentation. In this work, we have investigated an alternative solution by augmenting the data using an *online* approach. We perform data transformations when visiting each training batch. Thus, at each iteration, the current batch is randomly subjected to a combination of the following transformations:

- **Translation:** it consists in shifting the image by several pixels in one or more dimensions. For example, the transformation allows a tile not always to be centered on a nodule. The displacement parameters are bound to the tile size.
- **Rotation:** a random angle between 0 and 2π is picked, only on the X axis in 2D, only on the Z axis in 3D. If the labeled regions in the training mask are not preserved after rotation, the transformation is cancelled.
- **Flipping:** it is a quite easy transformation consisting in reversing the order of pixels/voxels in one or more dimensions.
- **Random scaling:** it randomly rescales the inputs (ratio in $[0.8, 1.1]$ in our experiences). This operation is followed by a centered or random crop.
- **Random cropping:** it randomly selects any section of the original tile and then resizes it to the original tile size.
- **Elastic deformation:** it corresponds to applying a force that deforms the shape of the objects in the image. Firstly, a grid of points assigns a random displacement to each

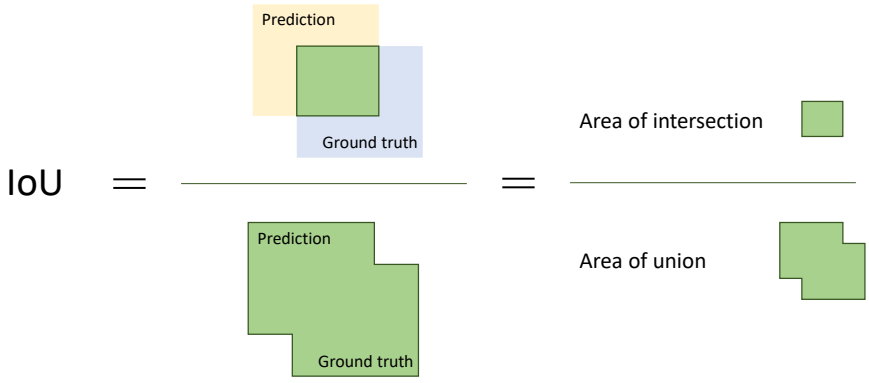


Figure 3.6: Intersection over Union (IoU) equation. The IoU metric reflects how close a prediction provided by an object detector is to the target object (the closer to 1, the better the match).

control point. Then a displacement of each pixel in the input image is calculated by interpolation with this grid.

3.3 Evaluating nodule detectors

Object detection in computer vision has experienced a significant rise of popularity in recent years due to its wide range of applications. In particular, there is a certain recurrence of competitions rewarding the best performing models on specific datasets. Depending on the edition of the competitions, the metrics used to establish a ranking of the models are dependent on the specificity of the underlying datasets. In this section, we focus on metrics used on the most popular competitions for object detection: PASCAL-VOC challenges [6], and COCO detection challenges [7]. We also examine the metric used in the LUNA16 challenge, on nodule detection.

3.3.1 Intersection over union

Object detection is a quite complex task implying to locate accurately and classify multiple targets on the same image(s). The evaluation of a model detector requires:

1. A set of ground-truth bounding boxes B_{gt} comprising all rectangular regions of all object classes to be detected in the dataset.
2. A set of predicted bounding boxes B_{pred} given by a model, each complemented by a class and a confidence score value.

To evaluate the quality of object detection models, it is common to quantify the proximity between detections and target objects. This means evaluating the proportion of overlap between predicted and ground-truth bounding boxes for each object class. Following Jaccard's index [Jaccard, 1901], this measure is denoted IoU and provides a similarity value between two boxes. Let b_{gt} be a ground truth bounding box and b_{pred} be any bounding box output from a model, then IoU is defined as the ratio of the area of their

[6] http://host.robots.ox.ac.uk/pascal/VOC/
[7] https://cocodataset.org

intersection to the area of their union:

$$\text{IoU}\left(b_{gt}, b_{pred}\right) = \frac{area\left(b_{gt} \cap b_{pred}\right)}{area\left(b_{gt} \cup b_{pred}\right)} \ .$$

As it is suggested by Figure 3.6, the best possible detection would perfectly match with the reference object box and give an IoU score of 1. On the contrary, $\text{IoU}\left(b_{gt}, b_{pred}\right) = 0$ corresponds to a situation where both boxes do not overlap each other.

3.3.2 Filtering detections

Since the image size often exceeds the memory capacity of the GPU, the prediction procedure is performed at the tile level. To increase the model accuracy, several strategies can be combined during inference: predicting on high overlapped tiles, using multiple models, apply test-time augmentation, ... These processes can result in numerous detections that overlap heavily, forming clusters of detections on the image that must be pruned to remove redundancies in the detectors' outputs.

Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is the most popular algorithm used for object detection task [Felzenszwalb et al., 2010; Girshick, 2015; Redmon et al., 2016]. The algorithm selects one bounding box among many overlapping items and partially ensures that each object is identified only once. The selection's criteria combines probability threshold and overlapping measure, such as IoU. As defined in Algorithm 3, for a list of proposal boxes B_{in} with its list of corresponding confidence score S_{in}, and an overlap threshold τ, the NMS algorithm gives a new list of filtered boxes B_{out} as follows:

1. **Select** the proposal bounding box with the highest confidence score in S_{in}.
2. **Remove** it from B_{in} and **append** it to B_{out}.
3. **Proposal comparison:** compute the IoU of the current proposal with every other proposal. If the IoU is greater than the threshold τ, remove that proposal from B_{in}.
4. **Repeat** from step 1 to step 3 until there are no more boxes in B_{in}.

3.3.3 Labeling of detections

Metrics for measuring the performance of object detectors are generally based on the threshold of IoU score. Calculating this score between the actual and predicted bounding box allows us to define the nature of each detection in terms of classification results. In this section, we focus on a binary problem (nodule vs. non-nodule regions), but the concepts can be easily generalized to problems for multi-classes problems.

Here, a few reminders of binary classifications definitions in ML. Interpretations are adapted for nodule detection problem:

- A **True Positive (TP)** corresponds to a nodule present in the image that has been correctly detected. The term "*correctly detected*" means that there is a predicted bounding box that has a IoU with the ground truth bounding box above a predefined threshold τ. Typically, $\tau = 0.5$ is used.
- A **False Positive (FP)** corresponds to all boxes wrongly identified as positives (i.e. containing an object). For nodule detection, the situation is similar to detect a nodule whereas there is none.

Algorithm 3: Non-Maximum Suppression algorithm

Input : τ: the NMS threshold
B_{in}: list of input bounding boxes
S_{in}: list of corresponding confidence score for each bounding box
$B_{out} \leftarrow \{\}$
$S_{out} \leftarrow \{\}$
while $B_{in} \neq \{\}$ **do**
  $i_{s_{max}} \leftarrow \arg\max(S_{in})$
  $s_{max} \leftarrow s_{i_{s_{max}}}$
  $b_{max} \leftarrow b_{i_{s_{max}}}$
  $B_{in} \leftarrow B_{in} \setminus \{b_{max}\}$
  $S_{in} \leftarrow S_{in} \setminus \{s_{max}\}$
  $B_{out} \leftarrow B_{out} \cup \{b_{max}\}$
  $S_{out} \leftarrow S_{out} \cup \{s_{max}\}$
  for $b \in B_{in}, s \in S_{in}$ **do**
    if $IoU(b, b_{max}) \geq \tau$ **then**
      $B_{in} \leftarrow B_{in} \setminus \{b\}$
      $S_{in} \leftarrow S_{in} \setminus \{s\}$
Output **:** B_{out}, S_{out}

- A **False Negative (FN)** refers to a missed object by the detector. It may be an absence of box on the object, or it may be boxes whose IoU value is below a certain threshold τ.
- A **True Negative (TN)** can be thought as a non-object regions where no detection has been made by the model. Because of the large imbalance between TNs and TPs, it is generally discarded for computing object detection metrics.

3.3.4 Precision-Recall curve

Regardless of the given confidence score, each bounding box on the image can be classified as TP or FP according to some criterion. For object detection, the IoU score with a fixed threshold $\tau = 0.5$ is classically used. In addition, the model may also have failed completely in locating some objects, which are then classified as FN. Consider a dataset consisting of G ground truths and a model with N detections. Assuming S output(s) as true positive(s) ($S \leq G$), the following equations describe the precision and recall values:

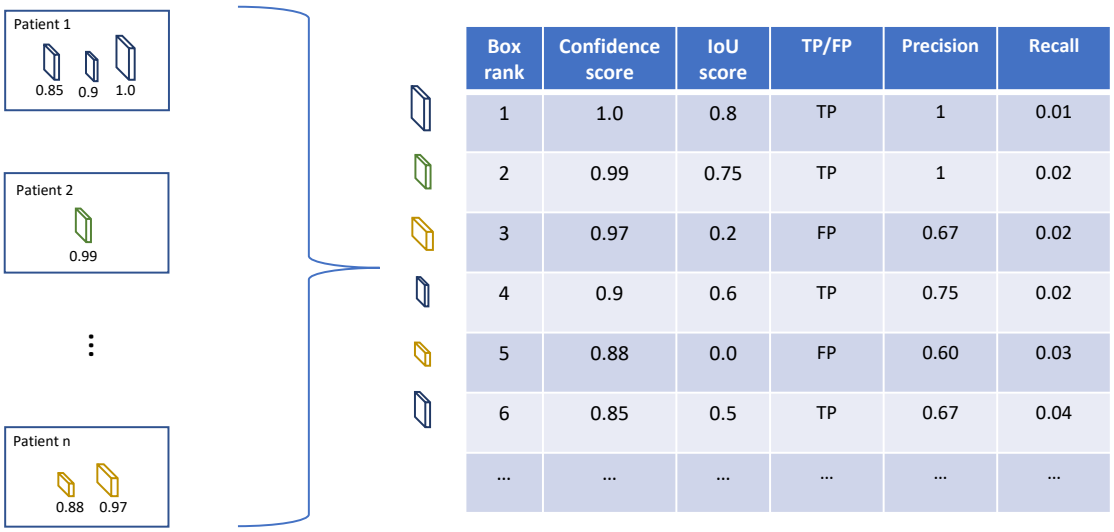
$$Precision = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{N-S} FP_n} = \frac{1}{N} \sum_{n=1}^{S} TP_n$$

$$Recall = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{G-S} FN_n} = \frac{1}{G} \sum_{n=1}^{S} TP_n \ ,$$

where sums simply refers to a count of detection with specific nature. Therefore, $\sum_{n=1}^{S} TP_n$ is the total number of TP detections.

Figure 3.7: An illustration of true positive (blue), false positive (red), false negative (orange) in object detection task adapted for pulmonary nodule detection.

Precision measures the accuracy of the model's predictions, *i.e.* the percentage of correct predictions. Recall measures how well the model find all real objects. Given a certain IoU threshold $\tau > 0$, all boxes can be labelled as TP, FP or FN, and a Precision-Recall (PR) curve is plotted for each class. Predicted boxes can be ranked in descending order according to their corresponding confidence score value. By iteratively using multiple confidence thresholds, each points of precision and recall coordinates can be obtained, as it is depicted in Figure 3.8.



Box rank	Confidence score	IoU score	TP/FP	Precision	Recall
1	1.0	0.8	TP	1	0.01
2	0.99	0.75	TP	1	0.02
3	0.97	0.2	FP	0.67	0.02
4	0.9	0.6	TP	0.75	0.02
5	0.88	0.0	FP	0.60	0.03
6	0.85	0.5	TP	0.67	0.04
...	...	...	...	...	...

Figure 3.8: Workflow for computing precision and recall values in a context of nodule detection task, assuming IoU threshold $\tau = 0.5$ and the amount of nodule to detect is $G = 100$.

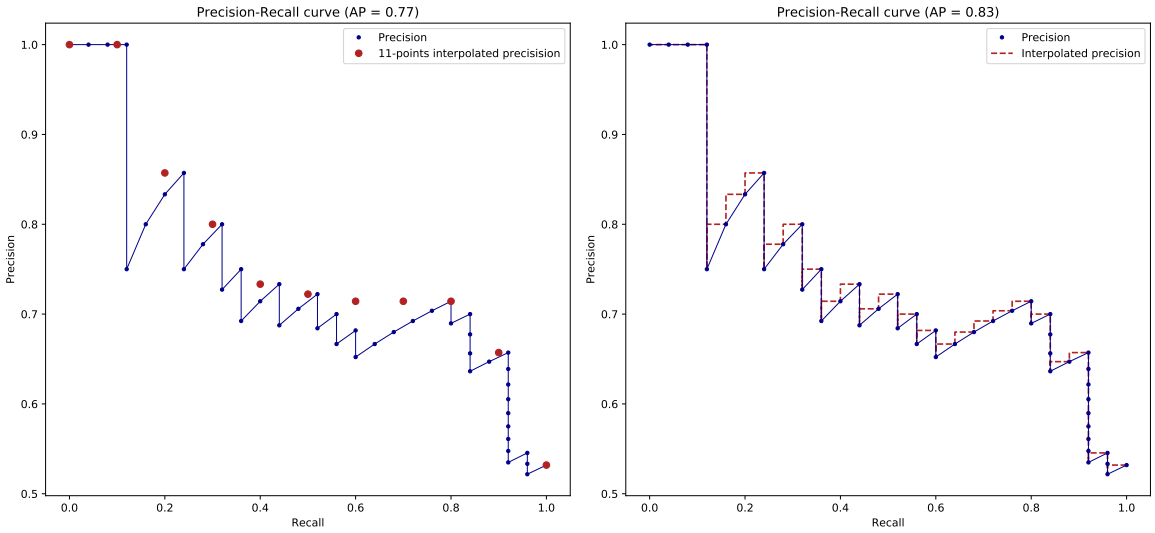


Figure 3.9: Precision-Recall curve with 11-points interpolation (left) and all points interpolation (right). The given average precision (AP) score depends on how many precision values are interpolated.

High recall but low precision values means that although the model has provided numerous boxes, most are FPs, while a small proportion actually contains a target object. In contrast, high precision with low recall means that most proposed boxes are TPs. For each object's class, a high performance object detector must maintain high precision as recall increases. By measuring the area under the PR, curve, the quality of the detections can be quantified. This score is named Average Precision (AP) score and can be formally expressed as follows:

$$\mathrm{AP} = \int_0^1 p(r)\, dr \quad .$$

Ranking detections leads to inadequate variations in the $p(r)$ function with a "wave" effect. With the introduction of PASCAL-VOC challenges in computer vision communities, the integral form is replaced by a finite sum over some recall values, and interpolations of precision values were suggested. AP computation proposed by Everingham et al. [2010] implies to interpolate precision over 11 points corresponding to recall values from 0.0 to 1.0 with 0.1 step:

$$\mathrm{AP} = \frac{1}{11} \sum_{r \in \{0.0, 0.1, \ldots, 1.0\}} p_{interp}(r) \quad ,$$

where $p_{interp}(r)$ is an adjustment made on the curve by assigning the maximum precision value for recalls values higher than r :

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \quad .$$

In addition to a lack of precision, this method cannot provide a satisfactory comparison between methods with a low AP score. This explains why a different AP calculation is adopted after 2008 challenges from PASCAL-VOC. The AP value is then computed by removing all zigzags in the curve and considering all recall values. This is equivalent to a sum of area of rectangles:

$$\mathrm{AP} = \sum_{k=1}^{n} (\mathrm{R}_k - \mathrm{R}_{k-1})\, \mathrm{P}_k \quad ,$$

Notation	Definition
$AP^{IoU=0.75}$	AP with an IoU threshold = 0.75
$AP^{0.5:0.05:0.95}$	Mean AP with IoU from 0.5 to 0.95 with a 0.05 step.
AP_{small}	AP for small object (Area $< 32^2$)
AP_{medium}	AP for medium object ($32^2 \leq$ Area $< 96^2$)
AP_{large}	AP for large object (Area $\geq 96^2$)

Table 3.2: Overview of AP metrics introduced by COCO challenge [Lin et al., 2014].

where P_k and R_k are respectively the precision and the recall values considering the top-k highest score bounding boxes detections. For multi object detection problem, AP can be computed for all classes and mean Average Precision (mAP) is then defined as the mean of AP for each class.

$$m\text{AP} = \frac{1}{\text{C}} \sum_{c=1}^{\text{C}} \text{AP}_c \ ,$$

where C is the number of class object. The AP score is commonly computed with a IoU threshold of 0.5, but latest COCO competitions computes AP through several IoU thresholds (see Table 3.2).

3.3.5 Free-Response ROC curve

With a perspective of global diagnosis, data labelling consists of assigning a category to each image. This procedure is compatible with Receiver Operating Characteristic (ROC) analysis, which graphically results in a ROC curve. Basically, the ROC curve represents the performance of a classification model by plotting *sensitivity* (synonym for recall) versus 1 - *specificity*, which denotes the false positive rate. For classification task, the calculation of the corresponding Area Under the Curve (AUC) score is considered a standard evaluation procedure for comparing models. However, this methodology has some significant limitations in the context of medical imaging, as it does not take into account the local interpretations required for some ROI to provide a global response at the patient level. As a result, the ROC analysis does not clearly reflect whether the detected regions are relevant to an accurate diagnosis.

With Free-response Receiver Operating Characteristic (FROC) paradigm [Bunch et al., 1978], ground truths are no longer limited to one rating per image, but include information about each abnormality. Using medical vocabulary, a lesion localisation (LL) characterizes a TP detection, while a non-lesion localisation (NL) outlines a FP detection. Similar to sensitivity, the Lesion localisatsion fraction (LLF) represents the ratio between the number of all LLs n_{LL} and the total number of lesions n_{lesion}:

$$\text{LLF} = sensitivity = \frac{n_{\text{LL}}}{n_{lesion}} \ .$$

Analogously, the Non-lesion localisation fraction (NLF) corresponds to the average number of FPs per scan:

$$\text{NLF} = \frac{n_{\text{NL}}}{n_{scan}} \ ,$$

with n_{NL} is the number of NLs and n_{scan} the total number of scans.

The FROC represents the plot of the collection of LLFs vs. NLFs values along the variations of a confidence score threshold ζ. The point $(0,0)$ refers to a threshold value ζ higher than all detection confidence scores (typically $\zeta = 1$ with softmax layer outputs). Gradually decreasing the threshold value allows to compute the relating LLF's and NLF's values and plot the whole curve. It should be noted, however, that a detection model does not necessarily always generate a bounding box for all lesions in the data. Thus, unlike the ROC curve, the FROC curve is not guaranteed to achieve a sensitivity of 1. Figure 3.10 shows an example of the graphical representation of ROC and FROC curves.

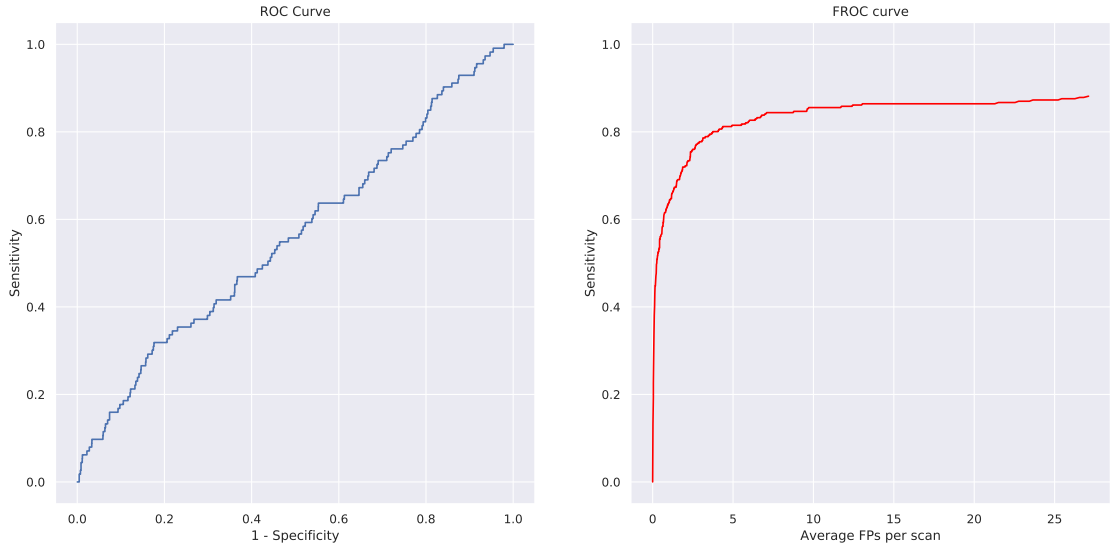


Figure 3.10: Illustration of ROC and FROC curve. In the medical context, the ROC curves are generally used to measure the quality of a global diagnosis, while the FROC curves show the quality of a detection system that can help in the diagnosis.

Competition Performance Metric (CPM) score

Establishing an appropriate Figure of merit (FOM) is crucial for performance evaluation of models. The value extracted from FROC data must be representative of the quality of correct detections while reflecting the poor ones. AUC score is commonly used with ROC data. However, the particularity of the FROC paradigm makes the comparison between detection systems more complex. Since the number of detections per image is not fixed a priori, curve's abscissas may extend beyond 1, then making the AUC scores irrelevant.

Nowadays, most CADx systems offer a parameter setting allowing to be more or less lax on the confidence scores attributed to the proposed detections. LUNA16 challenge evaluation metrics was based on Competition Performance Metrics (CPM) score Van Ginneken et al. [2010]. This metric refers to an average of LLF at 7 predefined NLF values (1/8, 1/4, 1/2, 1, 2, 4, 8):

$$\text{CPM} = \frac{1}{7} \sum_{i=\{1/8,1/4,1/2,1,2,4,8\}} \text{LLF}(i) \ .$$

Similar to a ROC analysis, the perfect object detection model will reach 1. In addition, measuring sensitivity at low levels helps to determine which models are best suited for routine clinical use.

3.4 Experiments and results

In this section, we propose a comparative analysis on the performance of each model presented in Section 3.2 with the metrics proposed in section 3.3.

3.4.1 Comparative analysis: 2D vs 3D models

Training setup

To determine which model is preferable in a clinical context, we examine the performance of 2D and 3D implementations of the following models: U-FPN, Mask R-CNN, Retina Net, Retina U-net. Each model was evaluated using the LIDC dataset with identical 5-fold cross-validation. Therefore, for each iteration, 80% ($n = 828$) of the scans are used as the training (60%) and validation (20%) set and the remaining 20% ($n = 207$) are used as the testing set. The 5 results from the folds are then averaged to obtain a single performance estimate. Each model was trained for 100 epochs with an Adam optimizer [Kingma and Ba, 2014] and a constant learning rate of 10^{-4}. For 2D models, the patch size was set to 300×300 and the batch size to 20. For 3D models, the patch size was set to $128 \times 128 \times 64$, limiting the maximum batch size to 10 for the U-FPN and 6 for all other models. Each training lasted on average 8 hours for 2D models, and 24 hours for 3D models, and was performed using an NVIDIA GeForce RTX 2080 Ti GPU card with 12 GB of graphical memory.

Testing phase

For each experience, the testing phase is applied in a loop to process each patient and includes 3 stages:

1. **Patching** the 3D scan into fixed size patches. To avoid detection issues at tile edges, we set for each dimension a minimum overlap of 25% between each neighboring patch. With our setting for patch size, this results in processing an average of 1275 patches per patient for 2D models and 150 patches per patient for 3D models.
2. **Inference** on all patches, and conversion of patch-level detection coordinates to scan-level coordinates.
3. **Filtering and/or post-processing**: The output segmentations of the U-FPN and Retina U-net models are converted to a bounding box list by applying a connected component algorithm. Also, for 2D models, the bounding boxes are transformed into a 3D cube by grouping the contiguous detections on successive slices. Regardless of the model used, all detections are filtered in 3D by applying the NMS algorithm.

Results analysis

For the purpose of comparability, each experience is evaluated in terms of both processing time and performance with AP and CPM scores.

Regarding the processing speed, we measure the average processing time of a patient for each model. These measurements are shown for each model in Figure 3.11. The graph tends to show that the parameterization we used greatly slowed down the average processing time of the scanner for all 2D models. This is mainly due to the significant difference in the total number of patches to be processed between the 2D and 3D models. On

the other hand, regardless of the configuration chosen, the fastest testing phase is performed with Mask-RCNN (average processing time of 38 ± 17 seconds/patient in 2D, 16 ± 6 seconds/patient in 3D).

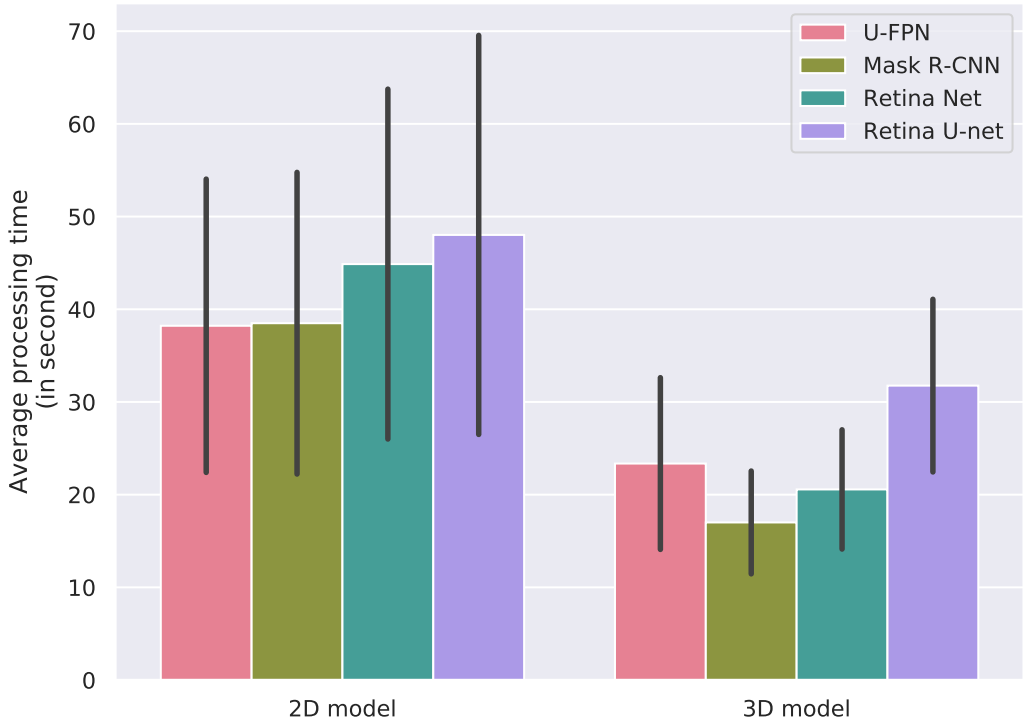


Figure 3.11: A comparison of the inference time for each tested model in 2D and 3D. Average processing time represents the average time required to process a full 3D scan of a patient. The bars represent the standard deviation of the measured times.

Model	Dimension	Number of outputs per patient (Mean ± stddev)	
		$s_{min} = 0.1$	$s_{min} = 0.2$
U-FPN	2D	30 ± 20	30 ± 20
	3D	20 ± 19	20 ± 19
Mask R-CNN	2D	463 ± 179	75 ± 48
	3D	98 ± 79	38 ± 31
Retina Net	2D	705 ± 92	56 ± 60
	3D	2597 ± 941	43 ± 31
Retina U-net	2D	745 ± 50	43 ± 33
	3D	2395 ± 1450	7 ± 5

Table 3.3: A comparative table of the average number of detections per patient for each model trained and then tested with the LIDC dataset as a function of the minimum confidence threshold s_{min}.

Performance is then evaluated using the AP and CPM metrics. To define TPs, the minimum IoU threshold is set to 0.1, which in this context corresponds to a coarse but suf-

Model	Mean sensitivity (stddev)						
	FPs = 1/8	FPs = 1/4	FPs = 1/2	FPs = 1	FPs = 2	FPs = 4	FPs = 8
U-FPN 2D	0.02 (0.00)	0.05 (0.01)	0.09 (0.01)	0.18 (0.02)	0.36 (0.03)	0.45 (0.03)	0.56 (0.04)
Mask R-CNN 2D	0.12 (0.05)	0.18 (0.07)	0.25 (0.07)	0.34 (0.07)	0.45 (0.08)	0.54 (0.07)	0.63 (0.06)
Retina Net 2D	0.09 (0.03)	0.12 (0.03)	0.18 (0.04)	0.25 (0.06)	0.33 (0.08)	0.41 (0.10)	0.50 (0.10)
Retina U-net 2D	0.09 (0.03)	0.13 (0.04)	0.19 (0.06)	0.27 (0.08)	0.37 (0.11)	0.44 (0.11)	0.54 (0.1)
U-FPN 3D	0.08 (0.03)	0.16 (0.07)	0.32 (0.12)	0.48 (0.13)	0.58 (0.07)	0.66 (0.05)	0.7 (0.04)
Mask R-CNN 3D	0.34 (0.06)	0.42 (0.08)	0.51 (0.06)	0.6 (0.05)	0.69 (0.05)	**0.76 (0.05)**	**0.82 (0.04)**
Retina Net 3D	0.28 (0.06)	0.36 (0.07)	0.46 (0.06)	0.55 (0.05)	0.64 (0.03)	0.72 (0.03)	0.76 (0.03)
Retina U-net 3D	**0.35 (0.07)**	**0.44 (0.06)**	**0.52 (0.05)**	**0.62 (0.03)**	**0.7 (0.03)**	**0.76 (0.02)**	0.78 (0.02)

Table 3.4: A comparative table of the different models tested with the LIDC dataset. Bold indicates the highest sensitivity for each level of average false positive per scan (FPs).

ficient localization. Although similar, measuring these two metrics provides an overview of the quality of detections provided by each model. The PR curves have the advantage of providing information about the distribution of FP and TP, whereas the FROC curves are easier to interpret and provide more of a patient-level view. In order to collect results, it is necessary to set a threshold for the minimum confidence score s_{min}, above which the results of the model are analyzed. In this sense, table 3.3 shows that too low a threshold (i.e., $s_{min} = 0.1$) can lead to an explosion in the number of detection results, especially for 3D Retina Net and Retina U-Net. This instability is not desirable in our context. However, if we set the threshold too high, there is a risk that predictions that could be true positive will not be considered. For our experiments, we therefore chose a threshold of 0.2, which seems to be an acceptable tradeoff. We can also note that, in a clinical context, the 3D Retina U-net model yields a more manageable average number of responses at this threshold (7 ± 5 detections per patients), indicating a lower number of false positives compared to other model.

