Université de Liège - Faculté des sciences

Détection d'objets rares au sein de coupes histologiques par apprentissage automatique

Présenté par:
Ahmed Debit

Promoteur:
Pr. Pierre Geurts

Membres du jury:
Raphaël Marée (Institut Montéfiore/GIGA)
Carine Munaut (GIGA)
Silvia Blacher (GIGA)

Travail de fin d'études *pour l'obtention du grade*
Master 2 Bioinformatique et Modélisation à finalité approfondie

Septembre 2014

“Le succès consiste à aller d’échecs en échecs sans perdre son enthousiasme.”

Winston Churchill

UNIVERSITÉ DE LIÈGE

Résumé

Faculté des sciences
Bioinformatique et Modélisation

Travail de Fin d'Études

Détection d'objets rares au sein de coupes histologiques par apprentissage automatique

par Ahmed DEBIT

Nous utilisons une des techniques de l'apprentissage supervisé en l'occurence la méthode des arbres aléatoires avec tirage aléatoire de sous-fenêtres pour la construction d'un modèle d'apprentissage dans le but de détecter et de classifier les follicules ovariens dans des images de lames histologiques de l'ovaire. Une application directe de la méthode sur les images d'entrée (exemples) est réalisée sans aucune étape de prétraitement de ces images. Cependant, une étape de post-traitement est appliquée sur les sorties de la méthode. Il s'agit d'un simple *Clustering*[1] de positions (coordonnées géométriques) des follicules détectés suivi d'une fusion (*Merging*) des régions chevauchées. Les résultats de la détection semblent encourageant, contrairement à ceux de la classification qui restent à améliorer.

[1] L'appélation *Clustering* fait penser aux algorithmes connus à l'instar de $K - means$, mais dans notre cas nous préférerons employer ce mot pour désigner l'algorithme utilisé pour la formation des clusters

Remerciements

Mes remerciements vont à toutes les personnes qui par leurs conseils et suggestions d'idées ont permis la réalisation de ce travail.

Je commence à adresser mes premiers remerciements au *Pr. Pierre Geurts*, et à *Raphael Maree* PhD, gestionnaire de la plateforme bioinformatique GIGA qui a eu l'idée de me proposer ce sujet. Je le remercie pour le temps qu'il a consacré, ses suggestions, et ses corrections tout au long du projet.

J'adresse mes remerciements également à toute l'équipe du GIGA plus particulièrement *Gilles Louppe*, et *Benjamin Stevens* pour leurs interventions.

Je tiens également à remercier toute l'équipe du Laboratoire de Biologie des Tumeurs et du Développement LBTD, particulièrement *Fransolet Maite* pour ses annotations.

Je remercie également tous mes ensignants de l'institut Montéfiore et ceux de la faculté des sciences.

Un grand merci pour toute ma famille en Algérie et en France, ainsi qu'à tous mes proches pour leurs soutiens et leurs aides, sans oublier mes amis résidant en Belgique en particulier *Hocine Bendou*.

Enfin merci à tous mon entourage.

Table du contenu

Liste des Figures

Liste des Tables

Chapitre 1

Introduction

À la naissance, l'ovaire humain est doté d'un nombre fini de follicules qui représentent le potentiel complet de la reproduction de l'individu. Ce nombre est estimé approximativement aux environs de 500K-1M [9]. Ce pool de follicules est en diminution progressive à travers le durée de vie de l'individu jusqu'à son épuisement, une période qui représente la fin de la durée de vie de la reproduction. Par conséquent, la durée de vie reproductive est finalement déterminée par le nombre de follicules de l'ovaire.

L'estimation précise du nombre de follicules ovariens à différents stades de la reproduction est un indicateur important du processus de la folliculogénèse en relation avec les signaux endocriniens et les mécanismes paracrine/autocrine qui régulent le croissance et la maturation des ovocytes ainsi que leurs cellules folliculaires [4]. Celà peut être important, par exemple pour déterminer les rôles respectifs des hormones gonadotropes et stéroïdes et les facteurs de croissance intraovarienne locales dans la régulation de la survie et la maturation des follicules à tous les stades de leur développement [4].