Figure 3.12 illustrates the PR curves of each model for each fold of the cross-validation. These curves clearly show the superiority of the 3D implementations compared to their 2D equivalents. As for the comparison between the models, Mask R-CNN gives the best average AP (average AP = 0.29), while in 3D the results between Mask R-CNN and Retina Unet are quite equivalent (average AP = 0.57 for both). Note also that in all models, most curves fall within an interval between 0.75 and 0.9 for recall value, suggesting that in each fold about 10% to 25% of the nodules were not detected at all.

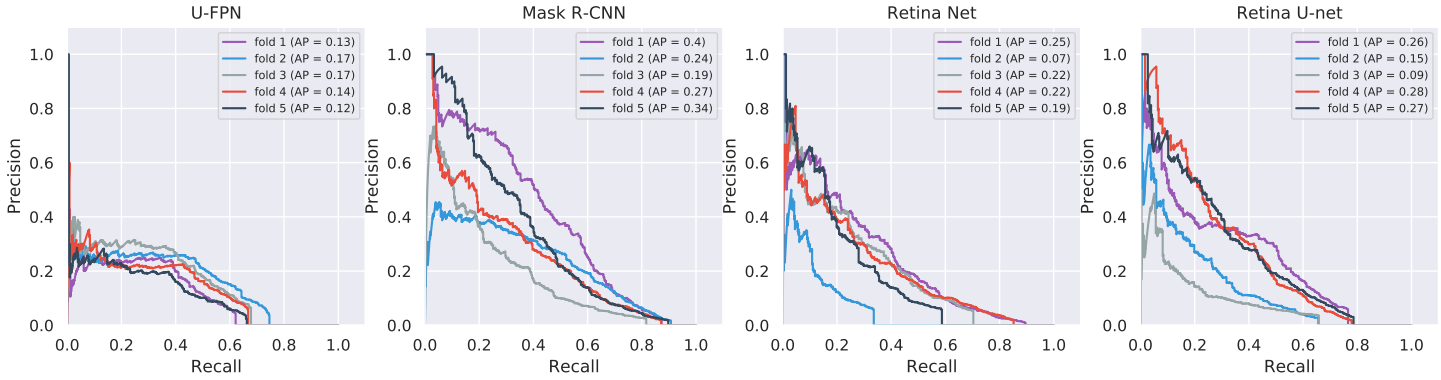
Figure 3.13 shows the FROC curve averaged over each fold of each image, and the average sensitivity values used to calculate the CPM score are shown in table 3.4. These results lead to similar conclusions as the previous ones: The R-CNN models for 3D masks and the retina U-mesh perform best in scoring CPM (average CPM = 0.59 for both). However, table 3.4 seems to indicate that Retina U-net has a slightly higher sensitivity than Mask R-CNN for low average FP values.

3.4.2 Filtering false positive detection

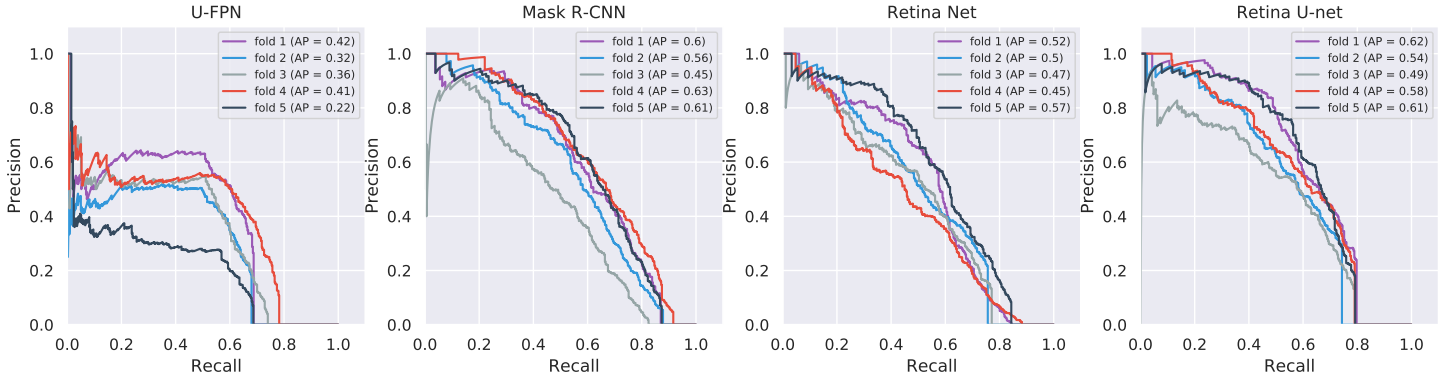
Lungs segmentation

Because image acquisition procedures are not necessarily standardized from one hospital to another, disturbances may occur during the inference process. There are many advantages to creating a lung segmentation mask for each patient. On the one hand, it can eliminate the most aberrant detections, such as those that might appear outside the lungs due to artifacts. On the other hand, the volume to be examined can be reduced in this way for large images.

For simplicity, our approach will basically be based on intensity threshold [Otsu,



(a) Precision-Recall curve for 2D models.



(b) Precision-Recall curve for 3D models.

Figure 3.12: Comparison of precision-recall curves between 2D and 3D models on a 5 fold cross-validation applied to the LIDC dataset.

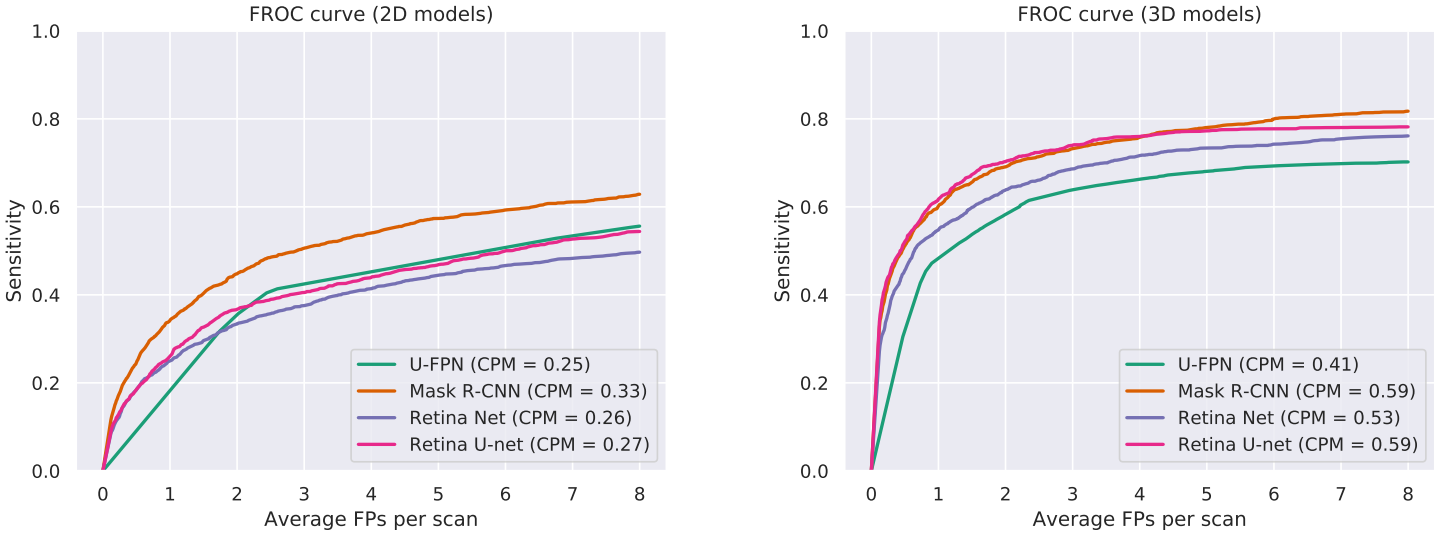


Figure 3.13: Average FROC curves for all 2D and 3D models tested on LIDC dataset.

1979], morphological operations, and a filtering on object sizes (see algorithm 4). Naturally, this is an approximate approach, since not all internal structures of the lung are considered. In addition, we have found that in very rare cases the process may result in lung pixels being connected to external elements (especially the bed). Figure 3.14 shows an example of 3D lung segmentation obtained by our approach. In this way, for each scan and for each 3D model, we were able to filter out the detections that did not overlap with the segmentation of the lungs. The results of the average CPM and AP metrics are presented in table 3.5, and show the improvement obtained for each model.

Algorithm 4: Lungs segmentation

Input : I: a 3D scan with standardized intensities

$M_1 \leftarrow I > \tau$ `// Binarize all voxels of I using Otsu threshold`

$M_2 \leftarrow$ fill_holes(M_1) `// Fill binary holes to obtain a primitive body mask`

Apply connected component algorithm on M_2 and keep the largest region only.
Obtain the body mask M_{body}.

$M_{th} \leftarrow I \leq \tau$

$M_{lung} \leftarrow M_{th} \times \mathbf{1}_{M_{body}>0}$ `// Remove positive pixels outside the body mask`

Apply closing transformation and connected component algorithm on M_{lung}.
Keep the largest region only to remove small artefacts.

Output : A lungs binary mask

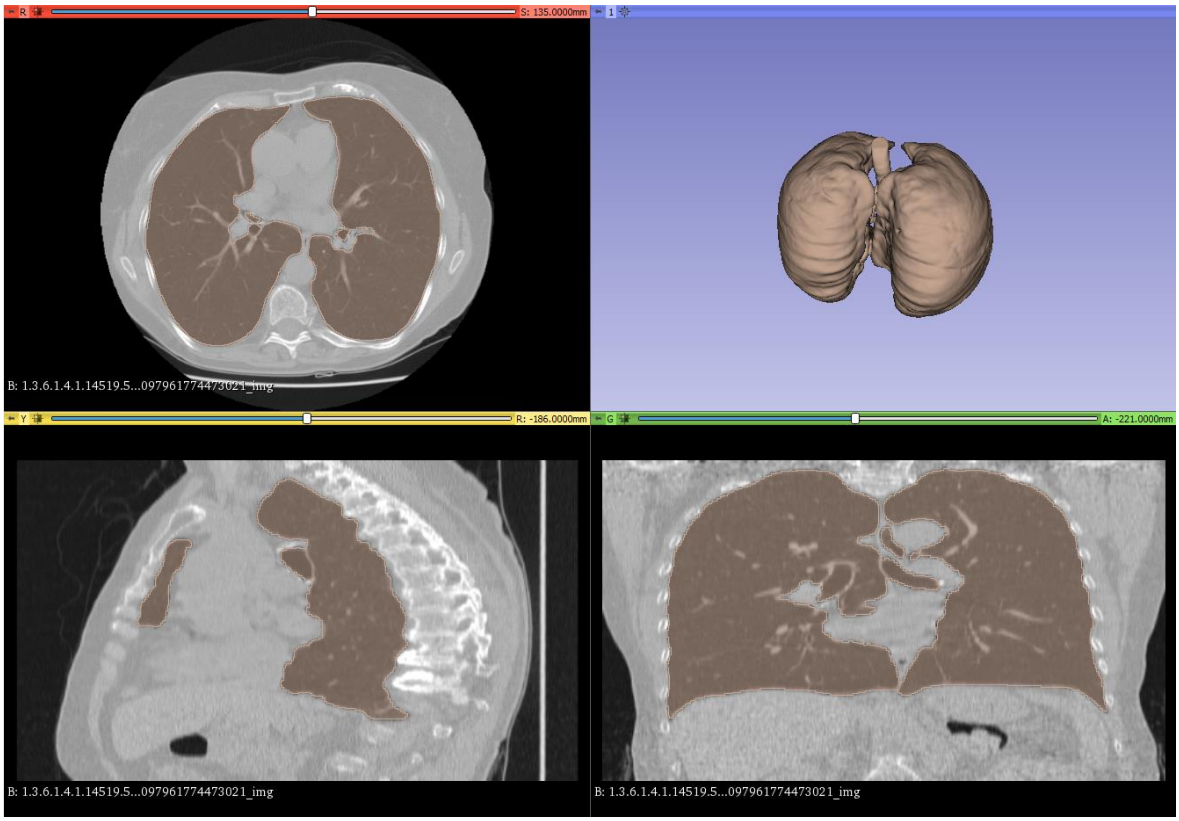


Figure 3.14: Visualisation of a lung scan CT under different views: axial (upper left), sagittal (lower left), coronal (lower right). The segmentation of lungs (brown) obtained with Algorithm 4 was overlaid in each view and the 3D representation of the segmentation is generated in the top right view. Any detection that does not overlay the segmentation is ignored when calculating score performances. This visualization was created using the open source software 3D Slicer.

3D model	Average CPM (stddev)	Average AP (stddev)
U-FPN	0.44 (0.06)	0.35 (0.07)
Mask R-CNN	**0.62 (0.06)**	0.57 (0.06)
Retina Net	0.57 (0.04)	0.50 (0.04)
Retina U-net	0.6 (0.03)	**0.57 (0.05)**

Table 3.5: A comparative table of average CPM and AP scores for each 3D model. These values are obtained after filtering detections outside of lung segmentation.

3.4.3 Influence of hyperparameters on nodule detection

To evaluate the impact of hyperparameters on the performance of detection models, we have made supplementary experiences with the 3D Retina U-net model. Using a single split of the LIDC dataset, we give here a brief description of the results obtained and our interpretations.

Patch size In this experience, different patch sizes were used for model training. The main idea was to investigate whether a smaller amount of data (and thus a larger batch size) would provide better performance and/or faster convergence of the model. Therefore, we repeated the same training with three different patch sizes: (64, 64, 64), (96, 96, 64), and (128, 128, 64). Our results showed that decreasing the patch size greatly increased the number of false positives, resulting in worsened results from CPM and AP scores.

Batch balancing Similar to the previous experiment, we wanted to observe the effects of foreground probability p_{fg} (see section 3.2.5) with the values 0.3, 0.5, 0.7. Actually, the default value $p_{fg} = 0.5$ is the best compromise. Thus, it seems to us that the most appropriate strategy is to choose a patch size that provides sufficient correlation with the location of the nodule in the lung.

Bounding box filtering algorithm Instead of applying the NMS algorithm, we tested the Weighted Box Clustering (WBC) algorithm proposed by Jaeger et al. [2018]. Following the same principle as the NMS algorithm, the WBC algorithm selects the box with the highest probability and weights its coordinates and score according to the neighboring boxes it overlaps. The weights depend on the size of the box, the position on the tile, and the total score. We show some visual examples in appendix A.1 (see figure A.1 and figure A.2). Despite an interesting reduction in the confidence score of false positives, using NMS gave a better AP or CPM score for all models tested.

3.5 Building a Computer-Aided System for patient diagnosis

3.5.1 Data challenge context

To extend the use of the nodule detection pipeline presented in the previous section, we participated in the 2019 edition of the Data Challenge organized by the French Society of Radiology (SFR) during the *Journées Francophones de Radiologie (JFR)* [Lassau et al., 2020]. The goal of this event was to bring radiologists and data scientists together on

important clinical issues by placing patient benefit at the core of the topics offered. Participants were offered three different topics: the classification of pulmonary nodules on 3D CT scans (challenge 1), the prediction of expanded disability status scale for multiple sclerosis on 3D MRI (challenge 2), and the assessment of abdominal Sarcopenia on 2D CT scans (challenge 3). Naturally, our team competed for Challenge 1 and was a collaboration between us, the IBM-Cognitive team in Montpellier and the Jean-Perrin Hospital Center in Clermont-Ferrand.

More specifically, challenge 1 is based on the medical observation that the risk of malignancy of a pulmonary nodule rises with increasing volume. In addition, the presence of small nodules (*i.e.* , nodules with a volume $< 100mm^3$) does not affect the probability of lung cancer occurrence [Horeweg et al., 2014]. Therefore, this challenge is presented as a classification task between patients. For each CT scan examined, the expected answer is either "does not contain nodules, or only small with a volume $< 100mm^3$" (negative class) or "probably has at least one malignant nodules, with a volume $\geq 100mm^3$" (positive class). In practice, to give one of the two possible answers, a recognition task is first required to identify the areas of interest. Then each area must be segmented to obtain an estimate of the volume of the detected object. By applying this method to all the detected areas, it is then possible to provide an overall response at the patient level. In the final stage, the responses for each patient are submitted in the form of a csv file containing the patient's anonymization identifier, the predicted class (negative or positive), and the confidence value for each class. The winner is determined by the method that achieves the best AUC value on the ROC curve generated from the csv file.

Database description

The dataset provided by the SFR consisted of 1031 3D scans DICOM CT divided into three batches of 343, 344, and 344 more images, submitted throughout the competition[8] (which lasted about a month). The first batch served as the training data set for the first month of development and was manually annotated to indicate the location of nodules greater than 100 mm^3 in volume. The second batch was also annotated and shipped 3 days before the day of the competition. Thus, the last batch is the test dataset, which was provided without annotations and serves as a reference to determine the winning team.

3.5.2 A three-stage pipeline for patient diagnosis

This section describes the structure of the pipeline that we proposed during the competition. The latter is divided into three parts: lungs segmentation, nodule detection, feature extraction, and nodule classification.

Lungs segmentation stage In this context, we used the approach described in section 3.4.2 to create an annotated lung segmentation dataset from the JFR dataset. Errors that may arise from this approach were corrected by hand. This dataset was then used as a training set for a 3D U-FPN model using a Dice loss function (see section 3.2.4).

Nodule detection stage Regarding the nodule detection phase, the labeling procedure was not harmonized between data providers. Thus, nodule localization could also be

[8]Important note: this challenge was developed fully in compliance with the General Data Protection Regulation (GDPR). Therefore, the participating teams have worked only with anonymized images during the competition and have committed not to keep any data after the competition.

accomplished by marking a voxel in the center of the nodule or by marking a wide circle on a single slice that did not map the contours of the nodule. We therefore decided to train a Retina U-net model on the LIDC dataset. In this context, the model was trained for 400 epochs using Adam optimization algorithm with a learning rate reduction strategy that starts at 10^{-3} and decreases by a factor of 10 every 100 epochs. The training was performed using a Volta V100 NVIDIA GPU with 32 GB of graphics memory, allowing to set a batch size of 16 and a patch size of $(128, 128, 64)$.

Nodule classification stage Our approach to provide a global patient-level answer is to perform a binary classification task for each detection given by the model in the previous step. For this step, we use a linear SVM model that we train using the features we extracted from each bounding box. Extracted features are:

- Confidence score given by the model.
- Bounding box center coordinates, dimensions (height, width, depth) and volume.
- Minimum, maximum and median of voxel intensities.
- Segmentation features. Using 5 intensity thresholds evenly distributed between the minimum and maximum intensity in each box, we measure the number of objects present, the average volume of each object, the average intensity of voxels above the threshold, and the average intensity of voxels below the threshold. The same procedure is applied for each segmentation provided by the detection model in each bounding box.

Based on all these features, we were able to achieve an AUC value of 0.881 when training the SVM model with the JFR dataset. To optimize the training, we used recursive feature elimination (RFE) [Sanz et al., 2018] as a feature selection algorithm. Thus, the model was able to achieve a score of 0.905 with only 10 features. We kept this model in the final phase of the JFR and were able to win the competition with an AUC score of 0.899 (see table 3.6).

Teams	AUC score
Aidence	0.878
Autonomous	N.S.
GAMC	0.793
Our team	**0.899**
LEVIATAN	N.S.
LyPhTe	0.838
NAIS	0.681
ONCONEURAL	0.644
Owkin	0.768

Table 3.6: AUC score results calculated for each team participating at lung cancer classification challenge during the JFR. Bold indicates our score. N.S. indicates that the team left the challenge. Results extracted from Lassau et al. [2020].

3.6 Conclusions and perspectives

The development of a diagnostic tool based on artificial intelligence may prove particularly useful and beneficial in the medical field. In this chapter, we have specifically addressed the problem of early detection of pulmonary nodules in 3D CT scans. The MDT

framework developed by Jaeger et al. [2018] provides a variety of CNNs implementations suitable for 2D or 3D object detection, as well as a batch processing strategy adapted to this context. This allowed us to perform a comparison of the different state-of-the-art models in terms of performance. This allowed us to confirm certain trends:

1. Although this approach requires fewer computational resources, a 2D model that operates separately and independently for each slice of a 3D scan appears to provide significantly lower recognition performance than a corresponding model that operates directly in 3D. This is true for traditional segmentation models (U-FPN) as well as for object detection models (Mask R-CNN, Retina Net, Retina U-net). Thus, these results show that a 3D representation is required to achieve good recognition performance.
2. By selecting the most efficient models, we can achieve very low processing times, which in our experience are between 15 and 30 seconds. Even though this time depends on the set number of patches per scan, it remains far below the reading times that radiologists can achieve without a CADx tool [Beyer et al., 2007; Hsu et al., 2021; Vassallo et al., 2019].
3. Overall, the Mask R-CNN model appears to be somewhat more efficient at detection compared to the other models examined. However, The number of detections generated on average is very high, which is not desirable in a clinical context, and it also inevitably leads to more false positives. Although the detection scores we measured are somewhat lower, the Retina U-net model appears to be more stable above a certain probability threshold.
4. Even with an efficient detection model, a strategy to reduce false positives seems necessary to improve performance results. Our approach to lung segmentation is simple and allows us to eliminate the most obvious aberrant detections.

Finally, these results have allowed us to implement in a short time a simple solution that provides a comprehensive response to patients. The solution proposed in the JFR provides very encouraging results, especially in terms of diagnostic support. This type of event also allows for broader collaboration between French hospitals, which may prove beneficial, for example, in the merging of annotation procedures.

Of course, some aspects of our method can still be optimized: for example, we did not use a multi-class model so that the detection model can directly distinguish between benign and malignant nodules. In addition, the nodule classification model that we used during the JFR is based on simple features that are probably somewhat redundant. In retrospect, using 3D-CNN to classify each region with a nodule seems to be a better standard method [Kaliyugarasan et al., 2021]. Finally, in our study, we consider our system to be fully autonomous. To get closer to a real-world application, we would need to measure the performance gain that radiologists could achieve by using such a tool.

Chapter 4

Training neural networks on synthetic microscopy images

Abstract

To improve the robustness of CNN models during training, a large amount of data is required. However, access to a large amount of data, which must also be of high quality, is a problem that arises regardless of the application domain. In this chapter, we specifically address microscopic imaging, which is inextricably linked to biomedical research. After a basic description of acquisition systems in section 4.1, we propose in section 4.2 the development of a generative model for synthetic data. We describe each step necessary to generate a new fictitious microscopy image and its associated groundtruth. In section 4.3 and section 4.4, we present the most commonly used models for segmenting cell nuclei in the literature and explore the metrics that can be used to evaluate the segmentation quality of a model. Finally, in section 4.5, we propose to evaluate the impact of using simulated data in the training set on the performance of these models.

Contents

4.1 Introduction 60

4.1.1 Microscopy imaging 60

4.1.2 The need for data 61

4.2 Modeling microscopy image 62

4.2.1 Nuclei shape 62

4.2.2 Population dispersal 64

4.2.3 Increasing population density 64

4.2.4 Nuclei contouring 66

4.2.5 Texture model 67

4.2.6 Artifacts and noises 67

4.2.7 Simulation scenario 69

4.2.8 Extension for 3D simulation 70

4.2.9 Graphical interface 72

4.3 Models for nuclei segmentation 73

4.3.1 Stardist 73

4.3.2 Cellpose 74
4.4 Segmentation metrics 75
4.4.1 F1-score 75
4.4.2 Jaccard Index (JI) 76
4.4.3 Aggregated Jaccard Index (AJI) 76
4.5 Experiments and results 76
4.5.1 Biomedical image datasets 77
4.5.2 Extraction of simulation parameters from real images 77
4.5.3 Training setup 78
4.5.4 Reducing of the amount of training data 79
4.5.5 Mixing real and simulated data as a training dataset 80
4.6 Conclusions 85

4.1 Introduction

4.1.1 Microscopy imaging

Most modern microscopes consist of a lens and an eyepiece. The combination of these two optical elements allows magnification and then high-resolution projection of the analyzed biological sample, so that it is visible to the human eye. The cell is the basic biological unit of all known living beings and plays an important role in the functioning of our organism. Microscopy and cells are inextricably linked in biomedical research, and the visualization of specific cellular organs, such as the nucleus, cytoplasm, or protein elements, has greatly increased our knowledge of cellular structures and the mechanisms in which they are involved [Huang and Murphy, 2004].

To obtain the clearest and highest contrast image possible, the simplest microscopy technique, called *brightfield*, consists in illuminating the biological specimen completely with a uniform white light source. However, since cells and their components are largely transparent, this technique is not suitable for their observation. The most common solution used in biology nowadays is the use of a contrast agent in the preparation of the sample. In fluorescence microscopy, this agent is usually a fluorescent biomarker that reveals the presence of certain molecules of interest (*e.g.*, DNA). For example, DAPI[1] is a well-known marker for the DNA that absorbs ultraviolet (UV) light and stains cell nuclei blue as in Figure 4.1.

Image analysis is the natural step that follows data acquisition. The goal is often to obtain quantitative results based on the analysis of the cell population under study. It is then necessary to individualize each cell to extract some features that allow relevant biological interpretations, such as changes in cell size/morphology or pixel intensities. Due to the large number of cells to be identified, automated methods are preferred over manual and/or visual methods for this task. In addition, automated quantification provides far more reproducible information and is less sensitive to the subjectivity of an expert analysis. In practice, conventional image processing techniques can be applied, such as thresholding the pixel intensities [Otsu, 1979] and applying a seeded-watershed algorithm [Beucher, 1979; Malpica et al., 1997; Wählby et al., 2004]. The segmentation parameters of

[1] Molecule name for 4',6-diamidino-2-phenylindole

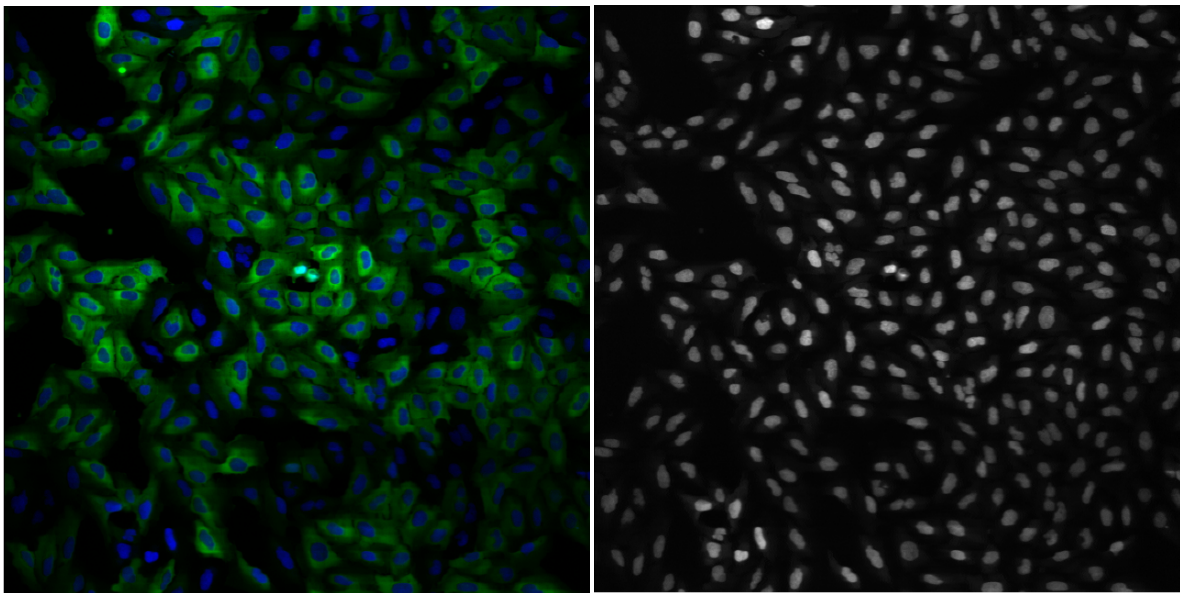


Figure 4.1: Human osteosarcoma U2OS cell line visualized by fluorescence microscopy. The left image shows the cytoplasm (green staining) and nuclei (blue staining) of the cell line. The right image is the blue channel showing only the cell nuclei. Image publicly available at Broad Bioimage Benchmark Collection website.

these methods often require a certain level of expertise to be correctly tuned. In addition, performance can be severely affected depending on the acquisition modality, the scale, or the experimental conditions. In contrast, Deep Learning methods have also shown great potential for segmenting nuclear instances in different types of images, without requiring further user interactions [Caicedo et al., 2019]. Therefore, this will be the preferred approach we will use in this chapter to solve nuclear segmentation problems.

4.1.2 The need for data

Regardless of its architecture, a neural network must use a large amount of data during its training. Most importantly, a high-quality dataset, when properly labeled, contributes to the robustness of the model and allows it to make better predictions for new data that have never been observed before. Cho et al. [2015] has thus shown that the accuracy of a model depends strongly on the size of the data set on which it is trained. As a result, numerous initiatives have been launched in the scientific community to build and publish image databases with rigorous annotations [Clark et al., 2013; Ljosa et al., 2012; Wang et al., 2017]. However, in practice, especially in the biomedical domain, the massive acquisition of labeled data remains a difficult task. A first reason for this is the difficulty of coordinating and harmonizing data acquisition and labeling between different research teams. Second, the provisions of the General Data Protection Regulation (GDPR) on the use and processing of personal data in the European Union are an important aspect to consider when conducting a research project. Finally, labeling data, especially when it involves pixel-wise annotations, is a rather tedious task that must be performed by humans.

An alternative workaround to this problem is to use a synthetic dataset for training deep neural network. This method also shows promising results for segmentation problems [Toda et al., 2020; Tremblay et al., 2018; Zhuang et al., 2019]. Compared to real data, synthetic datasets have the advantage of being very easy to obtain, and can also represent any situation, no matter how rare. Thus, a tool capable of generating an almost infinite

amount of quality data solves both the data acquisition and labeling problem. However, the algorithmic construction of such a tool requires a thorough knowledge of the application domains as well as an adequate modeling of all aspects that contribute to the formation of an image of interest. In this chapter, we propose a microscopy image simulation model with simple and concrete parameters and then investigate its impact on the performance of models suitable for nuclei cell segmentation.

4.2 Modeling microscopy image

The generation of new synthetic data corresponds to a sequence of different steps that first lead to a labeled mask of the cell nuclei whom pixels values are integer. Each value corresponds to a unique identifier for belonging to a particular nucleus (except for the value 0, which is assigned to the class "background"). The resulting mask stands for a synthetic ground truth. To obtain the corresponding image, we degrade the pixel values using various methods that mimic the acquisition procedures of microscopic systems. Thus, a simulation iteration corresponds to the generation of two results (images + nuclei mask) whose properties are determined by certain parameters, which we explain in more detail in this section.

4.2.1 Nuclei shape

As in a real acquisition process, the basic unit of our simulations corresponds to the cell nucleus. When analyzing a microscopic image, the simplest visual features are the size and/or shape of the cell nuclei. This is thus the first aspect we propose to model. Intuitively, nucleated objects are generally more or less blob-shaped. To obtain a sufficiently large choice of different shapes, we introduce a parameterized model to generate random polygons. Similar to Lehmussola et al. [2007], the model workflow starts using a parametric representation of an ellipse. The coordinates $\big(x(\theta), y(\theta)\big)$ of each point on an ellipse whose center is the origin $(0,0)$ are defined as follows:

$$\begin{aligned} x(\theta) &= r_x \cos(\theta) \\ y(\theta) &= r_y \sin(\theta) \ , \end{aligned} \tag{4.1}$$

where $\theta \in [0, 2\pi]$, and r_x, r_y are the radius on the x and y axes respectively. The special case $r_x = r_y$ corresponds to a circle. By sampling θ with constant interval, (4.1) allows to generate a set of points corresponding to the vertices of a polygon. However, the shape of a circle or ellipse is too rudimentary and regular to be considered realistic enough to simulate cell nuclei. Thus, we increase the variability of the generated shapes thanks to the 2D/3D object deformation model proposed by Durrleman et al. [2014]. Let $\{c_k\}_{k=1,2,\dots,n_v}$ denote a set of "control" points corresponding to the n_v vertices of a polygon, and $\{\mu_k\}_{k=1,2,\dots,n_v}$ a set of momentum vectors, where each $\mu_k \in \mathbb{R}^d$ is associated with each control point. The displacement of any control point c_i is given by the sum of radial basis functions K located at control point positions $\{c_k\}_{k=1,2,\dots,n_v}$:

$$v(c_i) = \sum_{k=1}^{n_v} \mathrm{K}(c_i, c_k)\mu_k \ , \tag{4.2}$$

where $\mathrm{K}(\cdot)$ is commonly a radial Gaussian kernel $\mathrm{K}(x, y) = \exp\big(-\|x-y\|^2/\sigma_d^2\big)$ and $\sigma_d > 0$ controls how far the deformation pattern spreads, so the final polygon looks sharper when

it is small. To avoid too large deformations, we also add a parameter α that controls the amplitude of the displacement. Figure 4.2 gives an overview of the shapes that can be created by using the same momentum vectors and by adjusting the values for α and σ_d. Finally, the x, y coordinates of the vertices of each new polygon are calculated using a combination of (4.1) and (4.2):

$$\begin{aligned} x &= x(\theta) + \alpha v(x(\theta)) \\ y &= y(\theta) + \alpha v(y(\theta)) \quad . \end{aligned} \tag{4.3}$$

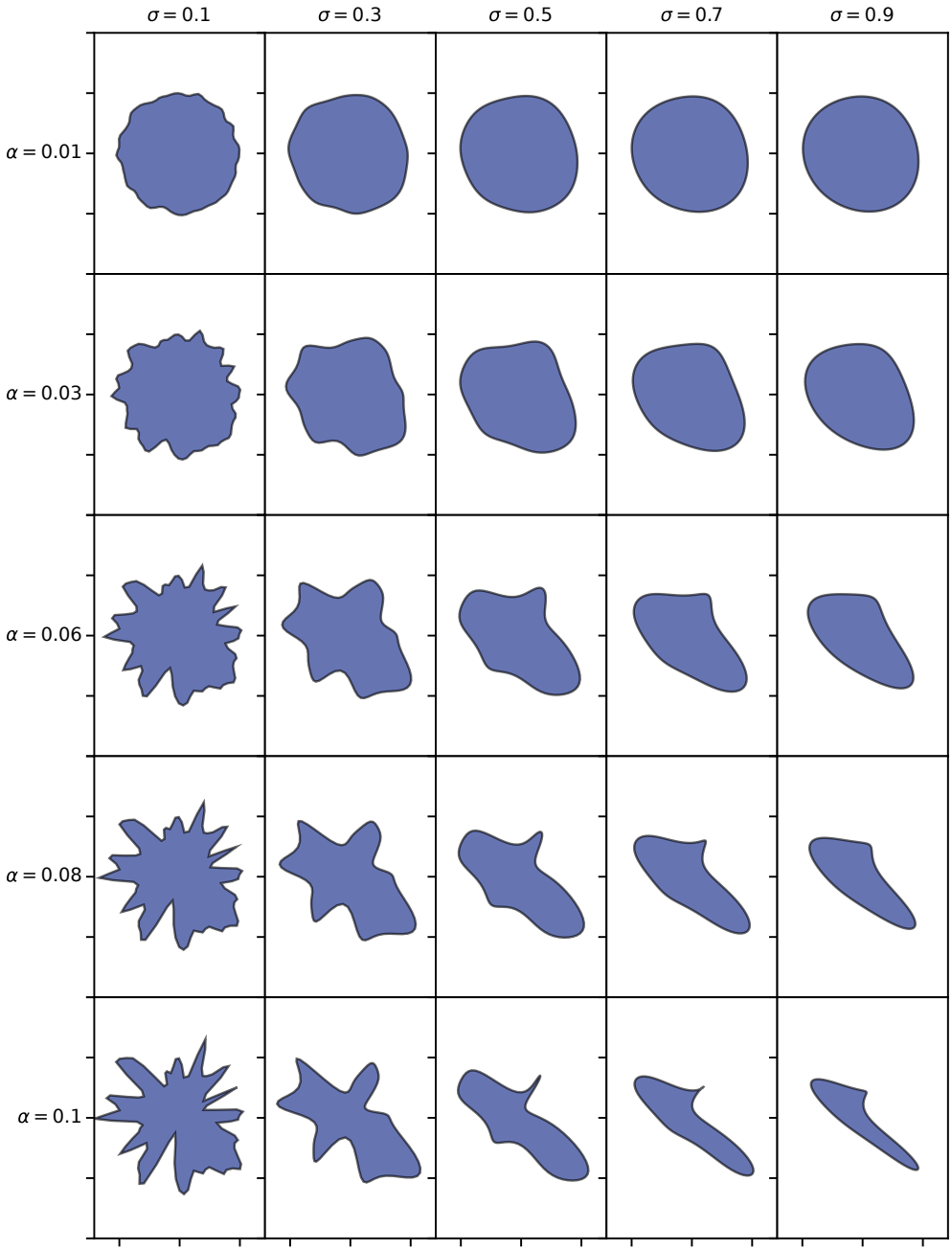


Figure 4.2: Overview of possible deformation applied to a disc of radius 1. The parameter α controls the magnitude of the distortion, while the parameter σ is a smoothing parameter. For the generation of cell nuclei, forms with values of $\alpha \geq 0.5$ and $\sigma \leq 0.06$ are preferred.

To give the generated nuclei a privileged orientation, we apply a rotation of β degrees to all vertices as follows:

$$\begin{pmatrix} x_\beta \\ y_\beta \end{pmatrix} , = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} , \tag{4.4}$$

where x_β and y_β are vertex coordinates rotated by β degrees. By applying (4.4) to a set of closely spaced nuclei, it is possible to give global orientation to certain regions of the image.

4.2.2 Population dispersal

After the individual characterization of each nucleus, the next important step is to model their spatial arrangement in the image. This step is performed under the constraint of a maximum allowable overlap rate O_{max} between the nuclei:

$$\frac{n_{px,inter}}{n_{px,new}} \leq O_{max} ,$$

where $n_{px,new}$ denotes the number of pixels belonging to the currently generated nucleus, and $n_{px,inter}$ is the corresponding number of pixels overlapping with other nuclei. A high overlap rate may result in nested nuclei, which may lead to an unrealistic situation and make the morphological aspect defined in the previous section obsolete. Therefore, for each newly placed nucleus, it must be checked whether the overlap requirement is met. If this is not the case, another location must be found for the placement. To improve efficiency, this aspect encourages us not to place each nucleus completely randomly. Instead, we opt for a strategy that requires as few placement attempts as possible.

For computational efficiency, we initialize a map corresponding to the possible location of each nuclei centroid. For this purpose, the centroids are arranged according to a Poisson-disc distribution [Bridson, 2007] that allows all objects to be placed evenly while maintaining a minimum distance d_{min} between each nucleus. Figure 4.3 shows the application of this algorithm to a 512×512 image with various minimum distances $d_{min} = 30, 60, 90, 120$. Roughly speaking, the algorithm is based on a loop in which one point at a time is randomly selected from a list of active points, and each new candidate generated from the current active point is checked for validity. If none of the k candidates are acceptable, *i.e.* , if there is at least one point less than d_{min} away for all candidates, the selected active sample is marked as inactive and is no longer used to generate candidates. When no more samples are active, the algorithm terminates. We also introduce a distance parameter to the border of the image d_{border}, which allows filtering the list of output points to avoid generating fragmented nuclei.

4.2.3 Increasing population density

For both the human eye and any automated method, the main challenge in segmenting cell nuclei appears when they are arranged in very dense clusters. Higher population density makes the task of finding good boundaries between cell nuclei very complex. The errors often observed in these situations correspond to either over-segmentation (*i.e.* excessive division of each object into multiple fragments) or under-segmentation (*i.e.* a tendency to group several nuclei together).

To increase the complexity of the generated data and improve the robustness of the segmentation models used, we propose a procedure to increase the population density. The main idea, shown in Figure 4.4, is to increase the number of average neighbors in the

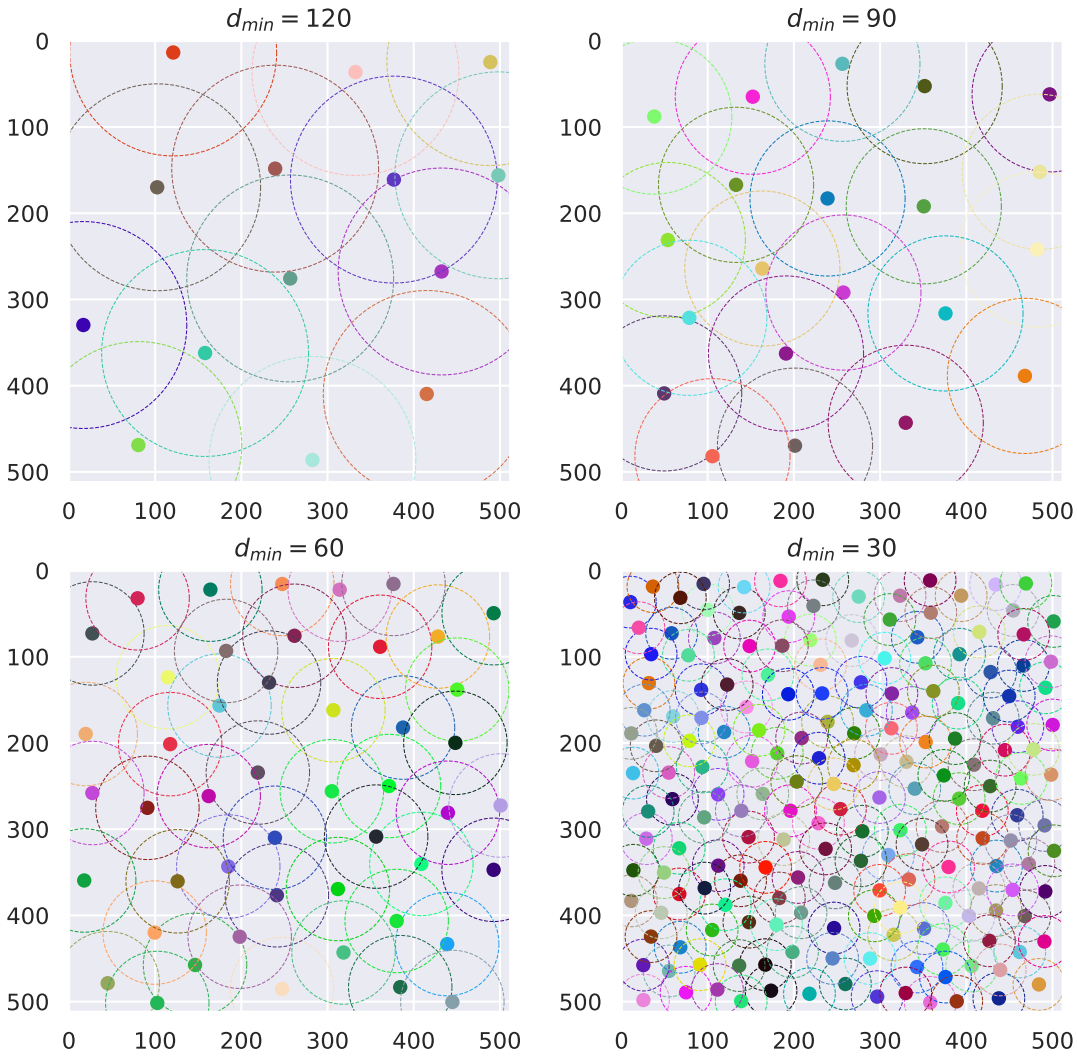


Figure 4.3: Visualization of cell population dispersion as a function of distance constraint d_{min}. The dotted circles show the minimum distance between a point and its nearest neighbors. Each point then serves as a candidate for the centroid of a newly generated nucleus.

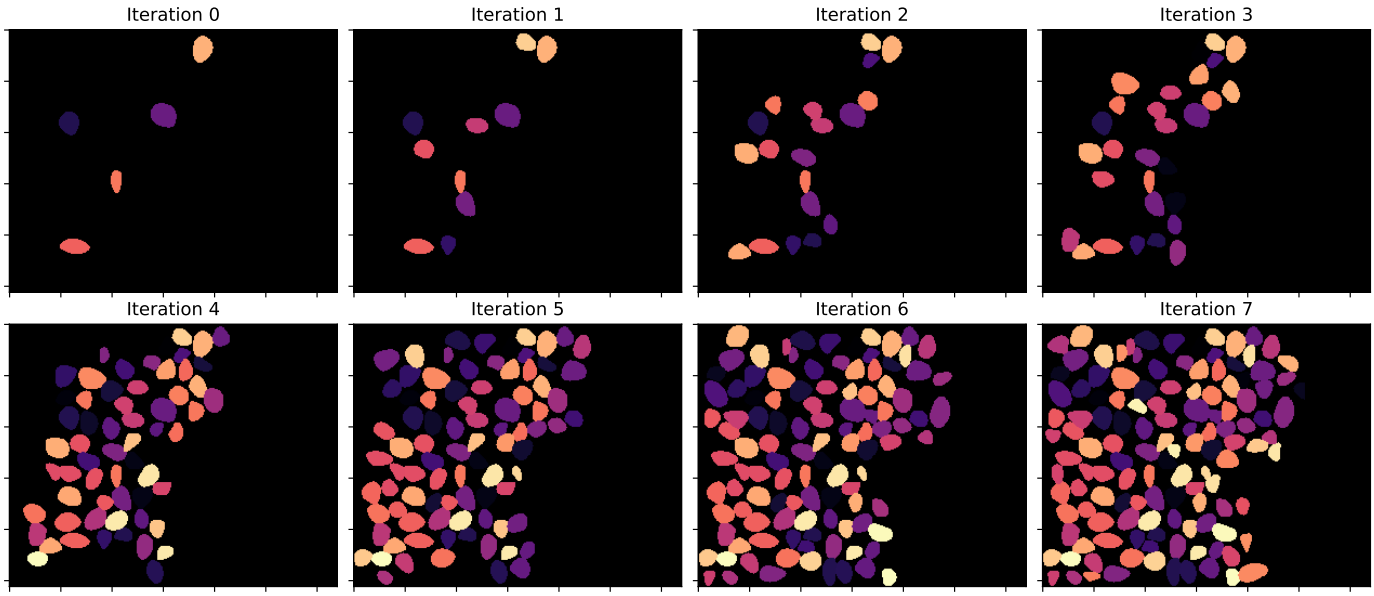


Figure 4.4: Operation to increase the population density of the nuclei generated by our simulation. At each iteration, the algorithm attempts to add one or more neighbors to each nucleus already present on the image, subject to the maximum overlap condition.

entire population. The concept of "neighborhood" is defined here relative to the size of the nucleus in question. In other terms, a nucleus with centroid p_1, constructed with radii r_x and r_y, and another nucleus with centroid p_2 are said to be neighbors if the following equality holds:

$$d(p_1, p_2) \leq \gamma \max(r_x, r_y) \ , \tag{4.5}$$

with $d(\cdot)$ the Euclidean distance and $\gamma > 1$ a factor regulating the neighborhood searching area. Empirically, we have observed that $\gamma \in [1.5, 2]$ gives a visually acceptable result. Finally, using (4.5), we can place a neighbor for each nucleus in the image. To limit the number of iterations for large populations, we also add a parameter regulating the maximum number of neighbors. This way, nuclei that have reached this threshold are then ignored in the procedure.

4.2.4 Nuclei contouring

Several segmentation methods rely on a ground truth of the contours of the cell nuclei. By combining a morphological mathematical operation, such as erosion, and element-wise subtraction, we can create a contour mask of each nucleus (see Figure 4.5b). To vary the thickness of the nuclei contours, the distance between each vertex and the centroid is reduced, resulting in a "hole" based on a smaller deformed version of the original polygon. This mask can be assimilated with a nuclear membrane mask (see Figure 4.5c), and the reduction factor can be linked with the membrane thickness. Like the nuclei mask, this mask can be degraded to deal specifically with the pixels at the edges of the nuclei, which are usually the most difficult to classify if they are located in very dense regions.

(a) Binarized nucleus mask (b) Nucleus contours mask (c) Membrane mask

Figure 4.5: Visual comparison of segmentation masks created for a single nucleus. (a): corresponds to a (binarized) nucleus mask, (b) is obtained by eroding (a) and subtracting the result from (a), (c) is obtained by applying a subtraction to (a) based on a reduction of the initial polygon used for (a).

4.2.5 Texture model

Chromatin is an essential macromolecular structure of the nucleus and consists of a mixture of proteins, RNA and DNA. The study of chromatin intensity profile can provide important information about cell activity [Rousselle et al., 1999]. Therefore, texture is another important feature to model in order to achieve a convincing visual aspect.

Basically, creating texture on an image is like adding noise. One of the most typical textures generator is the Perlin noise [Perlin, 1985]. Classically, Perlin noise is often used in computer graphics as a visual effect to create more realistic textures of various parts of the natural world (clouds, waves, mountain ranges, smoke, etc...). A visualization of this noise can be seen in Figure 4.6. The function used provides a texture map with a pseudo-random aspect that can be configured to regulate the level of detail of the final rendering. For our model, we rely on the framework by Lehmussola et al. [2007] which computes the value of the texture map t at location (i, j) as follows:

$$t(i,j) = \mathrm{B} + \sum_{k=0}^{n-1} p^k \eta_{i,j}(2^k) \ ,$$

where B is an intensity bias, $\eta(\cdot)$ is a noise function that produces noise at a fixed frequency, n is the number of octaves to iterate, p is the persistence parameter that determines how much each octave contributes to the total shape.

4.2.6 Artifacts and noises

After modeling all the specific features of the nuclei, we refine our simulation model by replicating their environment and the visual artifacts to which they might be exposed.

Uneven illumination

Uneven illumination is a common problem in image microscopy [Smith et al., 2015]. This phenomenon causes certain areas of the image to be significantly brighter (usually in the center), while other areas are quite shadowy and not very intense. Therefore, there can be significant variations in pixel intensity within an image. In our context, signal attenuation is generally assumed to be quadratic [Klein et al., 1998; Lehmussola et al., 2006; Svoboda

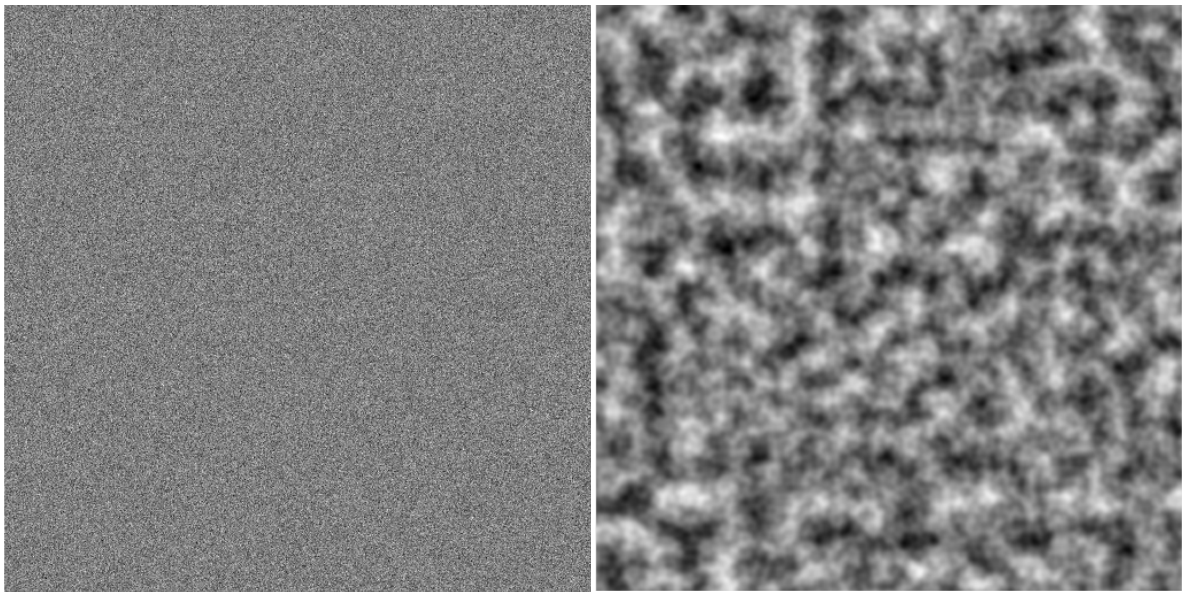


Figure 4.6: Left: Gaussian noise, right: Perlin noise.

et al., 2009]. For simplicity, we create a light exposure map with a quadratic exponential kernel to obtain a distance map from a randomly chosen reference point p in the image:

$$\mathrm{K}(p, p') = \exp\left(-\frac{\left\|p - p'\right\|^2}{2\sigma^2_{light}}\right) ,$$

where $\left\|p - p'\right\|$ denotes the distance from point p' to point p, and σ_{light} determines in which scale the light spreads in the image.

Point Spread Function (PSF)

Regardless of which optical system is used during the acquisition, the image result is not quite "true to reality". The point spread function (PSF) is a property of an imaging system that determines its impulse response to a point source. In other words, the PSF characterizes the optical aberrations that occur during the acquisition and thus produce a certain degree of blur. The most common way to model the PSF of a microscope is to apply Gaussian filtering [Kaufman and Tekalp, 1991], which amounts to convolving the image with the following function:

$$\mathrm{G}(x, y) = \frac{1}{\sqrt{2\pi\sigma^2_{\mathrm{PSF}}}} e^{\left(-\frac{x^2+y^2}{2\sigma^2_{\mathrm{PSF}}}\right)} ,$$

where x and y are the distance from the origin on the horizontal and vertical axis respectively, and σ^2_{PSF} is the kernel variance controlling the amount of blur.

Charge-coupled device (CCD)

The charge-coupled device (CCD), an essential element of a light microscope, is a photoreceptor system that captures photons reflected from an object. Each captured photon generates an electrical charge that is recovered by the camera's electronics to convert the

signal into an image. This process is a source of noise, and the arrival interval of the photons is modeled by shot noise, also called Poisson noise [Svoboda et al., 2009]. Alternatively, the background noise, as suggested by [Comeau et al., 2006], can also be approximated by an additive Gaussian noise with zero mean and variance σ^2_{bckg}, which is the solution we used for our experience.

4.2.7 Simulation scenario

This section contains a short summary of every important steps for creating an image and the corresponding nuclei mask. A visualization of different settings is shown in Figure 4.7.

1. **Centroids dispersal:** Initializing a list of points corresponding to candidates for the nuclei centroids. Following the procedure described in section 4.2.2, each point is at least d_{min} away from the others.

2. **Generate nuclei mask:** Loop on the list obtained in the previous step to generate and place each nucleus. Each radius of a nucleus are taken from a Gaussian distribution with r_x and r_y mean, and $\sigma^2_{r_x}$, $\sigma^2_{r_y}$ variance. At each iteration, the maximum overlap O_{max} requirement must be respected. This process outputs the nuclei mask M_{nuc} and can be further accelerated by setting criteria that govern the total number of objects created or the maximum number of attempts to insert each nucleus.

3. **Increasing nuclei population density:** (*optional*) Calculate the number of neighbors for each nucleus created. Add a neighbor near each nucleus whose number of neighbors is below a specified threshold. This process can possibly be repeated several times to generate a very dense population.

4. **Generate membrane mask:** Reduce each polygon that allowed the creation of the core mask by a factor proportional to the thickness of the membrane. Subtract the result to obtain the membrane mask M_{mem}.

5. **Intensity maps:** Generate the Perlin noise map I_P and the light exposure map I_{light}. The intensity map is obtained on the foreground pixels only and is multiplied by a constant k_{int}, which is approximately considered as the average intensity of the nuclei.

$$I_{nuc} = k_{int} \times I_P * I_{light} * \mathbf{1}_{M_{nuc}>0} \ . \tag{4.6}$$

   Add centered Gaussian noise N_g with variance σ^2_{PSF} on background pixels:

$$I_{bckg} = N_g * \mathbf{1}_{M_{nuc}=0} \ . \tag{4.7}$$

6. **Enhancement of membranes' intensities:** (*optional*) Apply (4.6) by replacing M_{nuc} with M_{mem}, giving I_{mem}. This operation results in an image with "holey" nuclei, i.e. the pixels near the center of the nuclei are at 0. The value of these pixels is increased depending on a coefficient k_{hole} determining the intensity ratio between the membrane and the interior of the nucleus.

$$I_{int} = I_{mem} + k_{hole} \times I_{nuc} * \mathbf{1}_{(M_{nuc}>0)\cap(M_{mem}=0)} \ . \tag{4.8}$$

   Note that (4.8) can also be applied only to a certain percentage of the nuclei population to ensure some diversity within the same image.

7. **Blurring:** Apply Gaussian filtering with a kernel Γ of variance σ^2_{PSF} on the output:

$$\mathrm{I}_{blurred} = (\mathrm{I}_{int} + \mathrm{I}_{bckg}) * \Gamma \ . \tag{4.9}$$

8. **Handling saturation:** Set negative values to 0 and set to the highest value of the number of bits in the image. So, for 8-bit images, this value is 255.