En outre, cette estimation a été utilisée comme une mesure de la réserve de la reproduction. Le nombre de follicules primordiaux est en diminution continue depuis la période fœtale jusqu'à la ménopause, période durant laquelle le nombre de follicules primordiaux dans l'ovaire atteint un seuil en dessous de lequel on constate une cessation de l'ovulation. Par conséquent, la quantification précise des follicules est capitale pour la compréhension du processus du vieillissement de l'ovaire (reproduction) et l'investigation de la physiologie ovarienne [10].

Dans ce contexte, l'objectif principal de ce travail est de fournir une méthode de détection (quantification) précise de follicules ainsi que leurs classifications suivant leurs stades de maturation au sein d'images de lames histologiques en haute résolution (0.45 μm/*pixel*; ±70000x40000 pixels). Les classes de maturation des follicules sont principalement et chronologiquement primordiale, primaire, et secondaire. La méthode doit opérer sur des images d'une lame entière.

L'utilisation des méthodes issues de l'apprentissage supervisé dans l'analyse des images est en forte croissance. Classiquement, ce genre de pratique commence par une phase de prétraitement des images à analyser: transformation, suppression du bruit, etc., suivie de l'application effective de la méthode d'apprentissage. La méthode d'apprentissage utilisée dans ce travail est la méthode des arbres aléatoires avec tirage alétoire de sous fenêtres ($Extra - trees$ with random subwindows) développée par l'équipe de Systèmes et Modélisation (Institut Motefiore, GIGA). Il est à noter que dans notre cas, aucun prétraitement n'est préalablement réalisé sur les images (annotations) d'entrée. Il s'agit d'une application directe d'une technique d'apprentissage sur les images suivie d'une étape de postraitement des prédictions.

Ce travail est scindé en sept chapitres. Nous avons jugé qu'il est utile de commencer par exposer quelques préceptes biologiques en relation avec le sujet traité. Nous enchainons ensuite par donner un bref état de l'art des méthodes utilisées pour la quantification de la réserve ovarienne en follicules. Dans le chapitre 4, nous rappelons les notions théoriques de l'apprentissage supervisé principalement les méthodes d'ensemble, et les arbres aléatoires. Nous avons consacré le chapitre 5 à détailler la méthode des $Extra - trees$ avec tirage aléatoire de sous-fenêtres en mettant l'accent sur les paramètres de la méthode. Nous exposons dans le chapitre 6, le framework de la méthode utilisée et son application. Les résultats et les tests de la méthode, ainsi qu'une mesure des performances en terme de temps d'exécution, et des différentes métriques de détection: sensibilité (recall), précision, F-measure, sont montrés dans le chapitre 7. Un résumé du travail, et des perspectives sont donnés en guise de conclusion.

Chapitre 2

Préceptes biologiques et description des données

Dans le système de reproduction de la femelle, un follicule est un "sac rempli" de fluide qui contient un oeuf immature ou ovocyte. Les follicules ont deux caractéristiques importantes: d'une part, ils secrètent des hormones (notamment l'œstradiol), et d'autre part, ces follicules grossissent régulièrement. Les follicules sont présents dans les ovaires. Au cours de l'ovulation, un œuf mature est libéré par un seul follicule, les autres follicules ne donnant pas lieu à un oeuf mature se voient désintégrés. Le follicule qui libère un ovule se transforme en un corps jaune.

2.1 Stades de développement d'un follicule (folliculogenèse)

Au début de la folliculogenèse, on compte jusqu'à 7 millions de cellules folliculaires dès la 20e semaine du développement embryonnaire, mais leur nombre va diminuer rapidement jusqu'à 400 000 follicules à la puberté pour aboutir à 1500 vers l'âge de 51 ans, âge moyen de la ménopause [11].