4.2.8 Extension for 3D simulation

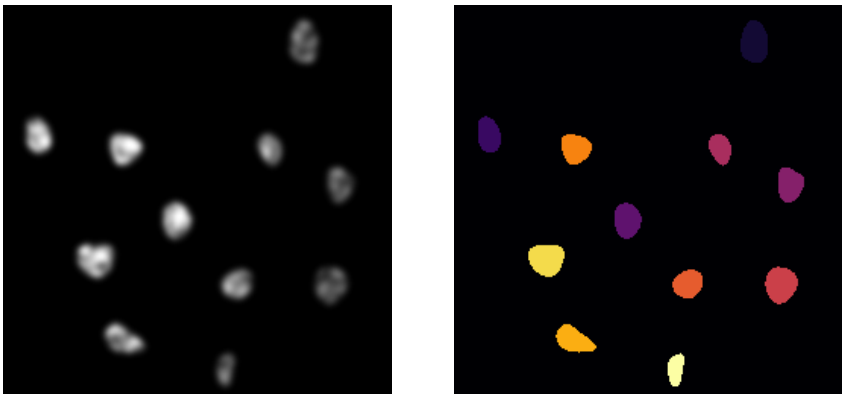
Most of the concepts presented so far are easily transferable to 3D simulation. As for the morphology of the nuclei, we have used the same shape model as that proposed by Weigert et al. [2020], namely star-convex polyhedra. Specifically, a nucleus, in its simplest form, is modeled in 3D by a set of points evenly distributed on the surface of a sphere. The coordinates of these points $(x_k, y_k, z_k)_{k=1,\dots,n_v}$ are calculated as follows:

$$\begin{aligned} z_k &= -1 + \frac{2k}{n_v - 1} \\ y_k &= \sqrt{1 - z_k^2} \times sin\left[2\pi(1 - \varphi^{-1})k\right] \\ x_k &= \sqrt{1 - z_k^2} \times cos\left[2\pi(1 - \varphi^{-1})k\right] \end{aligned}$$

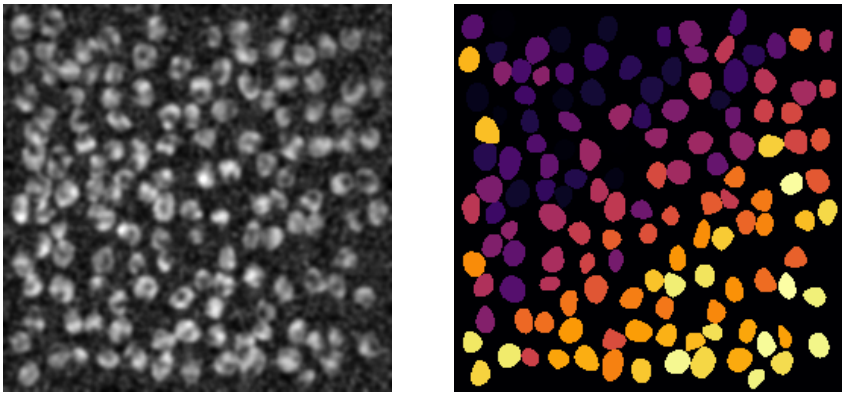
with $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The deformation model presented in section 4.2.1 remains compatible even with the addition of a third dimension and can be applied to any z coordinate. The surface of the 3D nucleus is then generated from the convex hull[2] resulting from the moved points. figure 4.8 thus shows a modeling of the shape of a 3D nucleus, obtained from the deformation of a sphere. To take into account the anisotropy of 3D data, we can apply the same process on ellipsoids of radii r_x, r_y and r_z. This 3D extension will be used in particular in chapter 5.

[2]Calculated from Qhull library. See http://www.qhull.org/

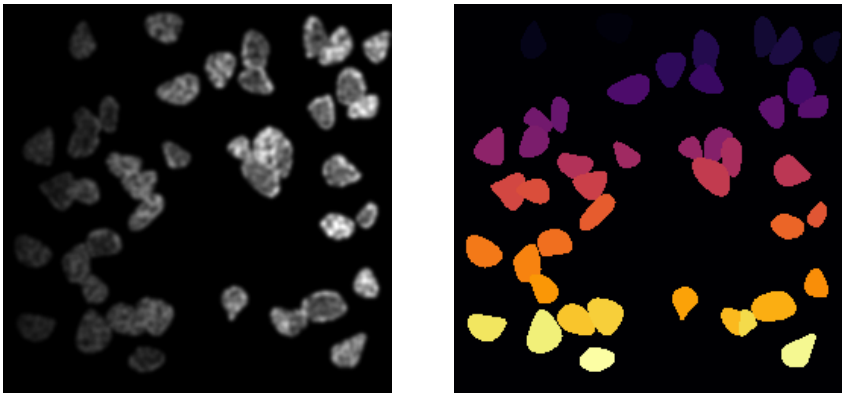
(a) Simulation 1



(b) Simulation 2



(c) Simulation 3



Simulation	r_x	r_y	d_{min}	O_{max}	Density augmented	k_{int}	k_{hole}	σ_{light}	σ_{PSF}	σ_{bckg}
Simulation 1	10	10	50	0	No	200	1	100	2	0
Simulation 2	5	5	10	0	No	100	0.5	200	1.5	10
Simulation 3	10	15	70	0.2	Yes (×2)	150	0.2	50	1	0

Figure 4.7: Visualization of 3 examples of simulated data. Left: Image, right: Ground Truth. Image dimension is 256×256. The simulation parameters used to generate this data are listed in the table above.

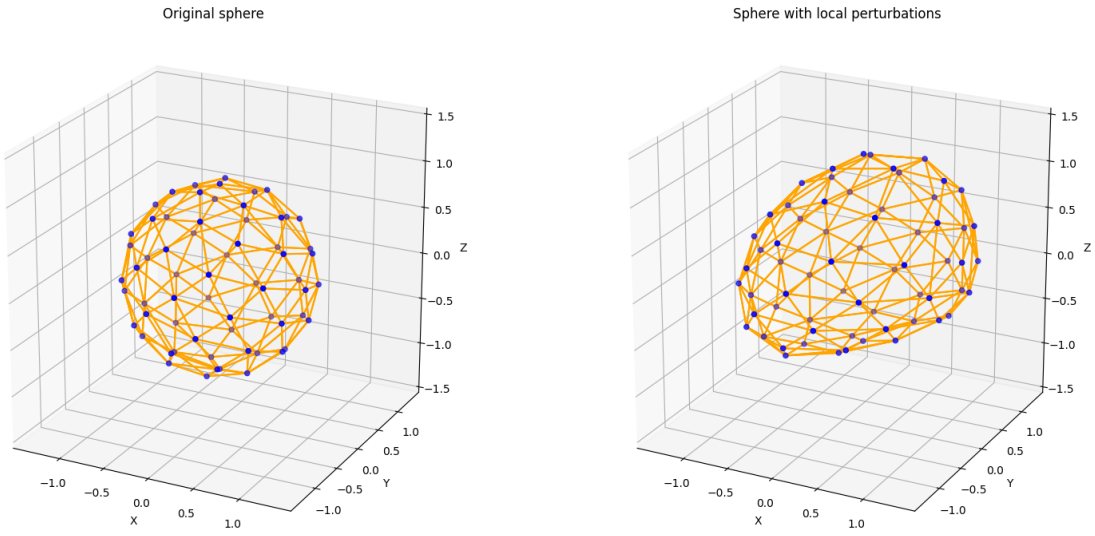


Figure 4.8: A Star Convex shape (left), with local perturbations (right)

4.2.9 Graphical interface

To simplify the use of the simulator, we have developed a graphical interface that uses the python library *PyQt*[3]. The interface can be seen on Figure 4.9 and summarizes all the parameters described in the previous sections. Thus, a user (expert or non-expert) can easily set simulation parameters using easy-to-use graphical objects (spinboxes, sliders, or checkbox). The interface also allows generating and visualizing *on the fly* a simulation example based on the parameters set by the user. The visualization includes: the simulated image, the nuclei mask, the membrane mask, and a montage showing the simulated image and the contours of nuclei mask with a colored overlay. In addition, an action button is integrated to write an entire dataset according to the current settings. We have also extended our simulator to a GPU version using the *CuPy* library[4]. This extension can be useful for generating a large amount of data quickly, and is used in particular in section 5.3.

[3] https://www.pythonguis.com/pyqt6-tutorial/
[4] https://cupy.dev/

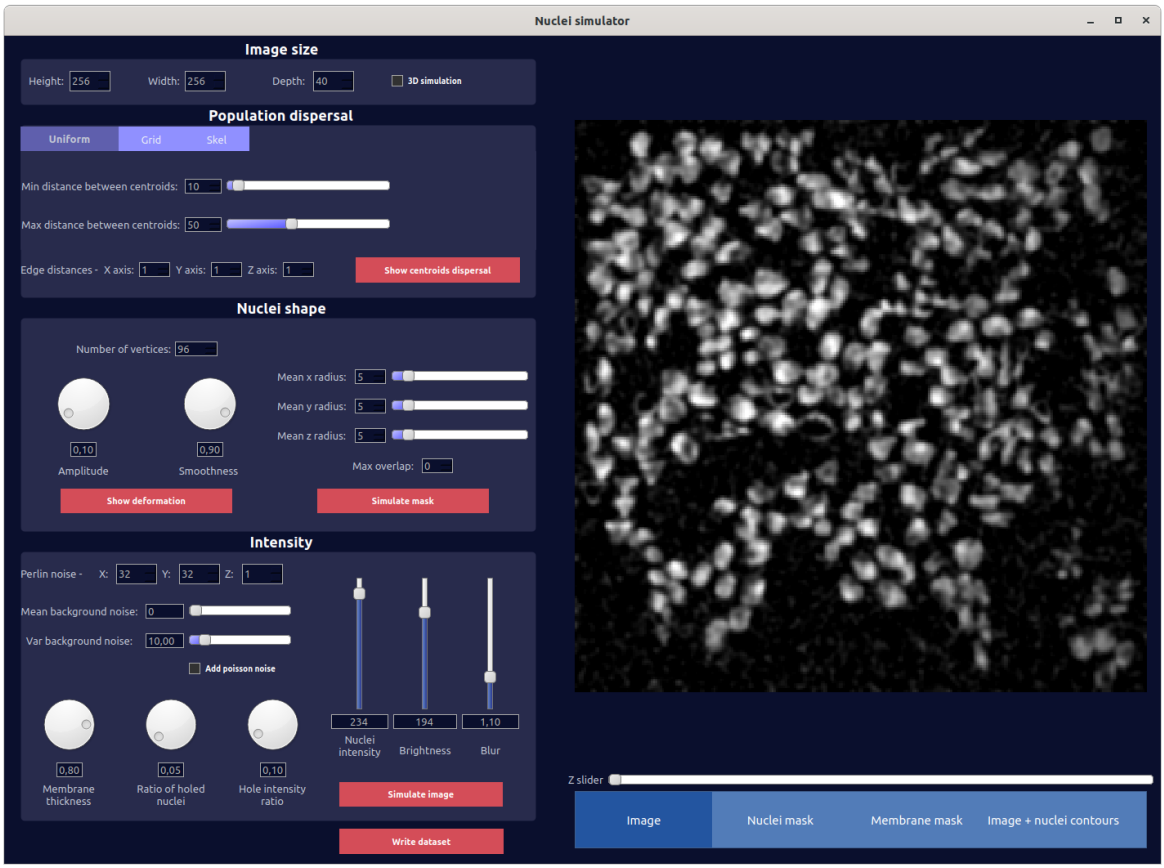


Figure 4.9: The graphical interface of our simulator. This allows to quickly see the effects of each parameter on the final rendering of the simulation.

4.3 Models for nuclei segmentation

In this section, we present two popular segmentation models derived from the U-net architecture, specifically designed for nuclei segmentation in microscopy images. These two models will be used and compared in section 4.5.

4.3.1 Stardist

Model design

The *Stardist* algorithm developed by Schmidt et al. [2018] is one of the most popular frameworks for cell detection and segmentation in microscopy and more generally in biomedical imaging [Fazeli et al., 2020; Stevens et al., 2022; Weigert and Schmidt, 2022]. The principle is to perform, for each input pixel indexed by i, j, a regression of the n radial distances $\{r_{i,j}^k\}_{k=1}^n$ separating it from the boundary of the nuclei to which it belongs. These distances form a set of equidistant radial directions around each pixel, and draws a star-shaped convex polygon from which the name of the model comes. At the same time, the algorithm also provides a probability map indicating for each pixel whether it belongs to an object (*i.e.* , a nucleus cell for our cases). It is important to note that each probability $d_{i,j}$ corresponds to the (normalized) Euclidean distance to the nearest background pixel (and not to the pixel classification based on a binary mask). Initially, only polygons whose probability $d_{i,j}$ is high enough are obtained. Then, the remaining candidate polygons are

filtered by a NMS algorithm applied with a conventional polygon clipping method.

Implementation details

Although the method described above is not specifically tied to any particular CNN architecture, a Stardist model often refers to a modified U-net algorithm. The main structure is therefore based on a contracting path followed by an expanding path (see section 2.3.5), to which a new 3×3 convolution layer with 128 filters and a ReLU activation function is added. The final two output layers are then integrated. The first is a convolutional layer with a single channel and sigmoid activation function in order to give the probability map of belonging to the nuclei. The second returns a distance map with as many channels as radial distances, thus allowing constructing a polygon for each pixel.

Training

Given the pixel-wise object probabilities and distances of the prediction $(\hat{p}, \hat{d}_k)$ and ground-truth (p, d_k), the loss function $\mathrm{L}(p, \hat{p}, d_k, \hat{d}_k)$ to be optimized during training is decomposed into a combination of two loss functions:

$$\mathrm{L}(p, \hat{p}, d_k, \hat{d}_k) = \mathrm{L}_{obj}(p, \hat{p}) + \lambda_d \mathrm{L}_{dist}(p, \hat{p}, d_k, \hat{d}_k) \ , \tag{4.10}$$

where $\mathrm{L}_{obj}(p, \hat{p})$ is an object loss reflecting how well the model classifies the foreground pixels, and $\mathrm{L}_{dist}(p, \hat{p}, d_k, \hat{d}_k)$ is a distance loss reflecting how well the boundaries of the nuclei are detected. λ_d is a constant that weights the distance loss compared to the loss object (by default $\lambda_d = 0.2$). For the object loss L_{obj}, standard binary cross-entropy is used:

$$\mathrm{L}_{obj}(p, \hat{p}) = -p \log(\hat{p}) - (1 - p) \log(1 - \hat{p}) \tag{4.11}$$

A variant of the mean absolute error is used for the distance loss L_{dist}. The goal is to increase the accuracy of the model for pixels near the center of the object (*i.e.* , theoretically for pixels with the highest probabilities). The distances are weighted by the probability of the corresponding pixel, and a regularization term $\lambda_{reg} = 10^{-4}$ is added for background pixels (*i.e.* $p = 0$):

$$\mathrm{L}(p, \hat{p}, d_k, \hat{d}_k) = p \cdot \mathbb{1}_{p>0} \cdot \frac{1}{n} \sum_k |d_k - \hat{d}_k| + \lambda_{reg} \cdot \mathbb{1}_{p=0} \frac{1}{n} \sum_k |\hat{d}_k| \ . \tag{4.12}$$

4.3.2 Cellpose

Model design

Cellpose [Stringer et al., 2021] is a deep learning segmentation model for cell and nuclear images that uses a smooth topological representation as a key component of its image processing pipeline. This topological space, called "cellpose flow", is a vector gradient flow [Xu and Prince, 1997] that directs pixels within a cell nucleus toward its center. The vectors do not necessarily point directly to the center of the nucleus, as this path could intersect the boundaries of the nucleus. Instead, the flows translate pixels to other pixels within the nucleus and across many iterations to fixed points: the centers of the nuclei. All pixels that end in the same center are grouped into regions of interest and labeled with the same ID.

On test images, the neural network predicts the horizontal and vertical gradients, which form vector fields or "paths". By following these paths, all pixels belonging to a given cell should be routed to its center. Thus, by grouping pixels that flowed to the same center point, we could simultaneously segment individual cells and recover their shapes. The cell shapes are further refined by removing pixels with cell probabilities less than 0.5.

Overall, thanks to its innovative vector flow approach, Cellpose offer improved generalization performance and the ability to accurately detect and distinguish between objects with complex shapes.

Implementation details

The CNN part of Cellpose consists of a U-net architecture that downsamples convolutional maps several times before mirror-symmetrically upsampling. In its nuclei version, it takes a single-channel 2D image as input, and performs 4 downsamplings/upsamplings using max-pooling, starting with 36 features maps up to 512 at the bottom of the network. Skip connections between upsampling and downsampling layers are done by direct summation and allows for better information flow. In addition, traditional U-net blocks have been replaced by residual blocks, each consisting of two convolutions with a filter size of 3×3, preceded by a batch normalization + ReLU operation. Finally, the last convolution layer on the upsampling path produces three output maps. The first two directly predict the horizontal and vertical gradients of the cellpose flows, and the last one is passed through a sigmoid and predicts the probability that a pixel belongs to a nucleus.

Training

Training is performed using the probability map output and the "cellpose flow" of segmented images, the latter being generated using pixel diffusion from the center of the cell nuclei and then calculating both the horizontal and vertical gradients. L_2 loss is used for gradient images and the cross-entropy loss is used for the probability map, flows are multiplied by a factor of 5 to even the relative contribution of each loss.

4.4 Segmentation metrics

To get an overview of the quality of a nuclei segmentation, an object-level metric and a pixel-level metric must be used. Pixel-level metrics evaluate the correspondence between predicted foreground pixels and ground truth, while object-level metrics evaluate the counting and classification of objects. In this section, each ground truth object is indexed by i and noted by G_i, while each segmented object, *i.e.* , resulting from a segmentation model, is indexed by j and noted by S_j.

4.4.1 F1-score

The most common object-level metric is the F1 score, defined as:

$$\mathrm{F1} = 2 \times \frac{\mathrm{P}recision \times \mathrm{R}ecall}{\mathrm{P}recision + \mathrm{R}ecall} \ , \tag{4.13}$$

with $\mathrm{P}recision = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and $\mathrm{R}ecall = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ and TP, FP and FN are the true positives, false positives, and false negatives, respectively. To compute F1 score, we need to define a criterion to determine whether an object G_i has been detected. In the rest of this chapter,

we assume that a nucleus G_i has been detected if at least 50% of its pixels are overlapped by any existing prediction S_j. Otherwise, the nucleus is classified as false negative. Finally, the false positives correspond to all the segmented objects S_j which do not overlap any ground truth objects G_i, or less than 50% of the number of pixels of G_i.

4.4.2 Jaccard Index (JI)

To evaluate segmentation quality at the pixel level, the simplest metric is the Jaccard Index (JI), another term that refers to IoU (see section 3.3.1). Given a ground-truth nucleus G_i and a segmented nucleus S_j, the JI is defined as follows:

$$JI(G_i, S_j) = \frac{|G_i \cap S_j|}{|G_i \cup S_j|} \quad . \tag{4.14}$$

In the context of nucleus segmentation, we can interpret this score as a correspondence in terms of shape between G_i and S_j. However, it should be noted that this score is computed only for true positives, and so, the presence of false positives is not penalized. The measurement of this value for a series of images is given by the mean JI (mJI), which is the average of the JI values obtained for each image.

4.4.3 Aggregated Jaccard Index (AJI)

Although the metrics presented so far provide rich information about the quality of the segmentation produced, the use of two metrics makes it difficult to compare two models in the case where each model performs better on one type of metric than the other. To unify the metrics both at pixel-level and at the object level, Kumar et al. [2017], presented the Aggregated Jaccard Index (AJI), which is an expanded version of the JI.

To compute this score, each nucleus G_i is assigned with the segmented object $S_j^*(i)$ maximizing the JI score, *i.e.*, $S_j^*(i) = \arg\max_j JI(G_i, S_j)$. Similar to the JI, the AJI calculates the ratio between the sum of the cardinal numbers of the intersection and the union of each assigned pair. To account for false positives, all segmented objects that do not match a nucleus are added to the denominator. Formally, the AJI for a ground truth with a total of N nuclei is calculated as follows:

$$AJI = \frac{\sum_{i=1}^{N} |G_i \cap S_j^*(i)|}{\sum_{i=1}^{N} |G_i \cup S_j^*(i)| + \sum_{m \in M} |S_m|} \quad , \tag{4.15}$$

with M the set of indices of segmented objects which were not assigned to a ground truth. As with mJI, the mean AJI (mAJI) is the average of this score per image.

4.5 Experiments and results

To evaluate the potential utility of our simulation model, we will use it to generate datasets that will serve as training sets for models such as Stardist and Cellpose which are designed for segmenting cell nuclei in microscopy images. To compare performance with annotated data, we will primarily work with two public datasets with different acquisition methods and varying complexity.

4.5.1 Biomedical image datasets

S-BSST265 dataset

BioStudies[5] was launched in 2015 and is a public database that aims to contribute to the reproducibility of research by linking scientific articles and data. For our study, we used one of the datasets listed in the Biostudies database under the identifier S-BSST265[6]. This dataset, provided by [Kromp et al., 2020], consists of 79 fluorescence microscopy images and includes a total of 7813 cell nuclei manually labeled by biologists. A detailed description of sample characteristics and acquisition methods is provided for each image in the form of a CSV file. In total, the images are from different biological tissues (human ganglioneuroblastoma tumor, human neuroblastoma tumor, Wilms tumor, ...) and were acquired with different modalities (fluorescence microscopy, confocal microscopy) and magnifications (20x or 63x). Moreover, the authors propose a subdivision of this dataset into training (41 images / 2727 nuclei) and test set (38 images / 5086 nuclei), which we adopt for the experience described in the next sections.

Data Science Bowl (DSB) 2018

The Broad Bioimage Benchmark Collection (BBBC) is a website[7] with a freely downloadable collection of microscopy datasets that have been used as benchmarks in over 500 studies. Each dataset is accompanied by a full description of the biological context of the analysis and a ground truth, the nature of which may vary (counts, pixel annotations, contours, bounding box, ...). For our experiments, we used the dataset with identifier BBBC038v1, which is particularly well known in the literature because it was used as the reference dataset for the Kaggle 2018 DataScience Bowl (DSB) international competition [Caicedo et al., 2019].

Actually, we only use the training data from the competition, since they are the only ones that contain segmentation for each nucleus. This results in a training set of 664 images and 28986 nuclei. Again, the images are from different organisms (humans, mice, flies) and show some heterogeneity in acquisition (different staining, magnification, illumination quality, etc.).

4.5.2 Extraction of simulation parameters from real images

To create a simulated dataset, we automatically extract the input parameters required for our simulation model. Using already labelled real images, we perform various feature extraction mainly by using *regionprops* method from python skimage library[8]. Figure 4.10 provides a visual overview of this procedure. For each real image, different simulated images can be generated from the parameters calculated as follows:

- **Size and shape**: the average radii of each nucleus are calculated from the average length of the minor axis and the major axis of each region. The eccentricity feature is also extracted to determine whether the nuclei are more circular or oval.
- **Dispersion**: the centroid of each labeled nucleus is extracted, and the average distance between all centroids is used as the minimum distance value d_{min}. The maximum overlap O_{max} is set to 0.1 for all generated images to encourage the appearance

[5] See: https://www.ebi.ac.uk/biostudies/
[6] Available here: https://identifiers.org/biostudies:S-BSST265
[7] See: https://bbbc.broadinstitute.org/image_sets
[8] See: https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.regionprops

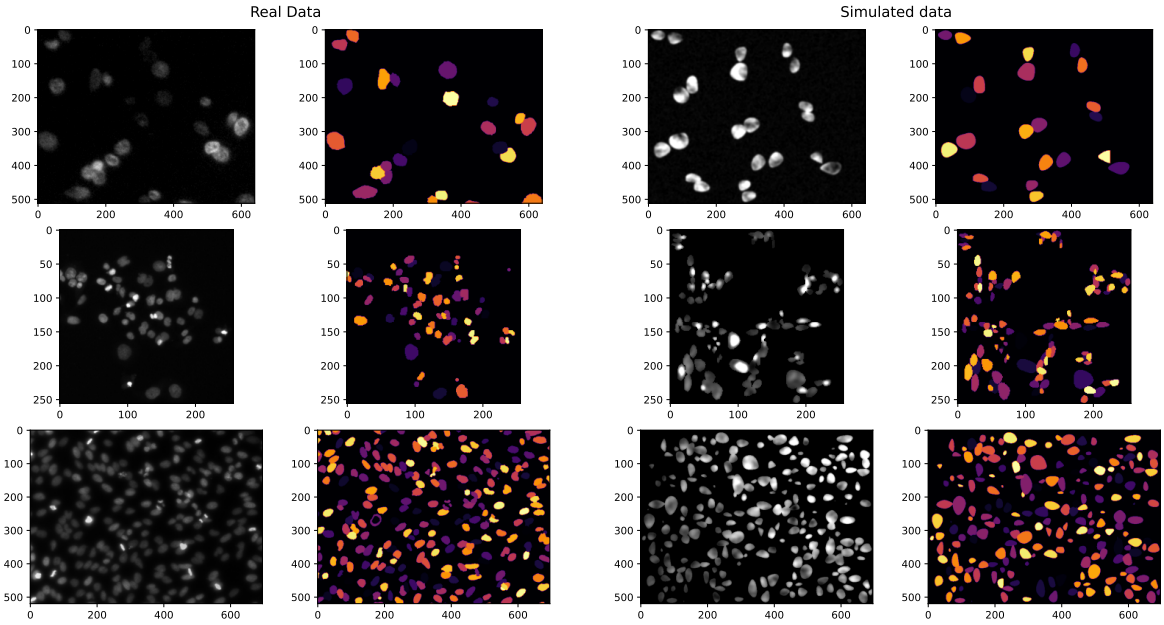


Figure 4.10: A visual comparison between the real data (left) and the simulated data (right). Each row is constructed as follows: real image/real ground truth/simulated image/simulated ground truth. Each simulated image/ground truth was obtained by automatic extraction of input parameters as described in section 4.5.2.

of clusters that are more difficult to segment. After creating an initial nuclei mask, we also calculate the average number of neighbors of each nucleus. If this value is far below that of the corresponding real image, we apply the procedure as many times as necessary to increase the population density (see section 4.2.3).

- **Intensities**: features such as mean and standard deviation of intensity can be extracted from the intensity histogram of pixels belonging to a nucleus. An equivalent method can be used to determine the mean and standard deviation background intensity. To measure the average intensities at the edges of the nuclei, we apply a mathematical erosion operation to each real mask with a disk as a structuring element. The subtraction of the real mask with the eroded mask gives a mask of the contours of all the nuclei, allowing the measurement of the average intensity of the contours and the simple calculation of the intensity ratio between the contour and the interior of the nuclei. For light exposure σ_{light}, the value is chosen randomly compared to the dimensions of the real image and ensures that few regions of the image are underexposed.

- **Blurring**: For each image, the level of blur is measured by the metric proposed by Crete et al. [2007] and serves as an approximation to obtain σ_{PSF}.

4.5.3 Training setup

In the following sections, the training of a given model is performed with the same hyperparameters, regardless of the dataset used and its size. Our choices of parameters correspond mainly to the choice offered in the authors' tutorial [9].

[9] See GitHub repositories: https://github.com/stardist/stardist and https://github.com/MouseLand/cellpose

- For the StarDist models, we use Adam optimization algorithm, initially setting the learning rate to 3×10^{-4} and gradually reducing it (*i.e.* divide by 2) when no improvement in validation loss is observed for 20 epochs.
- Cellpose models are trained with stochastic gradient descent (SGD) with a learning rate of 0.1, a momentum of 0.9, and a weight decay of 10^{-5}. The learning rate starts at 0 and anneals linearly to 0.1 over the first 10 epochs to prevent initial instabilities.
- Although trained weights are already available for each of the models, we perform a random initialization of the parameters of each model and use this initialization for each training performed. To avoid overfitting, we have also integrated an early stopping criterion for each training: if no improvement in validation loss is observed for 50 epochs, the training ends.
- For the S-BSST265 dataset, we always maintain the same separation between training and test images as suggested by the authors. For DSB 2018, we use a unique 5-fold split for the whole dataset, and we indicate the result of each experiment by averaging the result from each fold.

4.5.4 Reducing of the amount of training data

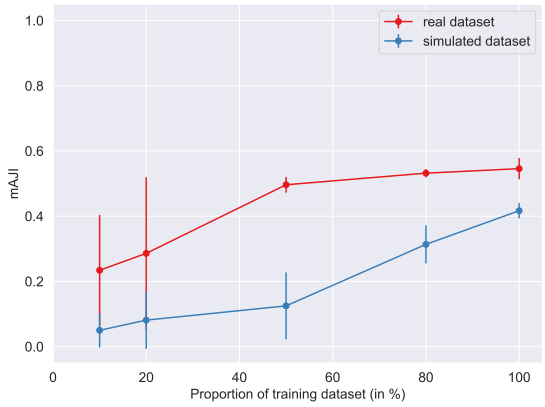
Experiment In this section, we measure the impact of reducing the amount of data on segmentation performance to provide a reference baseline. We evaluate this impact by training StarDist and Cellpose models with both a fully real dataset (*i.e.* S-BSST265 and DSB2018) and the corresponding fully simulated dataset obtained by the procedure described in the previous section. For each of the datasets studied, we take 5 different proportions of the total training dataset p_{train} size: 10%, 20%, 50%, 80%, 100%. Thus, for a given proportion, a subset of the training data set is randomly selected and actually used as training data (see table 4.1). For reasons of comparability, the unselected images are not included in the test data set, but simply ignored. We repeat this operation 5 times for each proportion to avoid any selection bias. Figure 4.11 shows the evolution of the mAJI score measured on B-SST265 and DSB 2018 for each model trained with real or simulated data of different sizes. In addition, table 4.2 shows the average scores (± stddev) obtained at the object level (F1 score) and at the pixel level (mJI).

	Number of training images in relation to p_{train} value				
	10%	20%	50%	80%	100%
S-BSST265	4	8	20	33	41
DSB 2018	53	106	265	425	531

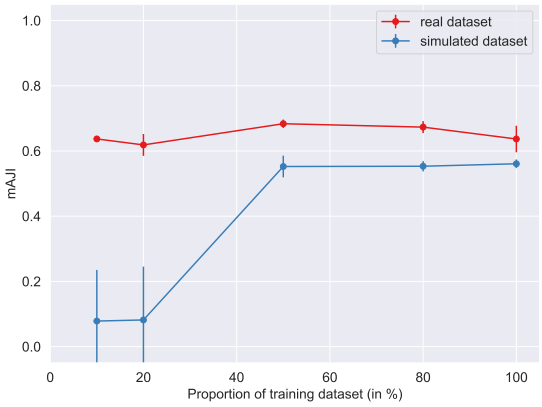
Table 4.1: Summary table of the size of the training dataset used according to the proportion p_{train} of available data for datasets S-BSST265 and DSB 2018.

Interpretations Overall, we observed that reducing the number of images to be trained significantly reduces segmentation performance, whether at the object level, pixel level, or mAJI score. This is true for both a fully simulated and a real dataset, and even more pronounced for S-BSST265, a dataset that contains only a few training images and where the use of DL approaches loses interest when the number of images used for training is further reduced. Surprisingly, this last point is not verified experimentally for Cellpose with

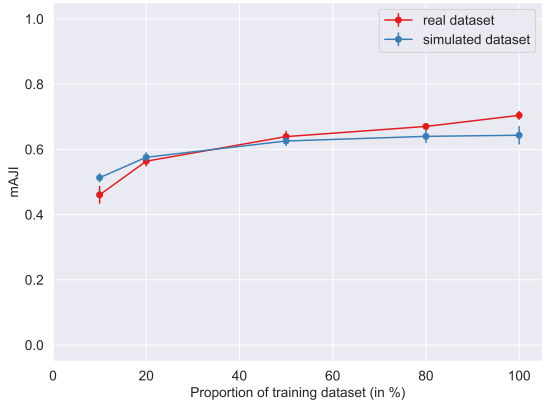
this dataset, and we therefore obtain equivalent performance on the mAJI score whether we use 10% (*i.e.* 4 real images) or 100% (*i.e.* 41 real images) of the original training data (see figure 4.11b). Nonetheless, we can infer from the other curves the need to increase the size of the training dataset to increase the performance of a given model on a fixed test set. It is also shown that, regardless of the size of the dataset chosen, using a simulated dataset as a training dataset instead of real data will almost always result in a less robust model. Therefore, in the remainder of this chapter, the baseline performance of a given model corresponds to the performance obtained when training with only real data.



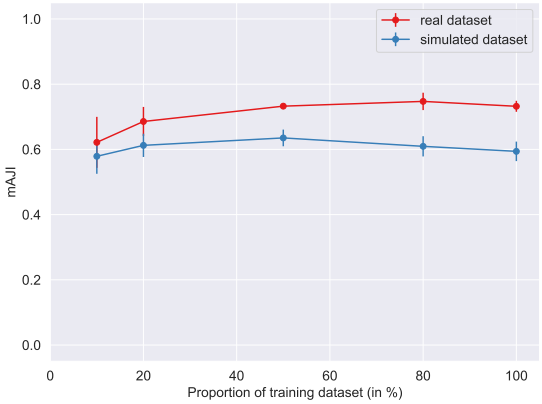
(a) Stardist performance on S-BSST265.



(b) Cellpose performance on S-BSST265.



(c) Stardist performance on DSB 2018.



(d) Cellpose performance on DSB 2018.

Figure 4.11: A comparison of the evolution of the mAJI score measured on the S-BSST265 and DSB 2018 datasets as a function of the size of the training set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. The red lines are derived from a model trained on a dataset consisting only of real images and the blue lines are derived from one consisting only of simulated images.

4.5.5 Mixing real and simulated data as a training dataset

In an effort to surpass the performance of training based on real data, we use training datasets composed of real and simulated data. More specifically, we want to investigate whether, for a fixed dataset size, the addition of simulated data can be beneficial for segmentation performance. For a certain fixed proportion p_{train} of the real data, we simulate an equivalent dataset with k_{aug} times more images in total. Thus, the variable k_{aug} corresponds to a data augmentation factor. Our experiment therefore consists in training again

Model + dataset type	Average F1 score (stddev) at p_{train} value				
	10%	20%	50%	80%	100%
Stardist + sim	0.05 (0.02)	0.07 (0.04)	0.1 (0.07)	0.35 (0.05)	0.54 (0.04)
Stardist + real	**0.08 (0.06)**	**0.22 (0.02)**	**0.44 (0.07)**	**0.55 (0.04)**	**0.58 (0.04)**
CellPose + sim	0.03 (0.01)	0.05 (0.01)	0.34 (0.05)	0.35 (0.02)	0.35 (0.01)
CellPose + real	**0.57 (0.03)**	**0.59 (0.05)**	**0.60 (0.03)**	**0.61 (0.01)**	**0.55 (0.1)**

(a) Object-level performance on S-BSST265

Model + dataset type	Average F1 score (stddev) at p_{train} value				
	10%	20%	50%	80%	100%
Stardist + sim	**0.61 (0.04)**	0.67 (0.04)	0.73 (0.04)	0.73 (0.04)	0.74 (0.04)
Stardist + real	0.57 (0.05)	**0.68 (0.04)**	**0.75 (0.05)**	**0.79 (0.02)**	**0.81 (0.01)**
CellPose + sim	0.70 (0.06)	0.74 (0.04)	0.75 (0.04)	0.76 (0.04)	0.74 (0.04)
CellPose + real	**0.75 (0.05)**	**0.80 (0.04)**	**0.83 (0.02)**	**0.85 (0.03)**	**0.84 (0.02)**

(b) Object-level performance on DSB 2018

Model + dataset type	Average mJI score (stddev) at p_{train} value				
	10%	20%	50%	80%	100%
Stardist + sim	0.05 (0.06)	0.08 (0.11)	0.15 (0.1)	0.41 (0.03)	0.50 (0.03)
Stardist + real	**0.24 (0.17)**	**0.31 (0.20)**	**0.51 (0.02)**	**0.54 (0.02)**	**0.57 (0.04)**
CellPose + sim	0.56 (0.04)	0.60 (0.04)	0.64 (0.03)	0.61 (0.02)	0.60 (0.01)
CellPose + real	**0.60 (0.06)**	**0.67 (0.03)**	**0.72 (0.01)**	**0.73 (0.02)**	**0.71 (0.01)**

(c) Pixel-level performance on S-BSST265

Model + dataset type	Average mJI scor (stddev) at p_{train} value				
	10%	20%	50%	80%	100%
Stardist + sim	**0.53 (0.02)**	**0.59 (0.01)**	**0.64 (0.01)**	0.65 (0.01)	0.66 (0.01)
Stardist + real	0.43 (0.03)	0.54 (0.01)	0.62 (0.01)	**0.66 (0.01)**	**0.68 (0.01)**
CellPose + sim	0.56 (0.04)	0.60 (0.04)	0.64 (0.03)	0.61 (0.02)	0.60 (0.01)
CellPose + real	**0.60 (0.06)**	**0.67 (0.03)**	**0.72 (0.01)**	**0.73 (0.02)**	**0.71 (0.01)**

(d) Pixel-level performance on DSB 2018

Table 4.2: Summary performance tables (at the object and pixel level) for the S-BSST265 and DSB 2018 datasets with Stardist and Cellpose models trained on real or simulated data only. The values in bold for a given model indicate the best value obtained depending on the type of dataset used.

Stardist and Cellpose models with different proportions p_{train} of the real dataset and different k_{aug} data augmentation factors based on data simulation. For the values of p_{train} we take the values from the previous experience, *i.e.* $p_{train} = 10\%, 20\%, 50\%, 80\%, 100\%$, and for augmentation factors we use $k_{aug} = 1, 5, 10$. For clarity, table 4.3 provides an overview of the composition of the training datasets as a function of the values of p_{train} and k_{aug} for the experiments performed with S-BBST265.

p_{train}	Number of real image	k_{aug}	Number of simulated image	Training dataset size
10%	4	1	4	8
		5	20	24
		10	40	44
20%	8	1	8	16
		5	40	48
		10	80	88
50%	20	1	20	40
		5	100	120
		10	200	220
80%	33	1	33	66
		5	165	198
		10	330	363
100%	41	1	41	82
		5	205	246
		10	410	451

Table 4.3: Composition of the training dataset for S-BBST265 according to the proportion of real data p_{train} and the augmentation factor k_{aug}.

In practice, for object recognition or segmentation problems, it is common to use various data augmentation methods to achieve performance improvement [Cygert and Czyżewski, 2020; Ghiasi et al., 2021; Jin et al., 2020]. In our case, we will compare ourselves to a data augmentation framework called Augmend, which is specifically designed for microscopy image analysis[10]. The implemented transformations are diverse and include: flips, 90 degree rotations, scaling, elastic deformation, intensity shifts, intensity noise.

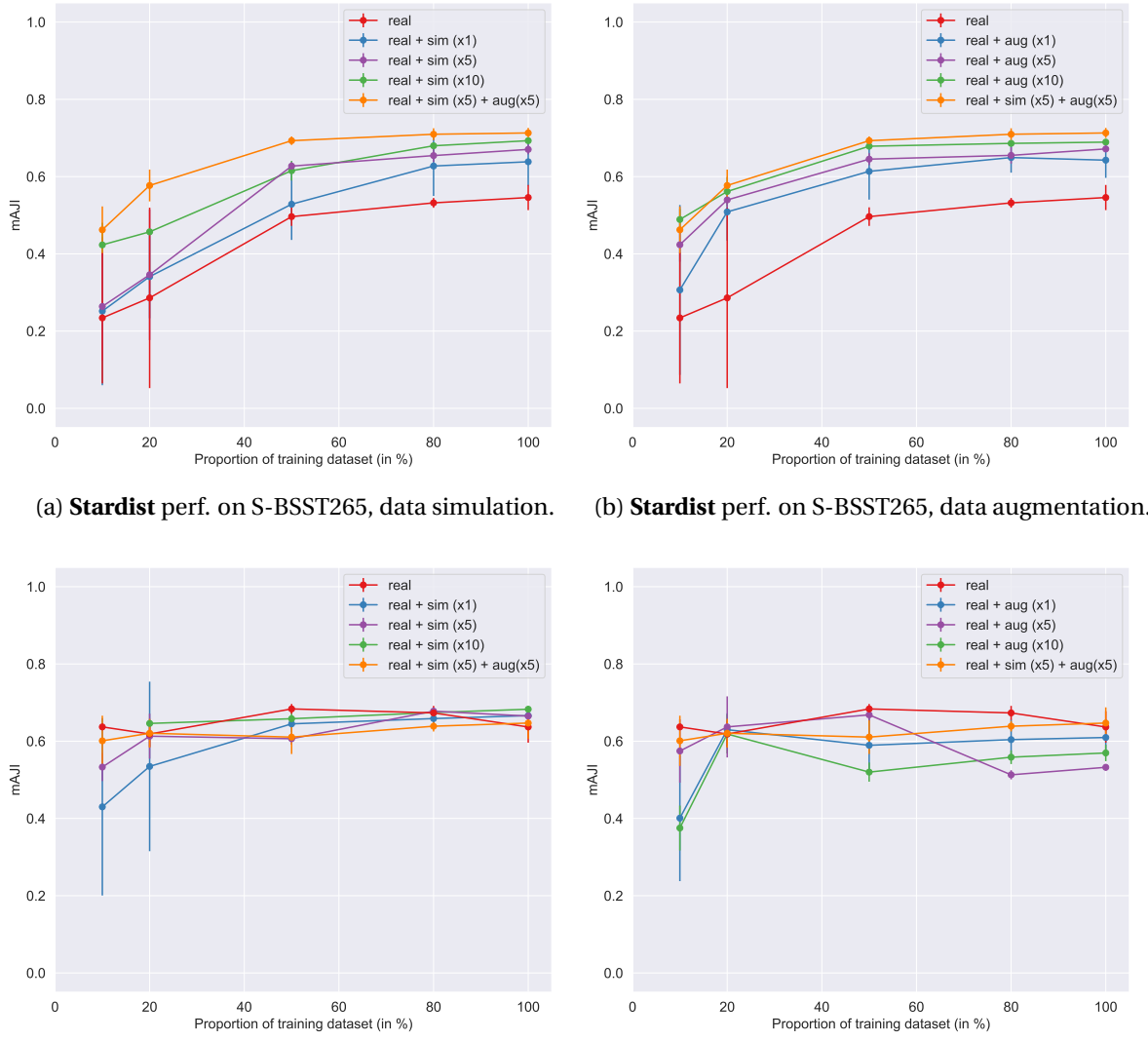
Therefore, to get a more complete comparison, we replicate the experiments with each model and each dataset, either by using a training set consisting of real data and simulated data, or by using the same real data but augmenting them with multiple random transformations. Finally, another experiment is performed by adding both simulated and augmented data. We then represent the mean mAJI values obtained in each experimental modality in figure 4.12 and in figure 4.13. From these graphs we can then draw the following conclusions:

1. Increasing the number of training images, whether performed by simulation or augmentation, is an effective way to improve segmentation performance. This is especially true for Stardist models with DSB 2018: the model trained with $p_{train} = 10$, $k = 10$ gives equivalent performance compared to the one trained using 100% of the real data only (see figure 4.13a, and figure 4.13b).

[10]Github repository: https://github.com/stardist/augmend

2. Cellpose is globally less sensitive to data augmentation, especially for the S-BSST265 dataset. For DSB 2018, we observe a slight improvement for $p_{train} = 10$ and $p_{train} = 20$. Also, neither the simulation nor the data augmentation seems to bring any performance gain.

3. The best composition for almost any configuration seems to be to include both simulated data and augmented data.



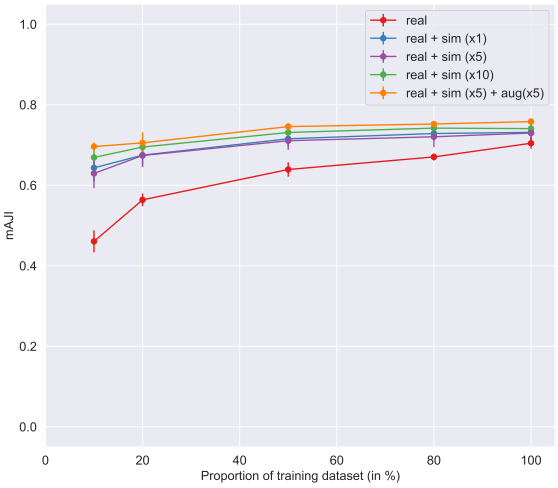
(a) **Stardist** perf. on S-BSST265, data simulation.

(b) **Stardist** perf. on S-BSST265, data augmentation.

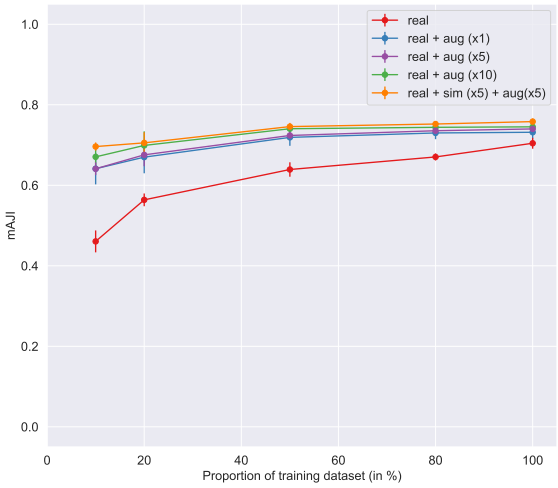
(c) **Cellpose** perf. on S-BSST265, data simulation.

(d) **Cellpose** perf. on S-BSST265, data augmentation.

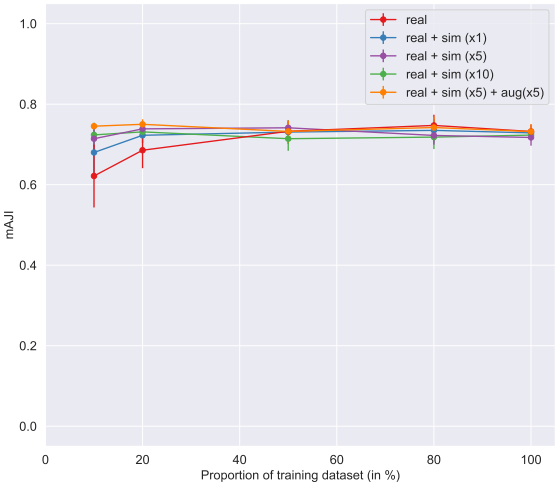
Figure 4.12: Evolution of mAJI score measured on S-BSST265 for Stardist and Cellpose, depending on the composition of the training data set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. Different data augmentation factors are applied to each proportion, either by simulation (left) or by complex transformations of real images (right). The yellow lines are from a data augmentation procedure from both simulations and data transformations, with a factor of 5 for each.



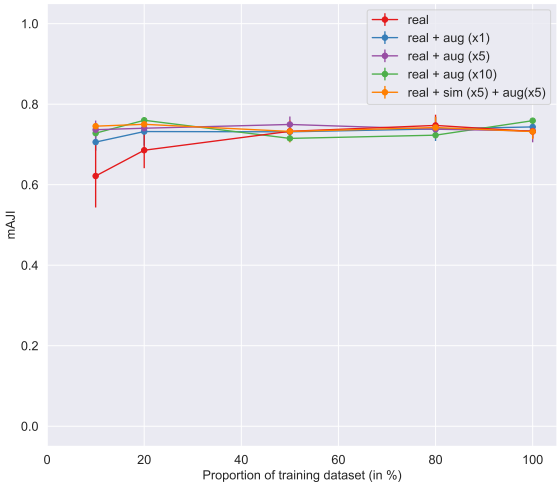
(a) **Stardist** perf. on DSB 2018, data simulation.



(b) **Stardist** perf. on DSB 2018, data augmentation.



(c) **Cellpose** perf. on DSB 2018, data simulation.



(d) **Cellpose** perf. on DSB 2018, data augmentation.

Figure 4.13: Evolution of mAJI score measured on DSB 2018 for Stardist and Cellpose, depending on the composition of the training data set. Each point corresponds to the average (± stddev) of the mAJI score measured with only a certain proportion of the available training set. Different data augmentation factors are applied to each proportion, either by simulation (left) or by complex transformations of real images (right). The yellow lines are from a data augmentation procedure from both simulations and data transformations, with a factor of 5 for each.

4.6 Conclusions

In this chapter we have presented the problem of image data acquisition and labeling when working with microscopy imaging. To address the manual labeling problem, we modeled the mechanisms involved in the formation and rendering of a microscopy image to create a data generation model. The resulting tool has interpretable and visually understandable parameters that make it compatible with a simple and easy-to-use graphical interface. In this way, we have created a tool that can generate an almost unlimited number of images that are automatically labeled with a fixed and known set of parameters.

Beyond the compelling visual aspect, we measured concretely how the use of simulated data affects the performance of DL models. This has led us to first examine the state of the art of nuclei segmentation models and then to cross-compare their performance using training sets of different sizes and compositions. Overall, we observed experimentally that performance decreases as the size of a training dataset decreases, which is a rather expected result and consistent with the reputation that DL models have. Our experiments have also shown us that the data generated by our model cannot fully replace a real dataset coming from biological studies and annotated by humans. The annotation process therefore seems to be a necessity, but its cost can at least be minimized.

The results of our experiments have thus shown that a high level of segmentation performance can be achieved from a small amount of annotated real data. The most appropriate approach in this context is then to find a way to artificially increase the size of the training data set. Simulation is a possible approach, and geometry or intensity transformations are also possible. Finally, a combination of these two approaches seems to be a method that yields the best results.

Chapter 5

Methods for quantification of ovarian cancerous tissues on 3D confocal microscopy

Abstract

Ovarian cancer is among the most lethal gynecologic cancers in women, and few studies have been performed to understand and characterize cell behavior during disease development. We present in section 5.1 a cohort of 119 patients consisting of tumor tissues and healthy tissues already imaged by confocal microscopy and visually quantified by Institut Curie. The quantification process, which consisted of counting the nuclei and centrosomes for each image, proved particularly complex given the three-dimensional nature of images. The strategy adopted in this context will be divided into two stages. In section 5.2, we will first propose a quantification method for centrosomes. In section 5.3, we use the 3D implementation of our simulation model in a loop based on a reinforcement learning principle. Therefore, we iteratively give a set of fully simulated training data that allows to get as close as possible to the count performed visually. Finally, in section 5.4, we aggregate our results for the nuclei and centrosome counts to obtain a Centrosome-Nuclei Index (CNI) quantification similar to that obtained by manual counting.

Contents

5.1 Introduction 88

5.1.1 Ovarian cancer 88

5.1.2 The centrosome 89

5.1.3 Experimental protocol 91

5.1.4 Motivations 91

5.1.5 Strategy for Centrosome-Nuclei Index (CNI) quantification 92

5.2 Methods for centrosome counting 92

5.2.1 Image processing approach 93

5.2.2 Machine Learning approach 95

5.2.3 Deep Learning approach 96

5.2.4 Evaluating centrosome counting models 97

5.2.5 Results and Interpretations 97

5.3 Learning to simulate nuclei cells 98
5.3.1 Problem statement 98
5.3.2 Reinforcement learning 99
5.3.3 Policy gradient 99
5.3.4 A strategy to optimize nuclei cells simulations 100
5.3.5 Application on 2D dataset 100
5.3.6 Application on 3D dataset 102
5.4 Quantifications of Centrosome-Nuclei Index 102
5.5 Conclusions 104

5.1 Introduction

5.1.1 Ovarian cancer

According to the National Cancer Institute (NCI), the number of new cancer cases in France in 2018 is estimated at 382,000 and the number of new cases of ovarian cancer at 5200 [Defossez et al., 2019]. Ovarian cancer remains the deadliest gynecologic cancer and the fourth leading cause of death in women [Reid et al., 2017].

It can arise on the surface of the ovary, fallopian tube, or in the abdominal cavity. Unlike other cancers, this cancer has a unique spreading system. Namely, the cancer cells lose their adherence [Iwanicki et al., 2011] and can detach from the original tumor and form secondary tumors or metastases [Mitra, 2016]. As shown in figure 5.1, 4 stages have been defined depending on the stage of tumor spread:

- Stage I: the cancer is confined to the ovaries.
- Stage II: the tumor spreads locally to the pelvic organs: uterus, fallopian tubes, bladder.
- Stage III corresponds to the extension of the tumor to the peritoneum or to the lymph nodes in the pelvis (pelvic lymph nodes).
- Finally, the cancer is classified as stage IV when it has spread in the form of distant metastases to distant organs such as the pleura (envelope of the lung) or the liver [Reid et al., 2017].

Diagnosis is usually made by physical examination of the patient (pelvic and recto-vaginal examination) and radiographs (ultrasound, scanner, MRI...) [Matulonis et al., 2016], but these are often performed late (at stage III or stage IV), as symptoms go unnoticed in the early stages of the disease. Treatment for ovarian cancer usually consists of a combination of surgery (removal of the ovaries, reduction of the number of metastases in the abdominal cavity, known as debulking) and chemotherapy. However, many patients develop resistance to treatment over time and eventually relapse.

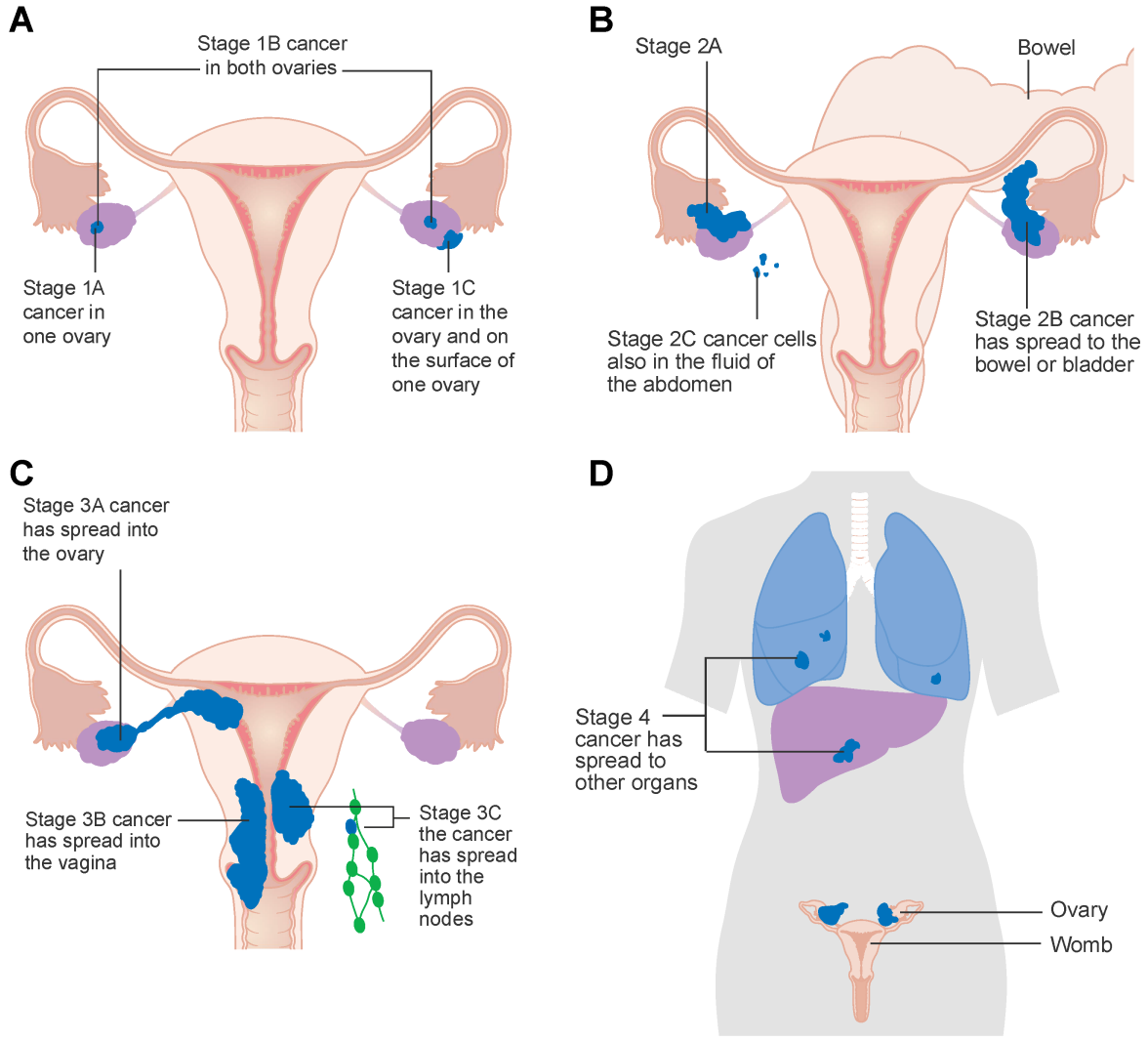


Figure 5.1: Representative scheme of the different stages of ovarian cancer. Image credit to Katopodis et al. [2019].

5.1.2 The centrosome

Centrosomes are the major centers for microtubule organization in animal cells and consist of a pair of centrioles surrounded by a matrix of pericentriolar material. Figure 5.2 depicts the important role that centrosomes play in many cellular processes, such as the organization of the poles of the mitotic spindle, which precisely divides the duplicated chromosomes into two daughter cells during mitosis [Conduit et al., 2015].

The presence of an increased number of centrosomes per cell, corresponding to *centrosome amplification*, has been frequently demonstrated in many cancers [Cosenza and Krämer, 2016; Godinho and Pellman, 2014; Nigg, 2006]. However, very few studies have been performed to quantify centrosomes in ovarian cancers [Hsu et al., 2005], and the few existing data are either not very robust (single centrosome marker, few samples) or come from model cells maintained in culture for several years, which cannot physiologically reflect the processes in a whole organism [Marteil et al., 2018].

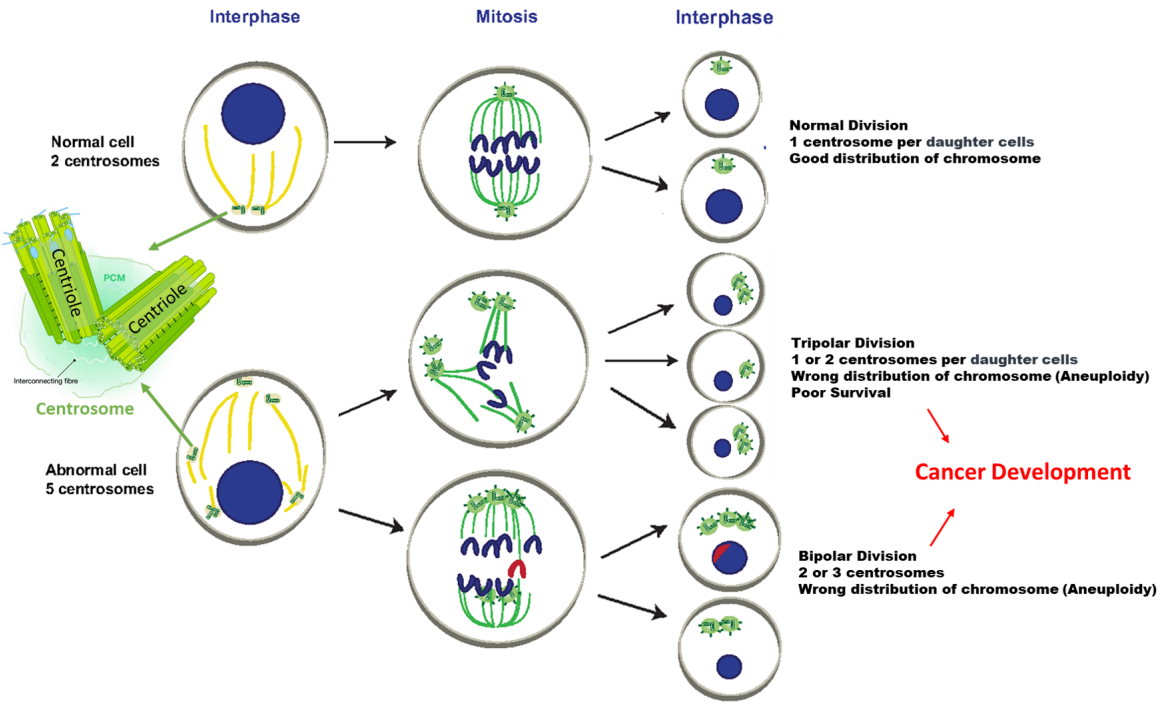


Figure 5.2: Consequences of centrosome amplification. A normal cell in interphase (top) contains two centrosomes with two centrioles (green barrels) each. After cytokinesis, two daughter cells with one centrosome each are generated. In the cells with extra centrosome (centriole duplication), two important scenarios have been described. Supernumerary centrosomes fail to cluster and form multipolar spindles that divide in a multipolar manner. This type of division is thought to have a poor outcome for cell survival. In cells where the extra centrosomes cluster to form a bipolar spindle, merotelic attachments (indicated by the red chromosome) can result in lagging chromosomes during anaphase, generating aneuploid cells. These aneuploid cells with unequal chromosome numbers can lead to several pathological conditions such as growth disorders or cancer. Figure adapted from Marthiens et al. [2012].