La chronologie du développement d'un follicule passe par plusieurs stades folliculaires:

follicule primordial $\Longrightarrow$ follicule primaire $\Longrightarrow$ follicule secondaire $\Longrightarrow$ follicule tertiaire

La plupart des follicules n'atteindront pas la maturité (stade tertiaire) nécessaire à l'ovulation et dégénéreront (c'est ce qu'on appelle l'atrésie). Dès qu'un ovocyte entame sa croissance, il se voit entourer d'une couche de cellules folliculaires aplaties disposées en couronne séparée de l'ovaire proprement dit (ie. stroma ovarien) par une membrane (membrane de *Slavianski*), c'est la formation du follicule *primordial* qui mesure environ 0.05mm de diamètre. Les follicules primordiaux constituent la partie dominante du nombre de follicules dans l'ovaire. Sous l'action des hormones, les cellules aplaties vont se transformer en cellules cubiques (aussi appelée prismatiques), c'est la formation du follicule *primaire* à peine plus gros. Le follicule *secondaire* est constitué d'une couronne folliculaire pluristratifiée (plusieurs couches de cellules prismatiques qui entourent l'ovocyte) et mesurant environ 0.2mm [12].

Le follicule secondaire peut évoluer en follicule *tertiaire* aussi appelé follicule *antral*. Il est caractérisé par une zone lacunaire (un vide dans la cellule contenant un liquide) et le fait que l'ovocyte est entouré de cellules prismatiques. Le follicule tertiaire peut évoluer en follicule de $Graaf$ qui peut atteindre 15 à 30 mm et dont on suppose qu'il est le seul follicule permettant l'ovulation. Lors de l'ovulation le follicule se rompt et expulse l'ovocyte dans le pavillon de l'ovaire (figure 2.1).

2.2 Caractéristiques des follicules

Un follicule est dit primordial s'il contient un ovocyte entouré d'une couche partielle ou complète de cellules applaties (granulosa). Un follicule primaire montre une couche unique de cellules de formes cubiques qui entourent un ovocyte. Des follicules sont classés intermédiaires (entre le stade primordial et primaire) s'ils contiennent un ovocyte qui est bordé à la fois par des cellules applaties et cubiques. Un ovocyte entouré de plusieurs couches de cellules granulosa cubiques sans antrum visible est classé comme follicule secondaire. Les follicules antraux précoces possèdent des petits espaces antraux, tandisque ces espaces deviennent plus importants dans le cas des follicules antraux (figure 2.2).

Cependant cette classification [4] ne tient pas compte de la présence/absence du noyau dans chaque classe folliculaire. En réalité, tous les follicules ne contiennent pas un noyau en leur sein, c'est la raison pour laquelle, on a ajouté ce critère de

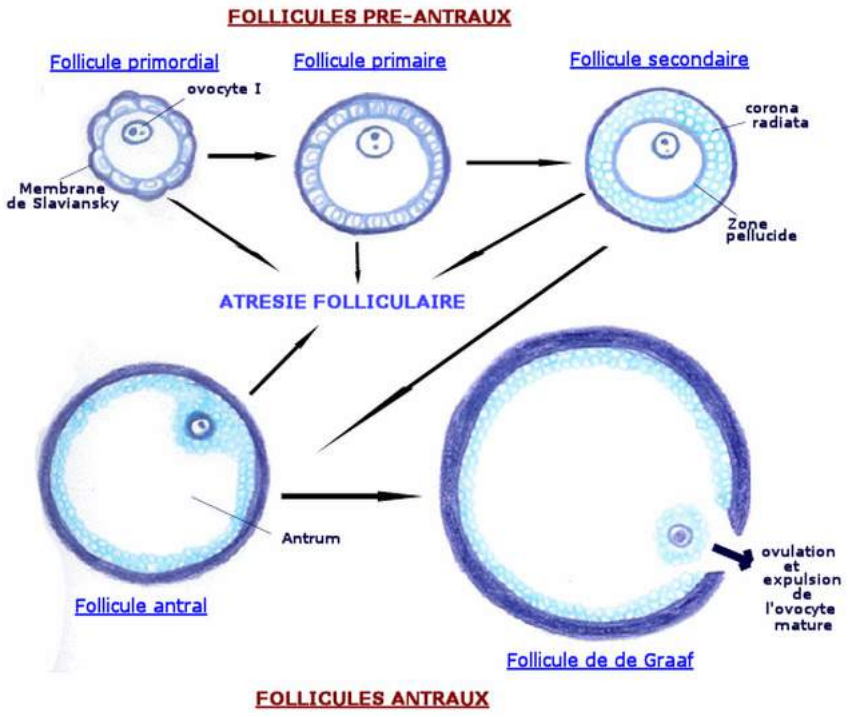


FIGURE 2.1: les différents stades de la folliculogenèse. Figure tirée des slides en lignes Physiologie et Infécondité Dr Maxime Barré 2009