5.1.3 Experimental protocol

To perform a robust study that would allow characterization of centrosome abnormalities in epithelial ovarian carcinomas (EOCs), Institut Curie was able to obtain 100 human tissues from EOCs and 19 healthy tissues. Sections of these tissues were immunostained and then analyzed microscopically. To avoid false-positive counting of centrosomes, double centrosome labeling was performed using two different antibodies CDK5RAP2 and pericentrin (PCNT), and only sites that showed *colocalisation* (a concept explained in section 5.2) of the two immunolabeled proteins were quantified.

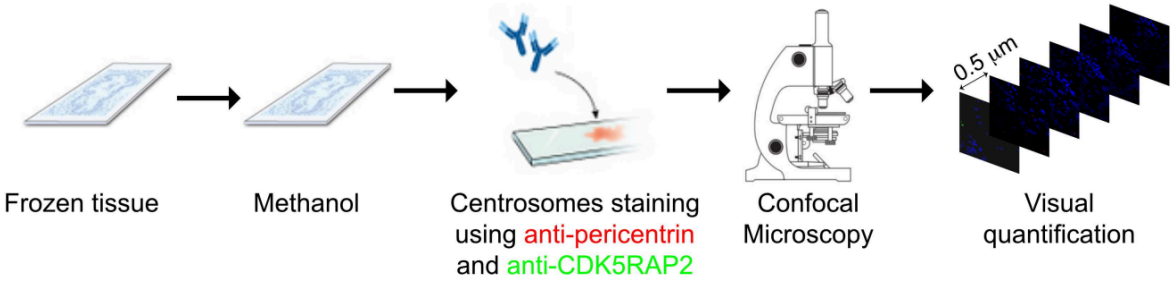


Figure 5.3: A schematic outline of the experimental protocol designed by Institut Curie in Morretton et al. [2019].

These counts were performed at a tissue thickness of 20 µm on a total of 10 randomly selected fields (*i.e.* images) per tissue and yielded an average of 5248 nuclei per patient, whereas other studies quantified only 3 fields with a thickness of 4 µm and a maximum of 500 cells [Hsu et al., 2005]. The impressive amount of data has made it possible to obtain robust results that take into account the considerable variability between different tissues, but also within the same tissue between different fields.

5.1.4 Motivations

Analysis of all these data was performed by a graduate student for two years. For each sample, the 10 selected fields were visualized and quantified using ImageJ software[1]. The number of nuclei and the number of centrosomes were counted visually, taking into account the three-dimensional nature of the images. For each sample, the Centrosome-Nucleus Index (CNI) was determined by dividing the total number of centrosomes by the total number of nuclei. Figure 5.4 shows the histogram of each quantification performed and the resulting CNI for the two tissue types (healthy and cancerous). The main findings of this study include, first, the presence of significant centrosomal amplification in part of the EOCs tissues, leading to higher CNI values than in healthy tissues. Second, high CNI values were shown to correlate with higher survival and better patient response to chemotherapy [Morretton et al., 2019].

Therefore, in order to broaden the research perspectives offered by these results, it is necessary to automate the process of ovarian tissue quantification, which has proven to be an incredibly complex task due to the nature of the tissue, and which is also very time-consuming due to the three-dimensional nature of the images.

[1]See https://imagej.nih.gov/ij/

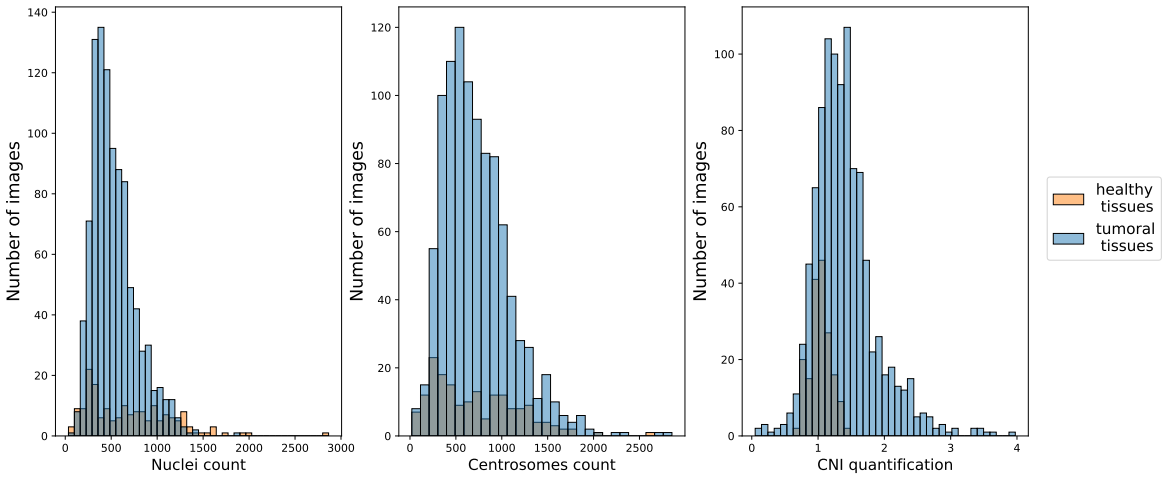


Figure 5.4: Description of the cohort quantified by Institut Curie. From left to right: counting distributions of nuclei, centrosomes and the resulting CNI for both healthy and tumoral tissues.

5.1.5 Strategy for Centrosome-Nuclei Index (CNI) quantification

The approach we propose to automatically quantify Centrosome Nucleus Index (CNI) is similar to that naturally followed by a human. We divide the problem into two independent subtasks, each leading to either the counting of centrosomes or nuclei for a given image:

- **Centrosomes counting:** we propose an initial automatic approach for colocalized spot detection similar to the visual approach. This non-learning approach will be our baseline and will then be compared to methods that incorporate Machine Learning or Deep Learning.

- **Nuclei counting:** For this problem, we propose to use the 3D version of our simulator developed in Chapter 4. In the absence of voxel-level labeling, we propose a reinforcement learning approach that consists of learning the parameters of the simulations that give the best performance on the real dataset.

5.2 Methods for centrosome counting

As far as we know, there are few, if any, fully automated methods for counting and/or detecting centrosomes. However, the analysis of colocalization in biological microscopy images is a much more described area in the scientific literature [Comeau et al., 2006; Dunn et al., 2011; Zinchuk and Grossenbacher-Zinchuk, 2009]. We will therefore address the basic principles of colocalization analysis in our approaches. In this section, we present several methods that we have developed for performing this task. We initially rely on traditional image processing methods (*i.e.* , without learning process), which we extend as we proceed by considering the specificity of the data. For comparison purposes, we integrate a method based on feature extraction and an ML model, and a second method based only on a CNN model designed for regression tasks. Finally, we propose a comparative analysis of each proposed method based on regression metrics.

5.2.1 Image processing approach

What is colocalization ?

Intermolecular interactions are at the heart of almost all biochemical mechanisms that can occur in the life of a cell. When studying these interactions, it is therefore important to unambiguously identify the molecular dynamics involved. In practice, this is done by fluorescently labeling the molecules under investigation. The cells are then placed under a microscope to obtain an image with at least two channels in different colors. The extent of interaction between two molecules is then quantified by colocalization analysis of the signals present in the final image. This means that we look for spatial correspondences, *i.e.* a certain degree of overlap, between different fluorescence signals. Figure 5.5a provides a concrete, easy-to-understand visual example of the concept of colocalization. In our study, the presence of colocalization between PCNT antibodies (red channel) and CDK5RAP2 (green channel) reflects the presence of one or more centrosomes.

Visual method

Our first method is based on basic image analysis techniques whose goals are: (1) the accurate localization of spots in the image and (2) the measurement of the overlap between two regions. We therefore propose a simple method in 5 steps:

1. **Channels segmentations:** This step is applied to each channel (red and green) separately. We start by using a Difference of Gaussian (DoG) [Young, 1987] as an enhancement process of each channel. For a given grayscale image I, the result $\Gamma_{\sigma_1^2,\sigma_2^2}(x,y,z)$ is obtained by applying the following operation:

$$\Gamma_{\sigma_1^2,\sigma_2^2}(x,y,z) = \mathrm{I} * \left(\frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2+z^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2+z^2}{2\sigma_2^2}} \right) , \tag{5.1}$$

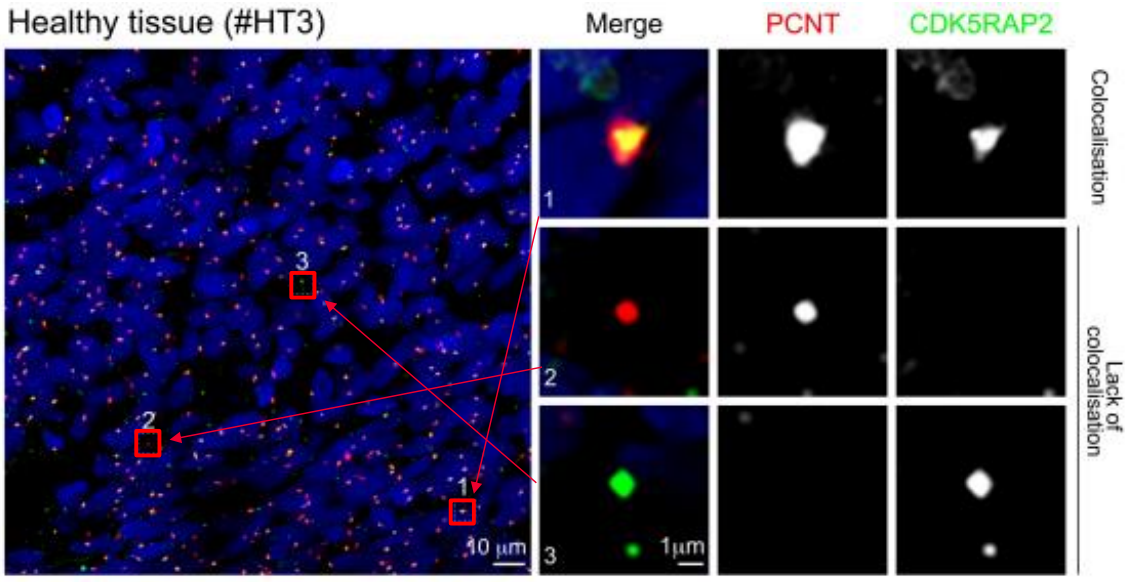
   with σ_1^2, σ_2^2 are the variance of a Gaussian blurred kernel, and $\sigma_1^2 < \sigma_2^2$ in general case. For our expererience, we set $\sigma_1^2 = 1$ and $\sigma_2^2 = 3$. Then each channel is binarized, keeping only the most intense voxels. This means that the intensity threshold is set to the $99th$ percentile of each channel image.

2. **Size filtering:** for each segmented channel, we eliminate all objects whose volume is below a certain threshold τ_{size}. Given the relatively small volume of each object whose diameter is close to the resolution of the voxels, we set this threshold to a low value, *i.e.* , $\tau_{size} = 15$.

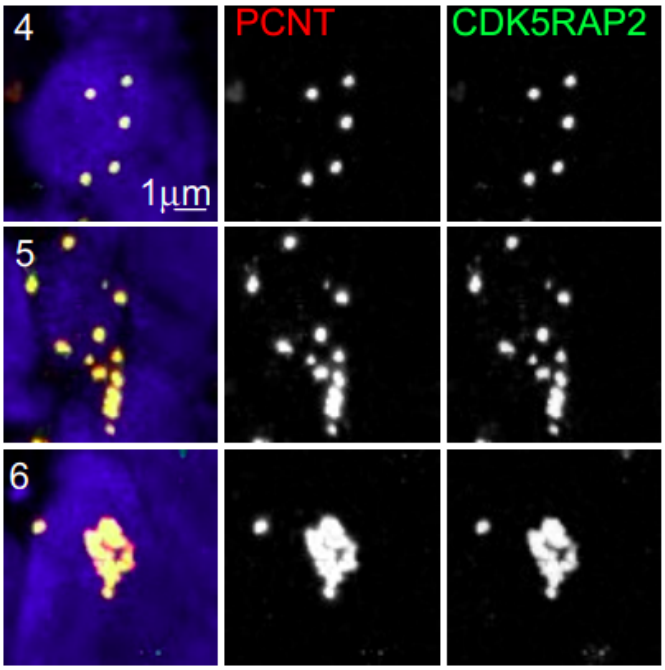
3. **Colocalization filtering:** in this procedure, the segmented regions of one channel are filtered based on their degree of overlap with the regions of the other channel. Thus, for a given region, it is assumed that the minimum overlap must be at least 10% of all voxels belonging to that region.

4. **Intersection of channels:** In this step, the two processed channels are merged by taking their intersection, *i.e.* , we obtain a binarized mask in which each positive pixel has "survived" the first three stages in both the red and green channels.

5. **Connected component algorithm:** this last step allows to label each detected region and thus to perform a centrosome count.



(a) Visualization of colocalized spot (centrosome) and non-colocalized spots on healthy tissue (HT).



(b) Visualization of different arrangements of centrosomes (4-isolated, 5-clustered, 6-super-clustered) in tumor tissues (TT)

Figure 5.5: Overview of the task of quantifying centrosomes. Automated methods must capable of measuring a sufficient level of colocalization (a) and, for large, inseparable aggregates, providing an approximation of the count (b). Figures adapted from [Morretton et al., 2022].

Watershed-based method

The method described above, while representative of the Institut Curie visual approach, is very sensitive to artefacts present in one channel or the other and does not allow local separation. As an alternative, we propose another method to the first one:

1. **Merging:** the red channel and the green channel from the first step are merged into a single image. The result of the merging is an image I_{merge}, which is calculated as follows:

$$I_{merge} = \min(R, G) \tag{5.2}$$

   where R and G correspond to the red and green channels, respectively. Thus, the high intensities of I_{merge} correspond to the most colocalized regions of an image. We also apply a DoG operation and intensity thresholding as in the previous method to get a binarized mask of I_{merge}.

2. **Local maxima and watershed:** A distance map from the nearest background voxel is computed, and the local maxima location of this map is given as initialization of a watershed algorithm.

3. **Aggregating regions:** To solve the problem of over-segmentation, which could also lead to overcounting of centrosomes, we propose an aggregation rule inspired by the NMS algorithm: we order the detections by their average intensity and group together those between which IoU score is greater than 0.1.

5.2.2 Machine Learning approach

In some samples, the presence of "clusters" and "superclusters" was highlighted (see figure 5.5b). This clustering of multiple centrosomes close to each other made visual counting difficult or even impossible, so the number of centrosomes in these clusters was estimated from the surface area/size of a normal centrosome. To overcome this issue, we decide to perform feature extraction based on the segmentation obtained with the previous method. Thus, for each image, we extract relevant features related to the counting procedure performed:

- **Counting feature:** is the centrosome count obtained by watershed-based method (1 feature).

- **Intensities features:** (2×7 features) are obtained by analyzing the intensity histogram of all detections in both channels. Then we take min, max, mean, standard deviation, median, 1st and 3rd quartile of the intensity distribution.

- **Volume features:** (8 features) are obtained by analyzing the volume histogram of all detections in the same image. Then we take min, max, mean, standard deviation, median, 1st, and 3rd quartile of the volume distribution. Finally, we also add the ratio of the total sum volume of the detections to the total volume of the image.

- **Colocalisation features:** (4 features): Following the recommendations on Dunn et al. [2011], we have extracted the main statistics needed to measure colocalization. Given

an image consisting of red channel R and green channel G, the Pearson's correlation coefficient (PCC) is defined as:

$$\text{PCC} = \frac{\sum_i (\text{R}_i - \overline{\text{R}}) \times (\text{G}_i - \overline{\text{G}})}{\sqrt{\sum_i (\text{R}_i - \overline{\text{R}})^2 \times (\text{G}_i - \overline{\text{G}})^2}} \ ,$$

where i is an index over pixels, and $\overline{\text{R}}, \overline{\text{G}}$ refers to the mean intensity of red and green channel respectively. This coefficient takes values between -1 and 1 and reflects how closely the intensity values of red and green are correlated. Therefore, a PCC coefficient with a value close to 0 indicates that the distribution of the red signal and the green signal are uncorrelated. Although it is an effective statistic for measuring colocalization, the main shortcoming of PCC is that it poorly measures the amount of signal that colocalizes.To overcome this problem we also use the Manders Overlap Coefficients (MOC) [Manders et al., 1993] defined as:

$$\text{MOC} = \frac{\sum_i (\text{R}_i \times \text{G}_i)}{\sqrt{\sum_i \text{R}_i^2 \times \sum_i \text{G}_i^2}} \ .$$

This quantity is also often accompanied by the coefficients M_1 and M_2, called Manders Colocalization Coefficients (MCC), defined as follows:

$$\text{M}_1 = \frac{\sum_i \text{R}_{i,colocal}}{\sum_i \text{R}_i}$$
$$\text{M}_2 = \frac{\sum_i \text{G}_{i,colocal}}{\sum_i \text{G}_i} \ ,$$

where $\text{R}_{i,colocal} = \text{R}_i$ if $\text{G}_i > 0$; $\text{R}_{i,colocal} = 0$ if $\text{G}_i = 0$; $\text{G}_{i,colocal} = \text{G}_i$ if $\text{R}_i > 0$ and $\text{G}_{i,colocal} = 0$ if $\text{R}_i = 0$.

After all these features were extracted, we used the Regression Learner application[2] available in Matlab 2020a version to find out which regression model has the best performance with this data. The procedure consists in entering this data and the expected response into the application and launching several model trainings of different types (linear regression, regression trees, SVM, random forest, ...). Finally, we observed that the Gaussian Process Regression (GPR) models [Seeger, 2004] perform best on this data (see section 5.2.4 for more details on how to measure performances).

5.2.3 Deep Learning approach

As a final solution to counting centrosomes, we propose to perform this regression task via a DL model. In contrast to the non-training methods presented previously, it is very difficult to process images directly in 3D and in full resolution in this context. The tiling process also seems rather inappropriate here since we don't have a per-tile ground truth count. Finally, to conserve GPU memory, we convert our three-dimensional dataset into a two-dimensional dataset by applying a maximum intensity projection in the depth direction (*i.e.* , the z-axis) to each image. The principle is to construct a 2D image whose elements correspond to the pixels with the highest intensity among all slices of the 3D image.

Therefore, we use this dataset as a training set to count centrosomes. We use a ResNet-18 model as the regression model, which we run for 100 epochs with a batch size of 5 images and a MAE loss (see next section) with an ADAM optimization algorithm and a constant learning rate initialized at 10^{-3}.

[2]See: https://fr.mathworks.com/help/stats/regressionlearner-app.html

5.2.4 Evaluating centrosome counting models

To assess the performance of our centrosome counting methods, we use several metrics that are classically used for regression tasks. For this section, we note y_i the number of centrosome obtained by manual counting on the i-th image (target value), and $\hat{y}_i$ the count obtained by any automatic method (predicted value).

The Root Mean Squared Error (RMSE) is the most commonly used metric for regression tasks. As the name suggests, this metric is the square root of the averaged squared differences between target and predicted values. RMSE is particularly useful for penalizing large estimation errors and is calculated as follows;

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \quad .$$

Mean Absolute Error (MAE) corresponds to the average absolute value difference between the target value and the predicted value. Because of its linearity, it has the advantage of being easily interpreted:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \quad .$$

Similar to MAE, Mean Average Percentage Error (MAPE) correponds to the mean of all absolute percentage errors between the predicted and actual values:

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad .$$

Finally, we also use the Pearson correlation coefficient, r, which is a specific measure that quantifies the strength of the linear relationship between two variables:

$$r = \frac{Cov(\hat{\text{Y}}, \text{Y})}{\sigma_{\hat{\text{Y}}} \sigma_{\text{Y}}} = \frac{\sum_{i=1}^{n} (\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (\hat{y}_i - \bar{\hat{y}})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}} \quad .$$

The coefficient varies between -1 and 1, and the further away r is from zero, the stronger the linear relationship between the two variables.

5.2.5 Results and Interpretations

Finally, we simply applied each of the methods to all images belonging to the cohort. Note, however, that the methods involving a learning process, i.e. the methods with ML and DL, were trained and tested according to a 5-fold cross-validation applied according to the patients (rather than looking at all the images independently of each other). Thus, the performance indicated for these models corresponds to an average of the results obtained in each fold. All counting scores are reported in section 5.2.5 and a scatter plot is available for each model in figure A.3 and figure A.4

Based on these results, our main interpretation is that healthy and cancerous tissue do not necessarily need to be quantified using the same method. In fact, methods based on image analysis (visual approach and watershed-based method) on healthy tissue are equivalent or even better than learning methods. However, it is clear that learning-based

Method	Healthy tissues				Tumoral tissues			
	RMSE	MAE	MAPE	r	RMSE	MAE	MAPE	r
Visual method	185.73	114.66	**19.70**	0.93	359.2	294.94	39.65	0.86
Watershed method	**145.72**	**102.73**	24.24	**0.96**	280.36	226.24	30.56	0.90
ML method	196.67	136.78	30.07	0.93	148.03	107.46	16.38	0.91
DL method	157.26	108.38	21.43	**0.96**	**146.89**	**95.18**	**13.47**	**0.93**

Table 5.1: Summary table of counting results for each of the methods used to quantify centrosomes on healthy tissue and tumor tissue.

methods give much better results in cancerous tissues on all scores: RMSE, MAE, MAPE and correlation. This is mainly explained by the presence of clusters and superclusters in cancerous tissues, which are particularly difficult to segment given the size of the centrosomes in the images. These methods will then tend to undercount, while the features extracted and processed by the methods with learning can help adjust the count.

5.3 Learning to simulate nuclei cells

Our approach to counting nuclei is essentially based on using our simulator developed in the previous chapter. Compared to Chapter 4, our main problem here is that we do not have pixel-level annotations. Therefore, in this section we implicitly assume that good nuclei segmentation in 3D inevitably leads to good nuclei counting. On the other hand, we cannot apply a feature extraction method as explained in section 4.5.2 and the very large heterogeneity of the data does not allow us to initiate a suitable setting based on a simple visual approach. Our goal in this section is therefore to find a setting for our simulation model that makes the count as close as possible to that of the manual counting.

5.3.1 Problem statement

To automatically find the best possible parameterization, the modeling of our problem is similar to that proposed by Ruiz et al. [2018] in a context of counting cars in traffic's scenes. Formally, we want to simulate a real-life dataset $\mathscr{D}_{sim}$ such that a nuclei segmentation model h_θ ensures high performance on real dataset $\mathscr{D}_{real}$. Our simulation model can then be viewed as a generative model $G(I,M|\psi)$ which allows to provide a set of image-mask pairs (I,M) governed by a set of parameters ψ. Therefore, we want to solve the following bi-level optimization problem:

$$\psi^* = \arg\min_{\psi} \sum_{I,M \in \mathscr{D}_{real}} \mathscr{L}(M, h_\theta(I;\theta^*(\psi)) \tag{5.3}$$

$$s.t. \quad \theta^*(\psi) = \arg\min_{\theta} \sum_{I,M \in \mathscr{D}_{sim}} \mathscr{L}(M, h_\theta(I,\theta)) \ , \tag{5.4}$$

with:

- ψ^* are the optimal simulation parameters.
- $\theta^*(\psi)$: are the nuclei segmentation model parameters obtained after a training with simulated data parameterized by ψ.
- $\sum_{I,M \in \mathscr{D}_{sim}} \mathscr{L}(M, h_\theta(I,\theta))$ denotes the loss function evaluated on $\mathscr{D}_{sim}$ with a model h_θ parametrized by θ.

- $\sum_{I,M \in \mathscr{D}_{real}} \mathscr{L}(M, h_\theta(I;\theta^*(\psi))$ refers to the loss function evaluated on $\mathscr{D}_{real}$ with a model $h_\theta(I;\theta^*(\psi))$ trained on $\mathscr{D}_{sim}$.

5.3.2 Reinforcement learning

To better understand the experiments we performed, we briefly introduce some notions of Reinforcement Learning (RL), which, like DL, is a sub-branch of Machine Learning. RL is the study of interactions between an *agent* an its environment. The agent chooses to take one action $a \in \mathscr{A}$ leading to be in state $s \in \mathscr{S}$ according to transition probabilities (P). For each action a, the environment provides a reward $r \in \mathscr{R}$ as feedback. Figure 5.6 illustrates this process simply.

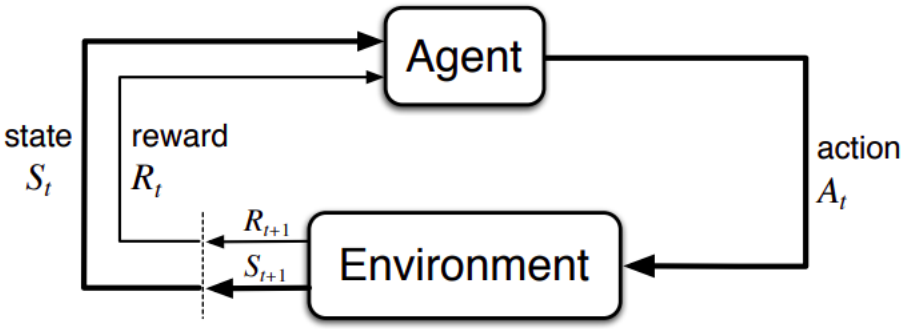


Figure 5.6: Interaction diagram between an agent and its environment in RL paradigm. Image credit to Shweta Bhatt's article.

More specifically, actions are performed according to a *policy* π that refers to the agent's behavioral strategy. This strategy is based on the observation of the state s, and the nature of the agent's policy can be:

- deterministic: $\pi(s) = a$
- stochastic: $\pi(a|s) = \mathbb{P}_\pi(A = a|S = s)$. If the policy is parameterized, we denote $\pi_\omega(.)$ a policy parameterized by ω.

Therefore, the goal of methods based on RL is to find the optimal strategy that the agent must follow in order to maximize the obtained reward.

5.3.3 Policy gradient

Policy Gradient Methods (PG) are a type of RL techniques based on the optimization of parameterized policies to achieve the optimal reward. In general, the policy is modeled with a function $\pi_\omega(a|s)$ parameterized by ω, whose variations influence the reward value. Thus, the optimization can be done with a gradient descent algorithm.

Mathematically, the goal of PG methods is to maximize the expected total reward $J(\omega)$, defined as:

$$J(\omega) = \mathbb{E}_{\psi \sim \pi_\omega}\left[R(\psi)\right] \ ,$$

where $R(\psi)$ is the total reward obtained by training a model with data parameterized by ψ. A standard approach to solve this optimization problem is to use gradient descent

(or ascent) algorithm. Thus, at each iteration t, the policy parameters ω_t are updated as follows:

$$\omega_{t+1} = \omega_t + \alpha \times \nabla_\omega J(\omega) \ ,$$

where α is the learning rate. Following the REINFORCE rule Williams [1992], the gradient can be written as:

$$\nabla_\omega J(\omega) = E_{\psi \sim \pi_\omega} \left[\nabla_\omega log(\pi_\omega) R(\psi) \right] \ .$$

An unbiased approximation of this quantity is:

$$\nabla_\omega J(\omega) = \nabla_\omega \log(\pi_\omega) \hat{A}(\psi) \ ,$$

where $\hat{A}(\psi) = R(\psi) - b$ is the reward advantage estimate and b a baseline, set to an exponential moving average.

5.3.4 A strategy to optimize nuclei cells simulations

We propose to follow a PG method to simulate data that allows a Stardist model to get as close as possible to the visual approach in terms of nuclei counts. Algorithm 5 presents our approach in a simplified form. Since the space of actions is continuous, we set the simulation policy to follow a Gaussian distribution parameterized by ω, *i.e.* , $\pi_\omega = \mathcal{N}(\mu, \sigma^2 I_n)$. Thus ω is a concatenation of n means $\{\mu_i\}_{i=1,2,...,n}$ and n standard deviations $\{\sigma_i\}_{i=1,2,...,n}$, where n is the total number of parameters in our simulator. For each μ_i and each σ_i, the derivatives are:

$$\nabla_\mu \log(\pi_\omega) = \frac{\psi_i - \mu_i}{\sigma_i^2} \tag{5.5}$$

$$\nabla_\sigma \log(\pi_\omega) = \frac{(\psi_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3} \ . \tag{5.6}$$

For the sake of simplicity, we will not try to optimize the standard deviations, so we set $\forall i, \sigma_i = 0.05$.

Algorithm 5: Learning to simulate nuclei cells

Input : A simulation model and parameters initialization, a nuclei segmentation model and corresponding weights θ,
for *iteration* $i = 1, 2, ..., n_{episode}$ **do**
  Use policy π_ω to generate a vector of simulation parameters ψ
  Generate a dataset D_{sim} of size M, following ψ parameterization
  Train Stardist for ξ epochs, initialized by θ
  Test on $\mathscr{D}_{real}$, and obtain reward $R(\psi)$
  Compute the advantage estimate $A(\psi) = R(\psi) - b$
  Update the policy parameters ω.
Output : A generator G with an optimized policy ψ_ω

5.3.5 Application on 2D dataset

In this section, we tackle the previously described problem with a 2D public dataset, which is simpler and less computationally expensive. The results obtained will guide us in our strategic choices for applying to ovarian tissue data.