classification dans notre étude. Par conséquent, six (06) classes de follicules sont utilisées dans notre cas.

2.3 Les classes folliculaires

2.3.1 Follicule primordial sans noyau

Quelques cellules folliculaires aplaties entourent l'ovocyte pour former un follicule primordial (figure 2.3).

2.3.2 Follicule primordial avec noyau

La ceinture de cellules folliculaires applaties est plus visible. le noyau de l'ovocyte peut être distingué au sein du follicule (figure 2.4).

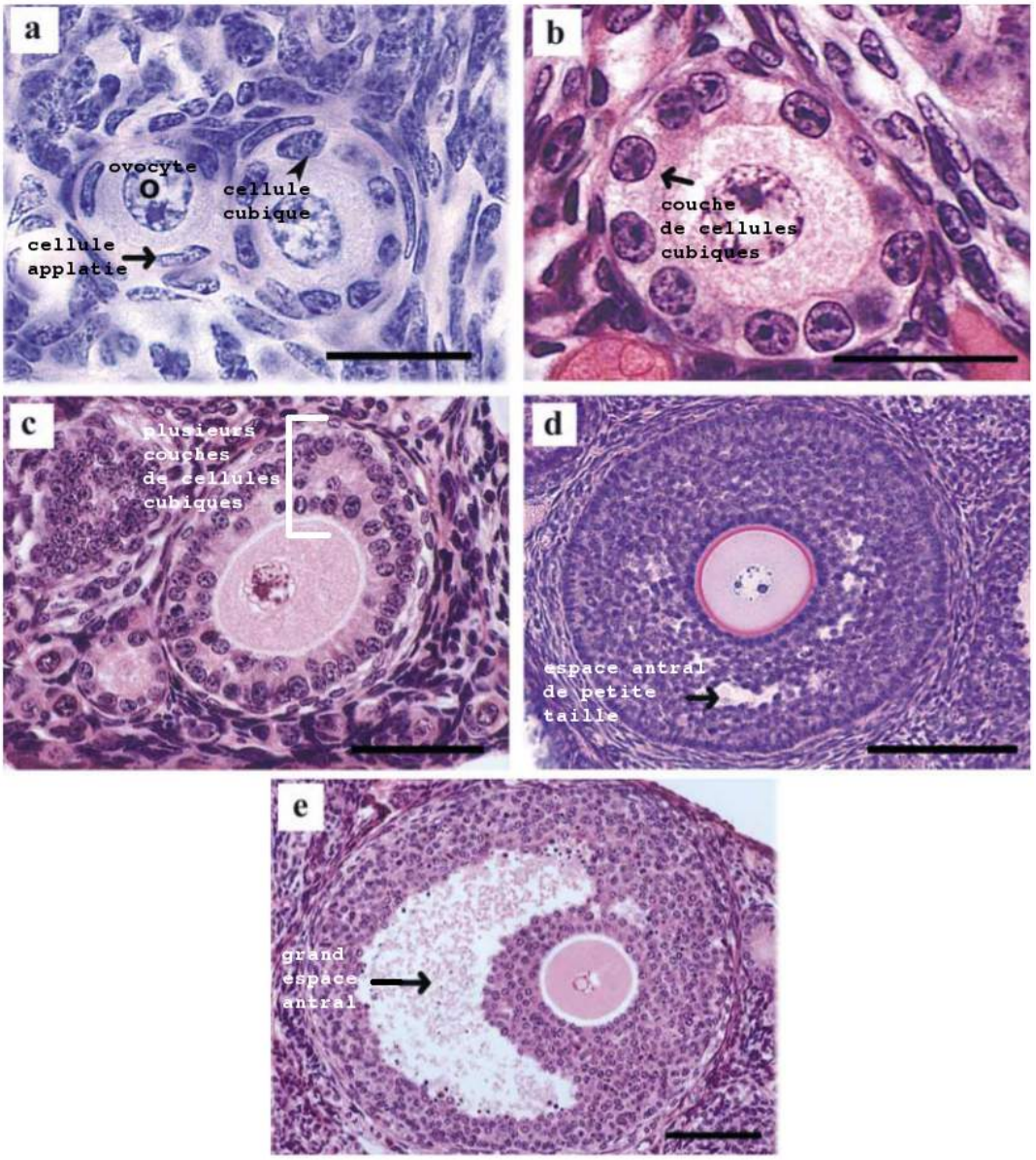


FIGURE 2.2: classification des follicules: (a) follicule primordial. (b) follicule primaire. (c) follicule secondaire. (d) follicule antral précoce avec un petit espace antral. et (d) follicule antral avec un espace plus grand. (image tirée et modifiée de l'article[4])

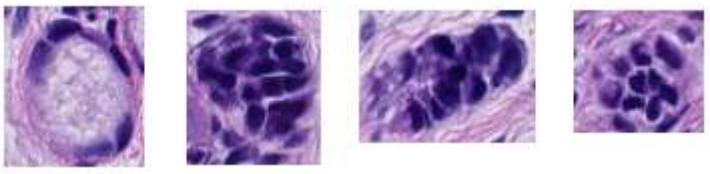


FIGURE 2.3: Variabilité des follicules primordiaux sans noyau