BBBC002-v1 dataset

We used the image set BBBC002v1 Carpenter et al. [2006] from the Broad Bioimage Benchmark Collection (BBBC) Ljosa et al. [2012]. This dataset consists of 5 samples from Drosophila melanogaster. Each sample consists of 10 fields, resulting in a total of 50 512×512 grayscale imaes. As a baseline, a table is given summarizing a count of the number of nuclei in each image. These counts were made by two different experts, and we always compare models outputs to the average of the two experts counts.

Training setup

For all experiments, we generate 50 simulation images that are used as a training set for a StarDist model (2D, then 3D). Stardist models are trained for 200 epochs with a batch size of 10, using an ADAM optimizer and an initialized learning rate)10^{-3} that decreases by a factor of 5 every 40 epochs.

Experiences and results

We are optimizing here our simulator to maximize the negative reward function MAE on the BBBC002-v1 dataset. The first experiment consists in studying the influence of the initialization of the simulation parameters on the convergence speed of the PG algorithm. We therefore take three random simulator settings for the first iteration of the PG loop and then plot each reward trajectory in figure 5.7. We can observe that the convergence speed of the algorithm is not the same depending on the initialization of the sampled simulation parameters. After observing the simulation data generated for each initialization, it turns out that the one that gives the green curve starts simulating very noisy data, with a nuclei size that is far too large compared to the average size of nuclei present in the real dataset. This could explain the delay in convergence. For the second experiment, we replaced the updating rule of policy parameters. Due to the simple implementation, our optimization choice fell on ADAM (see section 2.2.4 for details). We can then also observe in figure 5.7 that, for the same initialization parameters, the ADAM optimization algorithm enables much faster convergence and achieves globally higher reward values compared to vanilla gradient descent algorithm.

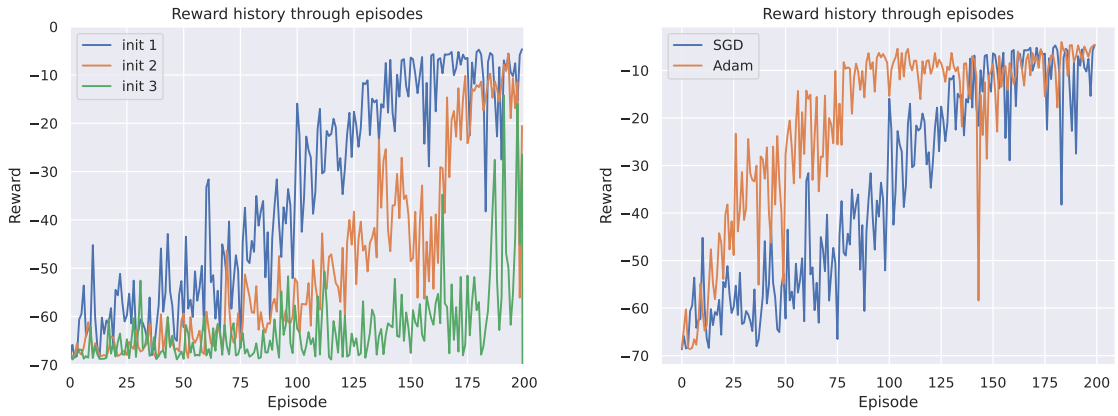


Figure 5.7: An overview of the reward curves for all experiments performed on the BBBC002-v1 dataset. Left: reward curves for different initialization of simulation parameters. Right: comparison of reward curves based on policy parameter optimization algorithm (SGD vs. Adam).

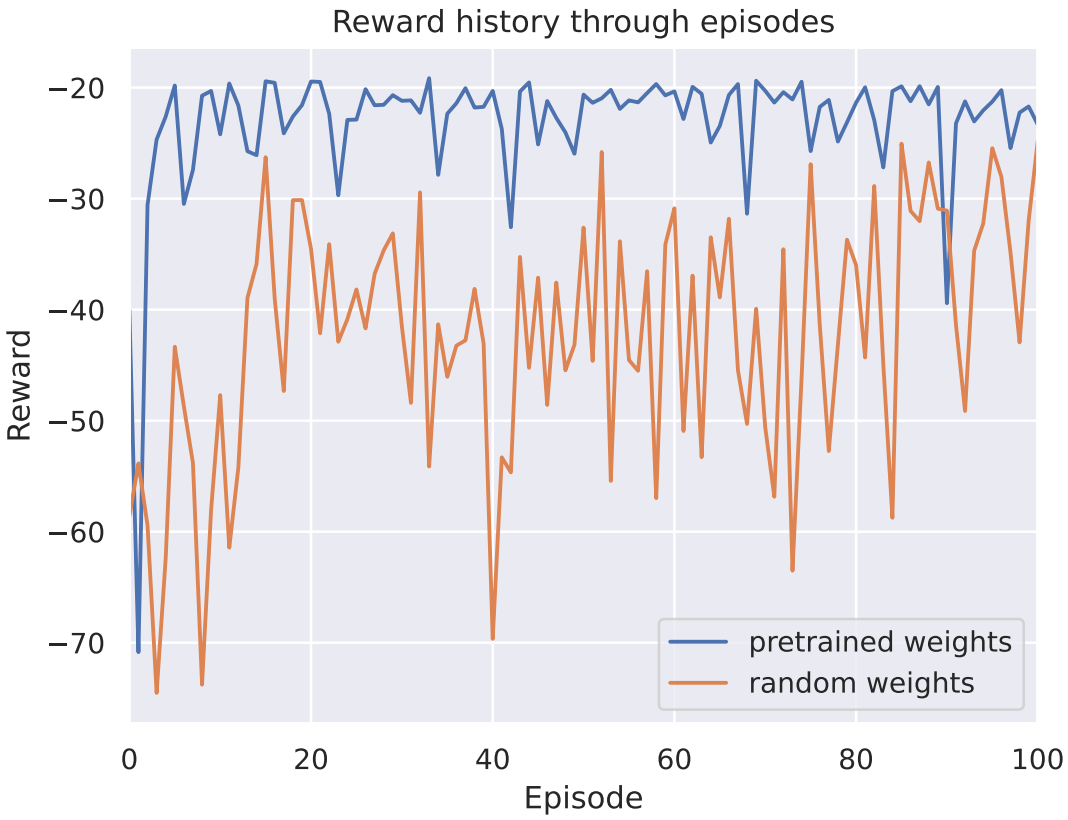


Figure 5.8: Comparison of reward curves using a pre-trained or untrained model.

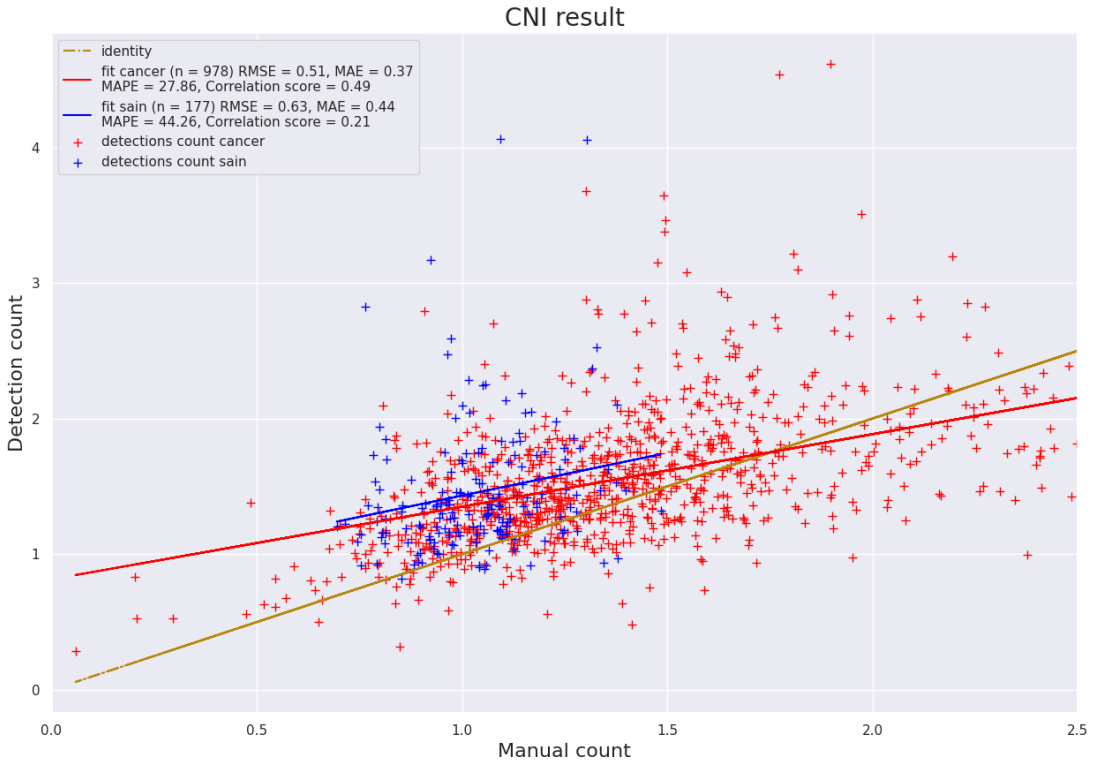
5.3.6 Application on 3D dataset

Finally, we extend the experience to a 3D application to ovarian tissue. Due to the greater heterogeneity of the data, the goal this time is to maximize the negative reward function MAPE, using Adam updates for policy parameters. Based on our observations from the previous 2D experience, we initialize the simulation parameters "by hand" to values that do not result in degenerate images (*i.e.* , noise level higher than foreground signal, average radii of the mean values higher than the dimensions of the images, exposure with zero light, highly blurred, etc.). In contrast to the 2D experiment, we also compare the performance that can be obtained with an h_θ model whose θ weights have already been trained in advance with any other dataset. The curves for each of the two experiments are shown in figure 5.8. We can thus observe that the pre-trained model seems to obtain much more stable rewards than the model initialized with random weights. In addition, the "plateau" reached is much higher ($R \approx -20$).

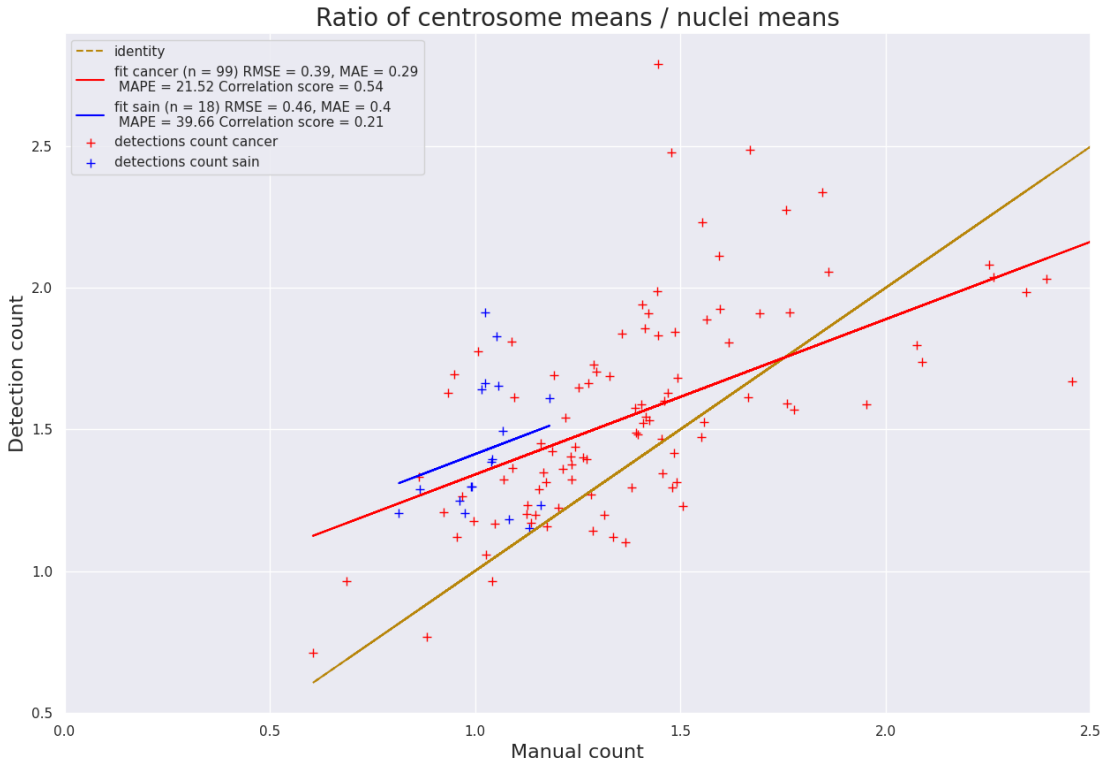
5.4 Quantifications of Centrosome-Nuclei Index

In this final part, we summarize the best results from the previous sections on centrosome and nuclei counts to examine regression metrics with CNI quantification.

For simplicity, we present here only the counting association that gave us the best results in CNI quantification, *i.e.* , we recover for each image the centrosome number given by the ResNet-18 model, and we take the number of nuclei resulting from a pre-trained 3D Stardist model, which is then re-trained with simulated data as explained in



(a) Scatter plot of CNI estimate at image level.



(b) Scatter plot of CNI estimate at patient level.

Figure 5.9: Overview of the results obtained for the CNI estimate for both tissue types.

the previous section.

It should be remembered that the quantification of CNI can be done at two levels: either at the level of the image or at the level of the patient. In the latter case, the CNI corresponds to the ratio between the average number of centrosomes and the average number of nuclei present in all fields of the same tissue.

figure 5.9 shows all results in the form of scatter plots. Also, figure A.5 shows a visualization of the ratio of automatically detected values to manually counted values. Overall, the quantified CNI values are relatively unsatisfactory at both the image and patient levels. Despite the fact that centrosome counting models are quite efficient, determining the ratio of the two quantities can be a major source of error. In particular, if we tend to overestimate the number of centrosomes and underestimate the number of nuclei, the error in estimating CNI will accumulate. Thus, although we have good performance scores on the counting of centrosomes and nuclei independently, our approach does not allow here to have a sufficiently relevant quantification of the CNI.

5.5 Conclusions

Counting cells or their components is a very difficult task, especially when done in three dimensions. The main risk is that the same object is counted more than once. In this chapter, we have proposed different counting methods depending on the type of object to be detected or the type of tissue to be examined. In healthy tissues, which do not show any particular degeneration of the cells, the centrosomes are arranged in isolation, and the colocalization between the red and green channels is strong there, resulting in rather intense yellow spots that can be easily detected by thresholding methods or conventional image analysis techniques. In contrast, centrosomes in cancer tissue have a more complex morphology and are extremely difficult to segment. We then show that ML and/or DL models are particularly efficient here compared to visual methods. Note, however, that here traditional methods have the advantage of providing visualization of the detections made, an avenue not explored with the deep regression model used here.

For the counting of the nuclei, the absence of real and public data in 3D forces us to use our nucleus simulator developed in the previous chapter. In the absence of ground-truth at the pixel level, we cannot automatically extract parameters to simulate a relevant dataset. Thus, our approach is to learn the simulation that would allow us to give the best results. We thus associate a set of images, obtained by certain settings, with a certain level of performance on real data. Our results have made it possible to show that a suitable initialization is necessary to achieve high performance, and especially in relatively convenient time. Also, constantly reusing weights that haven't been minimally pretrained is a losing strategy at long term, specially for 3D data.

Finally, after summarizing our best results on the number of nuclei and centrosomes, we did not provide a sufficiently relevant quantification of CNI to draw a biological or medical conclusion. However, this result in no way diminishes the quality of the methods used to perform the counts of centrosome or nuclei alone.

Chapter 6

Conclusions

Contents

6.1 Conclusions 105

6.2 Perspectives and future work 106

This chapter concludes the thesis with an evaluation of the work done and the possible perspectives at the end of this research.

6.1 Conclusions

In this work, we have investigated a number of problems that are common in biomedical imaging and involve either object recognition or image segmentation solutions. The main need that was frequently mentioned in each of the topics addressed is to reduce the amount of time people spend on repetitive, long, complex, and/or tedious tasks. Aside from simply saving time, developing automated methods to perform these tasks also helps improve patient care and may also allow for the expansion of certain research areas on a larger scale. We demonstrated this in the first part of the thesis by using the state-of-art object detection CNN model for early nodule detection on 3D CT scans. Due to their pyramidal structure, these models are able to provide detections at multiple size scales. Also, the presence of a public database, which has already been annotated by several experts and is relatively large, allowed us to effectively train several of these models. Although the process requires some computational power, we were able to compare the models in terms of both detection quality and analysis time. These results were particularly useful for the development of a CADx system during the Data Challenge organized by the JFR. The resulting system combines a ML and DL model and has produced very encouraging results, leading to the belief that AI will be the most important support for preventive medicine in the coming years.

Another important point raised in this thesis is the relationship between training data and model performance. Although this is a rather expected result, we show that in a nuclei segmentation task, segmentation performance tends to increase as the amount of data used in training is increased. To minimize the labeling cost of using real data, our approach was based on the development of a simulator that generates data in a nearly infinite amount. Although this approach cannot fully replace a real dataset, we found that it can be used as a method for data augmentation. When we combine this technique with other, more classical methods, such as the application of rotation or elastic deformation,

we can still see an improvement in performance, which also confirms the quality of our tool.

Finally, the last part of this work is mainly application-oriented and uses methods that are relatively well known in the scientific literature. The detection of centrosomes in healthy tissues, due to their regular shape and rather isolated arrangement, is an example of a method that mainly requires a good definition of some thresholds rather than the implementation of a CNN architecture. As for counting the nuclei, although we saw in the previous chapter that simulation is not necessarily the best option, in this context it is the only viable alternative. Thus, our approach is to control the simulator so that it generates relevant data relative to the task we are trying to perform. Thus, we observed that small changes, such as a good initialization of simulation parameter or a change in the parameters update rule, can be enough to make this type of algorithm converge.

6.2 Perspectives and future work

After all this work is completed, the possible perspectives we can consider arise naturally from each chapter of this manuscript:

- **Unifying a diagnostic pipeline for radiology**: Although the results obtained during the JFRs are very encouraging, the pipeline we have packaged can be seen as a stack of different models whose outputs correspond to the inputs of the others. Thus, the failure of one of the models at the beginning of the chain will most likely lead to the failure of the following models. An alternative to this problem would be to build a unified and multitask model capable of performing all the tasks required for the diagnosis of lung cancer simultaneously (*i.e.* lungs segmentation, nodule detection and classification). To improve the quality of these predictions, it might prove useful to add clinical data on the patient's age, sex, whether or not he is a smoker, etc., in addition to imaging data. Then, it would be interesting to do a full clinical trial on this system to look at the benefits in terms of patient care or patient survival.

- **An extension of the simulation model:** the model proposed in this thesis only generates images of cell nuclei. From new models of other cellular organs, we could enrich our model and its graphical interface. For example, it could have been interesting to also model the colocalization process in two then in three dimensions in order to generate centrosomes within the images. As for the nuclei, an approach by RL and PG methods could thus have been initiated to find what is the level of colocalization that one wishes to simulate. Another relevant possibility, would be to repeat the study conducted in Chapter 4 using multiple data generators. In particular, one would need to be able to include Generative Adversarial Networks (GANs), which are known for the quality and realism of their simulations.

- **Another meta-learning strategy**: The gradient descent method presented in this thesis is a relatively simple approach, but extremely computationally expensive. Moreover, the reward function we have chosen is questionable in that it rewards a correct count without attaching any significance to the segmentation that led to that count. From a practical point of view, situations may arise in which the count is relatively correct while the segmentation is visually incorrect. A semi-automatic approach would then consist in validating/correcting the segmentation that was made with each iteration, but would make the process even more cumbersome. Beside, in the case of large dataset like ours, the large heterogeneity of the data makes it unlikely that a single simulation mode will

be sufficient to resemble the real data. Thus, one approach might be to create multiple simulation modalities to produce greater diversity at each iteration, which would be more appropriate for very complex data sets. The policy gradient approach could be a multimodal policy gradient approach and would allow finding more realistic distributions of the features present in a dataset.

Bibliography

Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans. *Medical physics*, 38(2):915–931, 2011. xiii, 33, 34

Serge Beucher. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing*. CCETT, 1979. 60

F Beyer, L Zierott, EM Fallenberg, KU Juergens, J Stoeckel, W Heindel, and D Wormanns. Comparison of sensitivity and reading time for the use of computer-aided detection (cad) of pulmonary nodules at mdct as concurrent or second reader. *European radiology*, 17(11):2941–2947, 2007. 58

Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1), 2007. 64

PC Bunch, BUNCH PC, and SIMMONS GH. A free-response approach to the measurement and characterization of radiographic-observer performance. 1978. 48

Juan C Caicedo, Allen Goodman, Kyle W Karhohs, Beth A Cimini, Jeanelle Ackerman, Marzieh Haghighi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, et al. Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nature methods*, 16(12):1247–1253, 2019. 61, 77

Niccolò Camarlinghi, Ilaria Gori, Alessandra Retico, Roberto Bellotti, Paolo Bosco, Piergiorgio Cerello, Gianfranco Gargano, Ernesto Lopez Torres, Rosario Megna, Marco Peccarisi, et al. Combination of computer-aided detection algorithms for automatic lung nodule identification. *International journal of computer assisted radiology and surgery*, 7(3):455–464, 2012. 33

Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):1–11, 2006. 101

Junghwan Cho, Kyewook Lee, Ellie Shin, Garry Choy, and Synho Do. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv preprint arXiv:1511.06348*, 2015. 61

Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of digital imaging*, 26(6):1045–1057, 2013. 61

Jonathan WD Comeau, Santiago Costantino, and Paul W Wiseman. A guide to accurate fluorescence microscopy colocalization measurements. *Biophysical journal*, 91(12): 4611–4622, 2006. 69, 92

Paul T Conduit, Alan Wainman, and Jordan W Raff. Centrosome function and assembly in animal cells. *Nature reviews Molecular cell biology*, 16(10):611–624, 2015. 89

Marco Raffaele Cosenza and Alwin Krämer. Centrosome amplification, chromosomal instability and cancer: mechanistic, clinical and therapeutic issues. *Chromosome research*, 24(1):105–126, 2016. 89

Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Human vision and electronic imaging XII*, volume 6492, pages 196–206. SPIE, 2007. 78

Sebastian Cygert and Andrzej Czyżewski. Toward robust pedestrian detection with data augmentation. *IEEE Access*, 8:136674–136683, 2020. 82

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 21

Gautier Defossez, Sandra Le Guyader-Peyrou, Zoé Uhry, Pascale Grosclaude, Marc Colonna, Emmanuelle Dantony, Patricia Delafosse, Florence Molinié, Anne-Sophie Woronoff, Anne-Marie Bouvier, et al. Estimations nationales de l'incidence et de la mortalité par cancer en france métropolitaine entre 1990 et 2018. *Synthèse. Saint-Maurice: Santé publique France*, 2019. 88

Kenneth W Dunn, Malgorzata M Kamocka, and John H McDonald. A practical guide to evaluating colocalization in biological microscopy. *American Journal of Physiology-Cell Physiology*, 300(4):C723–C742, 2011. 92, 95

Stanley Durrleman, Marcel Prastawa, Nicolas Charon, Julie R Korenberg, Sarang Joshi, Guido Gerig, and Alain Trouvé. Morphometry of anatomical shape complexes with dense deformations and sparse parameters. *NeuroImage*, 101:35–49, 2014. 62

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 27, 47

Elnaz Fazeli, Nathan H Roy, Gautier Follain, Romain F Laine, Lucas von Chamier, Pekka E Hänninen, John E Eriksson, Jean-Yves Tinevez, and Guillaume Jacquemet. Automated cell tracking using stardist and trackmate. *F1000Research*, 9, 2020. 73

Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. 23

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 44

Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928, 2021. 82

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. xiii, 23, 25, 44

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. xiii, 22

SA Godinho and D Pellman. Causes and consequences of centrosome abnormalities in cancer. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1650): 20130467, 2014. 89

Michael Götz. Mic-dkfz/lidc-idri-processing: Release 1.0.1. Dec 2018. doi: 10.5281/ZENODO.2249217. URL https://zenodo.org/record/2249217. 35

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 23

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 16, 37

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 25

Nanda Horeweg, Joost van Rosmalen, Marjolein A Heuvelmans, Carlijn M van der Aalst, Rozemarijn Vliegenthart, Ernst Th Scholten, Kevin ten Haaf, Kristiaan Nackaerts, Jan-Willem J Lammers, Carla Weenink, et al. Lung cancer probability in patients with ct-detected pulmonary nodules: a prespecified analysis of data from the nelson trial of low-dose ct screening. *The Lancet Oncology*, 15(12):1332–1341, 2014. 56

H-H Hsu, K-H Ko, Y-C Chou, Y-C Wu, S-H Chiu, C-K Chang, and W-C Chang. Performance and reading time of lung nodule identification on multidetector ct with or without an artificial intelligence-powered computer-aided detection system. *Clinical Radiology*, 76(8):626–e23, 2021. 58

Lih-Ching Hsu, Malathy Kapali, Julie A DeLoia, and Holly H Gallion. Centrosome abnormalities in ovarian cancer. *International journal of cancer*, 113(5):746–751, 2005. 89, 91

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017. xiii, 20

Kai Huang and Robert F Murphy. From quantitative microscopy to automated image understanding. *Journal of biomedical optics*, 9(5):893–912, 2004. 60

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992. 39

Vladimir Iglovikov, Sergey Mushinskiy, and Vladimir Osin. Satellite imagery feature detection using deep convolutional neural network: A kaggle competition. *arXiv preprint arXiv:1706.06169*, 2017. 19

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 15

Marcin P Iwanicki, Rachel A Davidowitz, Mei Rosa Ng, Achim Besser, Taru Muranen, Melissa Merritt, Gaudenz Danuser, Tan Ince, and Joan S Brugge. Ovarian cancer spheroids use myosin-generated force to clear the mesotheliumcontractile forces generated by ovarian cancer spheroids promote mesothelial clearance. *Cancer discovery*, 1 (2):144–157, 2011. 88

Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901. 43

Paul F. Jaeger, Simon A. A. Kohl, Sebastian Bickelhaupt, Fabian Isensee, Tristan Anselm Kuder, Heinz-Peter Schlemmer, and Klaus H. Maier-Hein. Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection, 2018. xiv, 33, 36, 37, 38, 55, 58

Yong Won Jin, Shuo Jia, Ahmed Bilal Ashraf, and Pingzhao Hu. Integrative data augmentation with u-net segmentation masks improves detection of lymph node metastases in breast cancer patients. *Cancers*, 12(10):2934, 2020. 82

Sathiesh Kumar Kaliyugarasan, Arvid Lundervold, and Alexander Selvikvåg Lundervold. Pulmonary nodule classification in lung cancer from 3d thoracic ct scans using fastai and monai. 2021. 58

Periklis Katopodis, Dimple Chudasama, Gurleen Wander, Louise Sales, Juhi Kumar, Manreen Pandhal, Vladimir Anikin, Jayanta Chatterjee, Marcia Hall, and Emmanouil Karteris. Kinase inhibitors and ovarian cancer. *Cancers*, 11(9):1357, 2019. xvi, 89

Howard Kaufman and A Murat Tekalp. Survey of estimation techniques in image restoration. *IEEE control systems magazine*, 11(1):16–24, 1991. 68

Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. Cnn-based segmentation of medical imaging data. *arXiv preprint arXiv:1701.03056*, 2017. 29

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 10, 50

Aiko Klein, Richard Van Den Doel, Ian T Young, Stephanie Ellenberger, and Lucas Van Vliet. Quantitative evaluation and comparison of light microscopes. *image*, 1000 (1):2, 1998. 67

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 4, 14, 16

Florian Kromp, Eva Bozsaky, Fikret Rifatbegovic, Lukas Fischer, Magdalena Ambros, Maria Berneder, Tamara Weiss, Daria Lazic, Wolfgang Dörr, Allan Hanbury, et al. An annotated fluorescence image dataset for training nuclear segmentation methods. *Scientific Data*, 7(1):1–8, 2020. 77

Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE transactions on medical imaging*, 36(7):1550–1560, 2017. 76

N. Lassau, I. Bousaid, E. Chouzenoux, J.P. Lamarque, B. Charmettant, M. Azoulay, F. Cotton, A. Khalil, O. Lucidarme, F. Pigneur, Y. Benaceur, A. Sadate, M. Lederlin, F. Laurent, G. Chassagnon, O. Ernst, G. Ferreti, Y. Diascorn, P.Y. Brillet, M. Creze, L. Cassagnes, C. Caramella, A. Loubet, A. Dallongeville, N. Abassebay, M. Ohana, N. Banaste, M. Cadi, J. Behr, L. Boussel, L. Fournier, M. Zins, J.P. Beregi, A. Luciani, A. Cotten, and J.F. Meder. Three artificial intelligence data challenges based on ct and mri. *Diagnostic and Interventional Imaging*, 101(12):783–788, 2020. ISSN 2211-5684. doi: https://doi.org/10.1016/j.diii.2020.03.006. URL https://www.sciencedirect.com/science/article/pii/S2211568420300838. xvii, 55, 57

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989. 11

Antti Lehmussola, Jyrki Selinummi, Pekka Ruusuvuori, Antti Niemisto, and Olli Yli-Harja. Simulating fluorescent microscope images of cell populations. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 3153–3156. IEEE, 2006. 67

Antti Lehmussola, Pekka Ruusuvuori, Jyrki Selinummi, Heikki Huttunen, and Olli Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE transactions on medical imaging*, 26(7):1010–1016, 2007. 62, 67

Ruirui Li, Wenjie Liu, Lei Yang, Shihao Sun, Wei Hu, Fan Zhang, and Wei Li. Deepunet: A deep fully convolutional network for pixel-level sea-land segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(11):3954–3962, 2018. 19

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. xvii, 27, 48

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 936–944. IEEE Computer Society, 2017a. doi: 10.1109/CVPR.2017.106. URL https://doi.org/10.1109/CVPR.2017.106. 28

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017b. 27

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. xiii, 27

Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637, 2012. 61, 101

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 17, 18

David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999. 21

Norberto Malpica, Carlos Ortiz De Solórzano, Juan José Vaquero, Andrés Santos, Isabel Vallcorba, José Miguel García-Sagredo, and Francisco Del Pozo. Applying watershed algorithms to the segmentation of clustered nuclei. *Cytometry: The Journal of the International Society for Analytical Cytology*, 28(4):289–297, 1997. 60

EMM Manders, FJ Verbeek, and JA Aten. Measurement of co-localization of objects in dual-colour confocal images. *Journal of microscopy*, 169(3):375–382, 1993. 96

Gaëlle Marteil, Adan Guerrero, André F Vieira, Bernardo P De Almeida, Pedro Machado, Susana Mendonça, Marta Mesquita, Beth Villarreal, Irina Fonseca, Maria E Francia, et al. Over-elongation of centrioles in cancer promotes centriole amplification and chromosome missegregation. *Nature communications*, 9(1):1–17, 2018. 89

Véronique Marthiens, Matthieu Piel, and Renata Basto. Never tear us apart–the importance of centrosome clustering. *Journal of cell science*, 125(14):3281–3292, 2012. xvi, 90

Ursula A Matulonis, Anil K Sood, Lesley Fallowfield, Brooke E Howitt, Jalid Sehouli, and Beth Y Karlan. Ovarian cancer. *Nature reviews Disease primers*, 2(1):1–22, 2016. 88

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. xii, 5, 6

Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016. 29, 39

Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 479:480, 1969. 6

Anirban K Mitra. *Ovarian cancer metastasis: a unique mechanism of dissemination*. IntechOpen, 2016. 88

Ihab S Mohamed. *Detection and tracking of pallets using a laser rangefinder and machine learning techniques*. PhD thesis, European Master on Advanced Robotics+(EMARO+), University of Genova, Italy, 2017. xii, 11

Jean-Philippe Morretton, Aurélie Herbette, Camille Cosson, Bassirou Mboup, Aurélien Latouche, Pierre Gestraud, Tatiana Popova, Marc-Henri Stern, Fariba Nemati, Didier Decaudin, et al. Centrosome amplification favours survival and impairs ovarian cancer progression. *bioRxiv*, page 623983, 2019. xvi, 91

Jean-philippe Morretton, Anthony Simon, Aurélie Herbette, Jorge Barbazan, Carlos Pérez-González, Camille Cosson, Bassirou Mboup, Aurélien Latouche, Tatiana Popova, Yann Kieffer, et al. A catalog of numerical centrosome defects in epithelial ovarian cancers. *EMBO Molecular Medicine*, 14(9):e15670, 2022. xvi, 94

Erich A Nigg. Origins and consequences of centrosome aberrations in human cancers. *International journal of cancer*, 119(12):2717–2723, 2006. 89

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979. 52, 60

Diego M Peña, Shouhua Luo, and Abdeldime Abdelgader. Auto diagnostics of lung nodules using minimal characteristics extraction technique. *Diagnostics*, 6(1):13, 2016. 33

Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985. 67

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. xiii, 26, 44

Brett M Reid, Jennifer B Permuth, and Thomas A Sellers. Epidemiology of ovarian cancer: a review. *Cancer biology & medicine*, 14(1):9, 2017. 88

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. xiii, 23, 24

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. xii, 19, 37

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 6

Cécile Rousselle, Sylvain Paillasson, Michel Robert-Nicoud, and Xavier Ronot. Chromatin texture analysis in living cells. *The Histochemical Journal*, 31(1):63–70, 1999. 67

Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. *arXiv preprint arXiv:1810.02513*, 2018. 98

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 9

Alex Martins Santos, Antonio Oseas de Carvalho Filho, Aristófanes Corrêa Silva, Anselmo Cardoso de Paiva, Rodolfo Acatauassú Nunes, and Marcelo Gattass. Automatic detection of small lung nodules in 3d ct data using gaussian mixture models, tsallis entropy and svm. *Engineering applications of artificial intelligence*, 36:27–39, 2014. 33

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018. 15

Hector Sanz, Clarissa Valim, Esteban Vegas, Josep M Oller, and Ferran Reverter. Svm-rfe: selection and visualization of the most relevant features through non-linear kernels. *BMC bioinformatics*, 19(1):1–18, 2018. 57

Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer, 2018. 73

Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004. 96

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. xii, 14, 16

Kevin Smith, Yunpeng Li, Filippo Piccinini, Gabor Csucs, Csaba Balazs, Alessandro Bevilacqua, and Peter Horvath. Cidre: an illumination-correction method for optical microscopy. *Nature methods*, 12(5):404–406, 2015. 67

Michiel Stevens, Afroditi Nanou, Leon WMM Terstappen, Christiane Driemel, Nikolas H Stoecklein, and Frank AW Coumans. Stardist image segmentation improves circulating tumor cell detection. *Cancers*, 14(12):2916, 2022. 73

Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106, 2021. 74

David Svoboda, Michal Kozubek, and Stanislav Stejskal. Generation of digital phantoms of cell nuclei and simulation of image formation in 3d image cytometry. *Cytometry Part A: The Journal of the International Society for Advancement of Cytometry*, 75(6):494–509, 2009. 67, 69

Stephen J Swensen, James R Jett, Thomas E Hartman, David E Midthun, Jeff A Sloan, Anne-Marie Sykes, Gregory L Aughenbaugh, and Medy A Clemens. Lung cancer screening with ct: Mayo clinic experience. *Radiology*, 226(3):756–761, 2003. 32

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 16

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. 2012. 10

Yosuke Toda, Fumio Okura, Jun Ito, Satoshi Okada, Toshinori Kinoshita, Hiroyuki Tsuji, and Daisuke Saisho. Training instance segmentation neural network with synthetic datasets for crop seed phenotyping. *Communications biology*, 3(1):1–12, 2020. 61

Le-Anh Tran and My-Ha Le. Robust u-net-based road lane markings detection for autonomous driving. In *2019 International Conference on System Science and Engineering (ICSSE)*, pages 62–66. IEEE, 2019. 19

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018. 61

Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886. IEEE, 2011. 22

Bram Van Ginneken, Samuel G Armato III, Bartjan de Hoop, Saskia van Amelsvoort-van de Vorst, Thomas Duindam, Meindert Niemeijer, Keelin Murphy, Arnold Schilham, Alessandra Retico, Maria Evelina Fantacci, et al. Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the anode09 study. *Medical image analysis*, 14(6):707–722, 2010. 35, 49

Lorenzo Vassallo, Alberto Traverso, Michelangelo Agnello, Christian Bracco, Delia Campanella, Gabriele Chiara, Maria Evelina Fantacci, Ernesto Lopez Torres, Antonio Manca, Marco Saletta, et al. A cloud-based computer-aided detection system improves identification of lung nodules on computed tomography scans of patients with extra-thoracic malignancies. *European Radiology*, 29(1):144–152, 2019. 58

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001. xiii, 21, 22

Carolina Wählby, I-M Sintorn, Fredrik Erlandsson, Gunilla Borgefors, and Ewert Bengtsson. Combining intensity, edge and shape information for 2d and 3d segmentation of cell nuclei in tissue sections. *Journal of microscopy*, 215(1):67–76, 2004. 60

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017. 61

Martin Weigert and Uwe Schmidt. Nuclei segmentation and classification in histopathology images with stardist for the conic challenge 2022. *arXiv preprint arXiv:2203.02284*, 2022. 73

Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3666–3673, 2020. 70

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. 100

Helen T Winer-Muram. The solitary pulmonary nodule. *Radiology*, 239(1):34–49, 2006. 32

Chenyang Xu and Jerry L Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 66–71. IEEE, 1997. 74

Richarda Young. The gaussian derivative model for spatial vision. i- retinal mechanisms. *Spatial vision*, 2(4):273–293, 1987. 93

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 16

Yingru Zhao, Geertruida H de Bock, Rozemarijn Vliegenthart, Rob J van Klaveren, Ying Wang, Luca Bogoni, Pim A de Jong, Willem P Mali, Peter MA van Ooijen, and Matthijs Oudkerk. Performance of computer-aided detection of pulmonary nodules in low-dose ct: comparison with double reading by nodule volume. *European radiology*, 22(10): 2076–2084, 2012. 33

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018. 29

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, 2019. 29

Peiye Zhuang, Alexander G Schwing, and Oluwasanmi Koyejo. Fmri data augmentation via synthesis. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1783–1787. IEEE, 2019. 61

Vadim Zinchuk and Olga Grossenbacher-Zinchuk. Recent advances in quantitative colocalization analysis: focus on neuroscience. *Progress in histochemistry and cytochemistry*, 44(3):125–172, 2009. 92

Appendix A

Appendix

A.1 Appendix on chapter 3

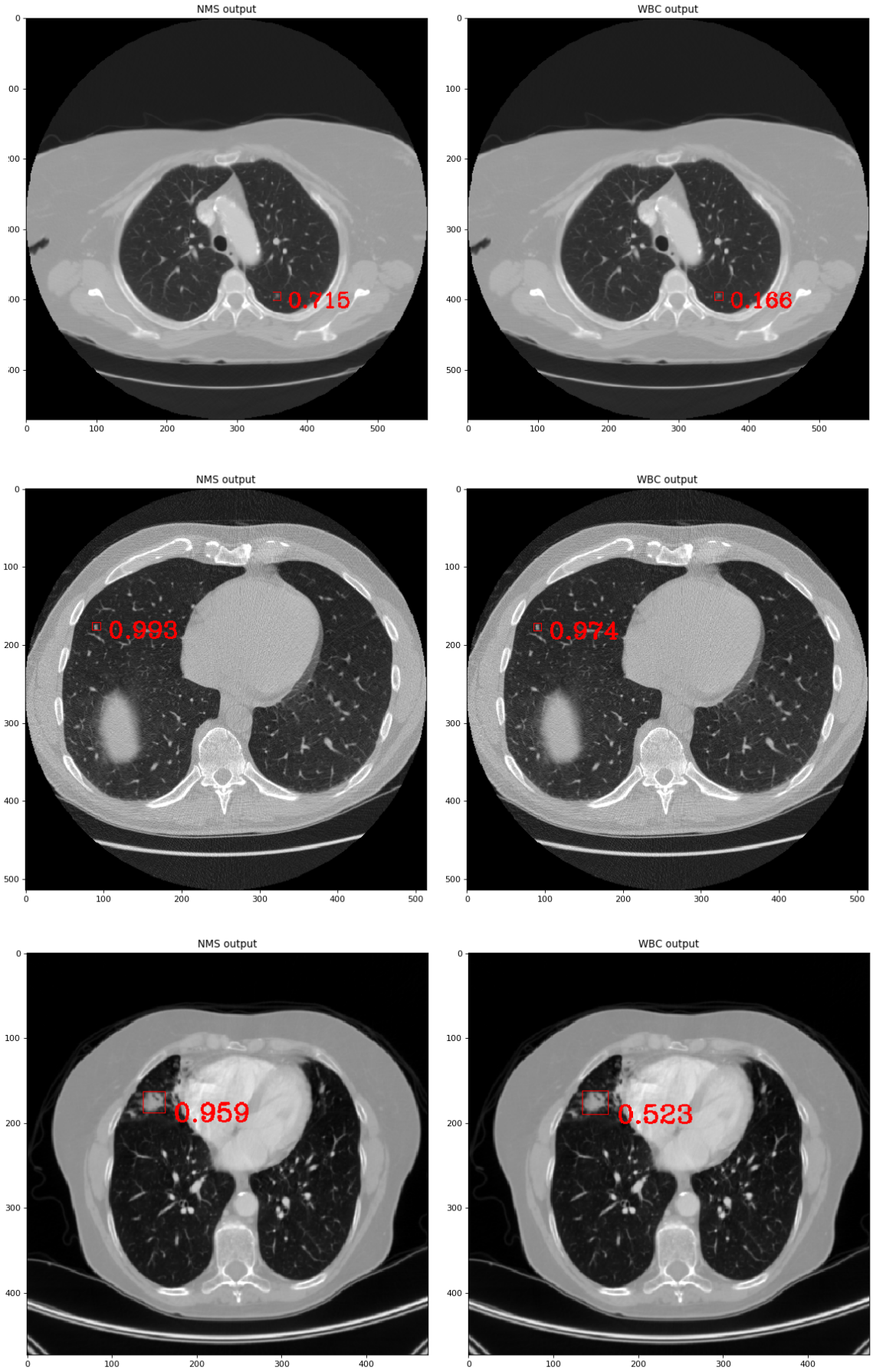


Figure A.1: A visual comparison of false positive detections obtained after applying NMS algorithm (left) versus WBC algorithm (right).

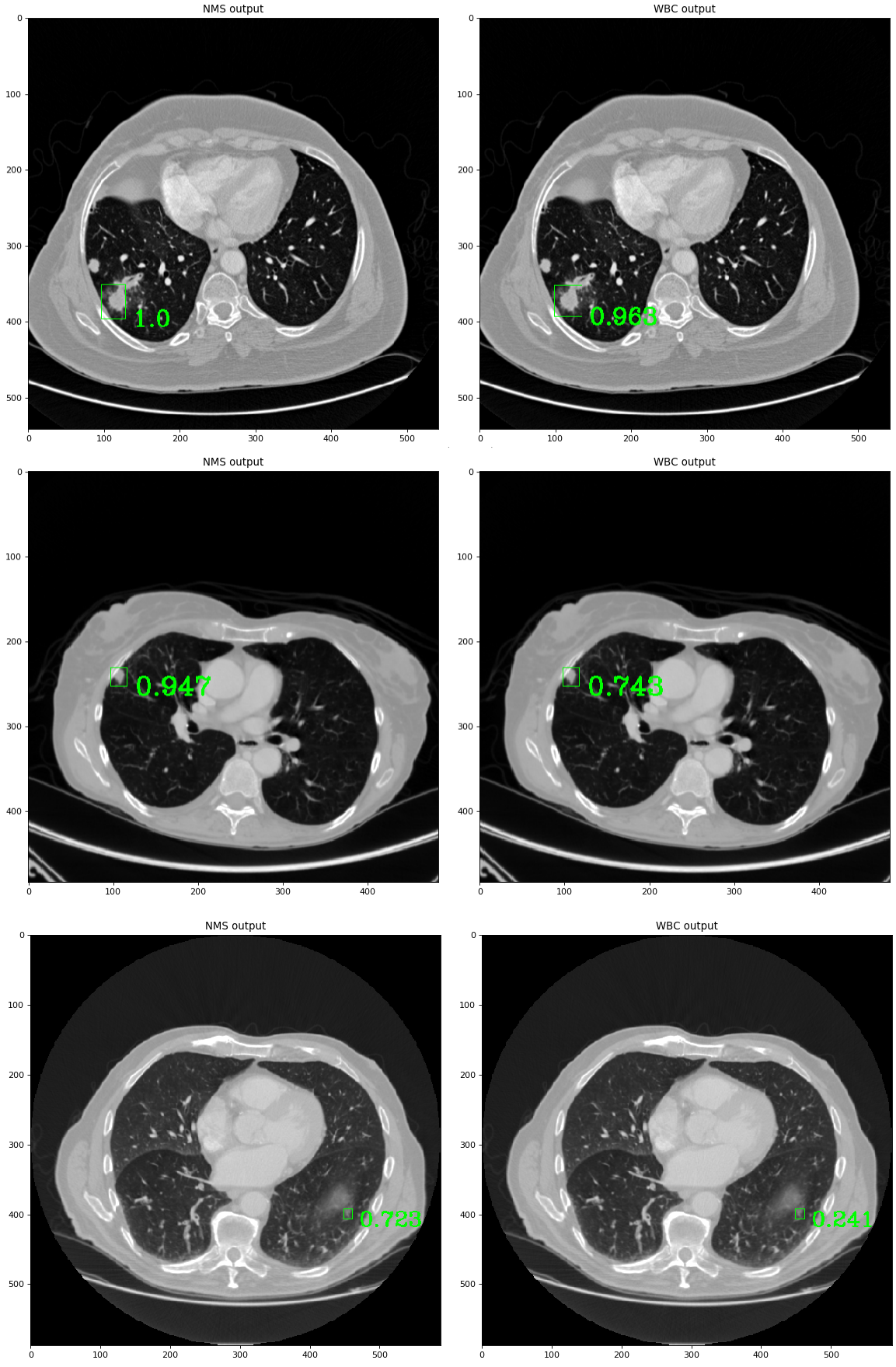
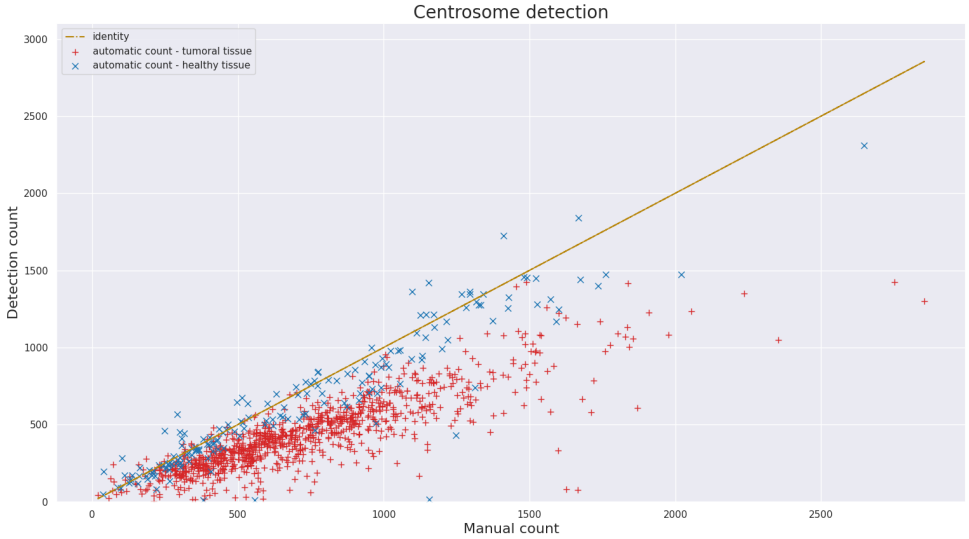
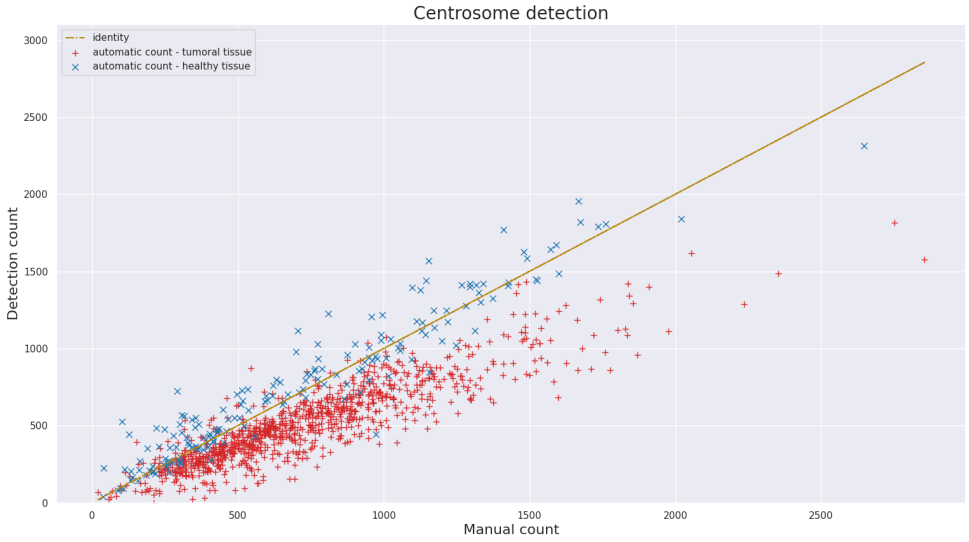


Figure A.2: A visual comparison of true positive detections obtained after applying NMS algorithm (left) versus WBC algorithm (right).

A.2 Appendix on Chapter 5

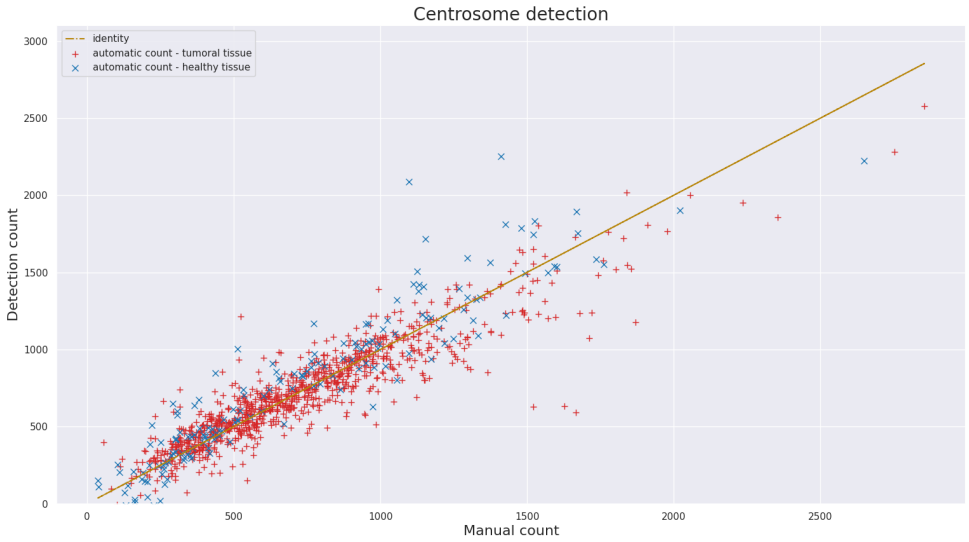


(a) Visual approach

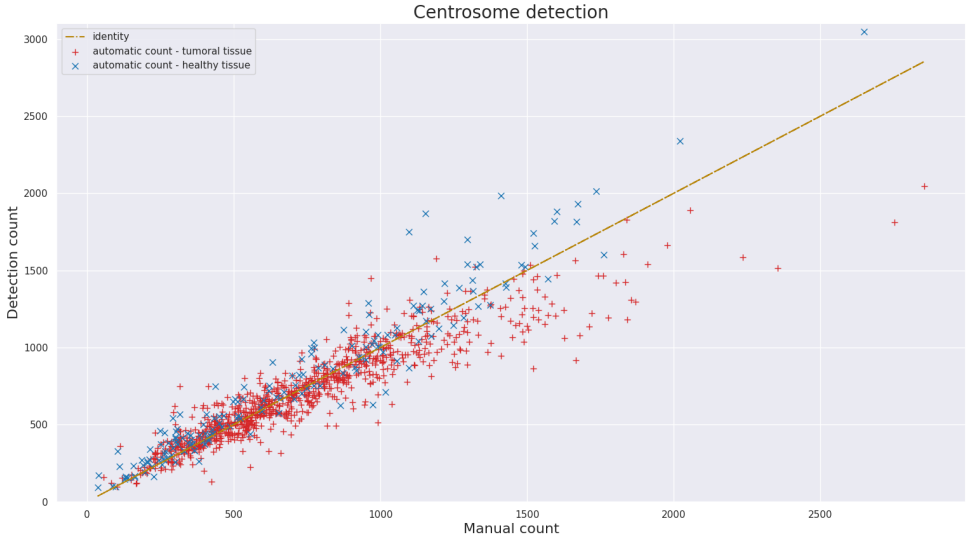


(b) Watershed-based approach

Figure A.3: Performance overview of different approach for centrosome counting.

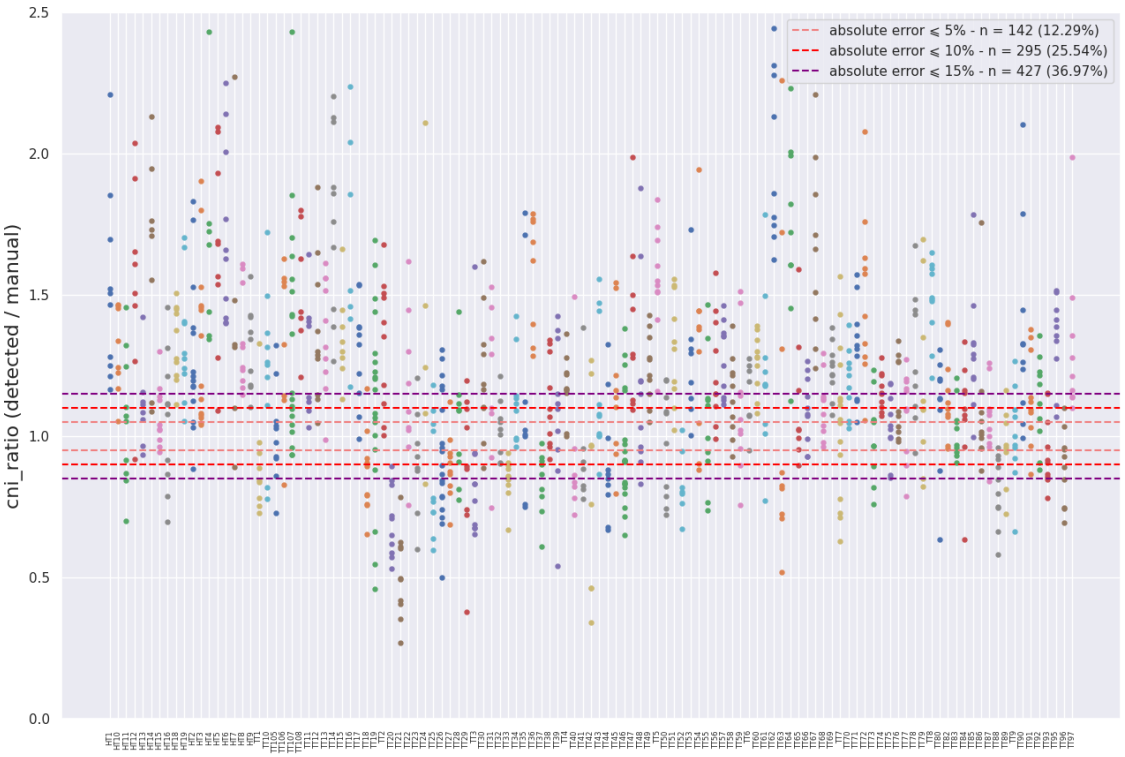


(a) Machine learning approach

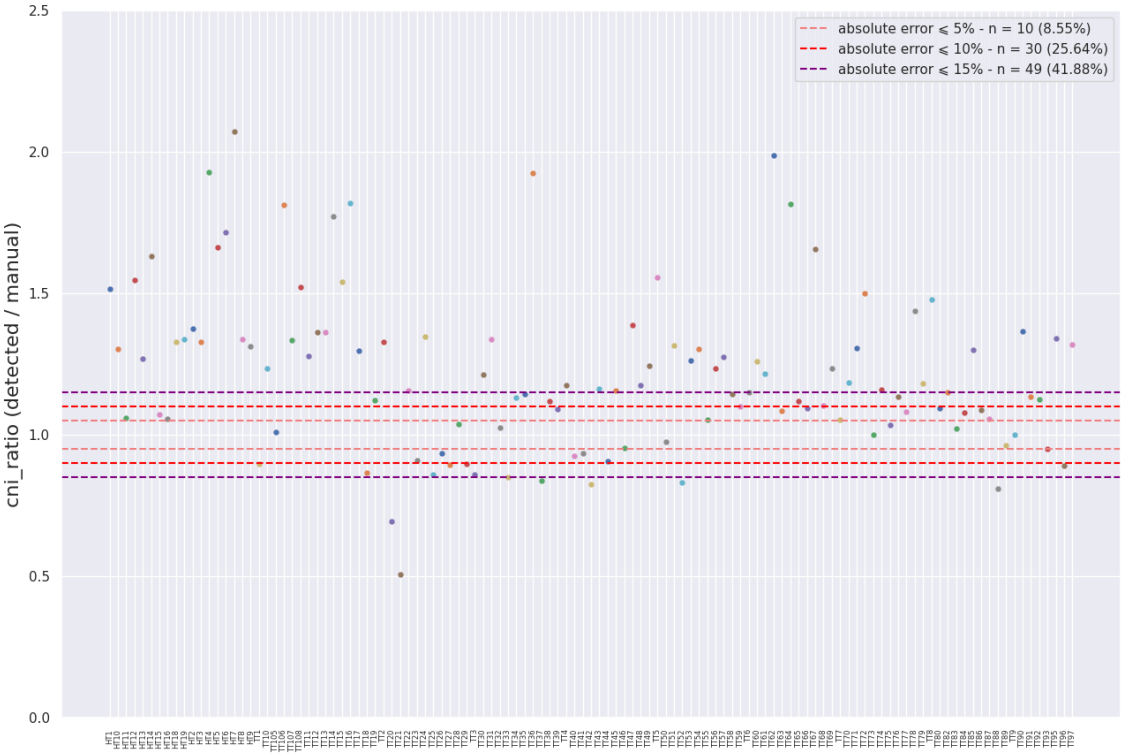


(b) Deep learning approach

Figure A.4: Performance overview of different approach for centrosome counting.



(a) Visualization of the relative error of the CNI estimate at the image level



(b) Visualization of the relative error of the CNI estimate at the patient level.

Figure A.5: Visualization of the relative error of the CNI estimate for both image and patient level.