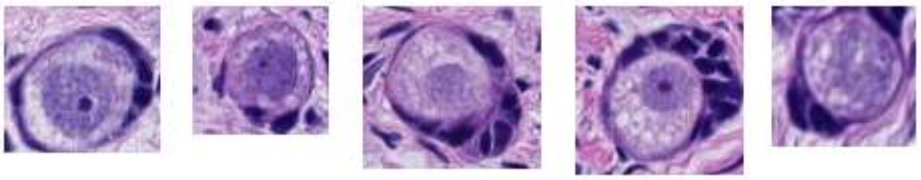


FIGURE 2.4: variabilité des follicules primordiaux avec noyau

2.3.3 Follicule primaire sans noyau

Dans la figure 2.5, sont représentés deux follicules primaires sans noyaux: des cellules cubiques sont formées autour de l'ovocyte dans la figure de gauche et le follicule est de forme ronde. À droite, plusieurs cellules cubiques sont formées, et le follicule est de forme complètement différente du précédent.

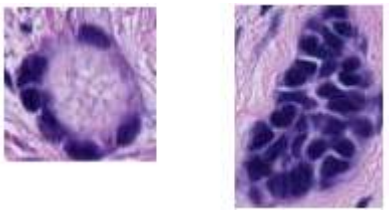


FIGURE 2.5: Deux follicules primaires sans noyau

2.3.4 Follicule primaire avec noyau

Les cellules cubiques sont un peu grandes et forment une ceinture plus épaisse, l'ovocyte et le noyau sont visibles au centre.

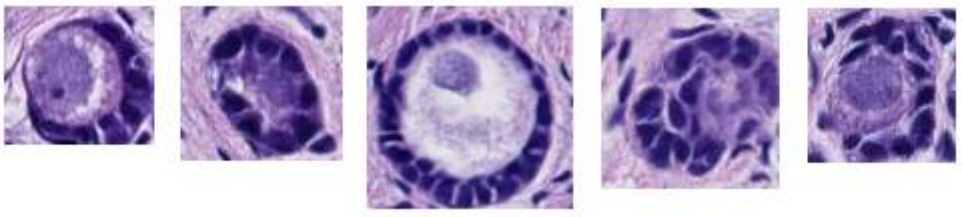


FIGURE 2.6: Des follicules primaires avec noyau

2.3.5 Follicule secondaire sans noyau

Une multicouche de cellules cubiques entourent l'ovocyte. Sa taille est plus grande.

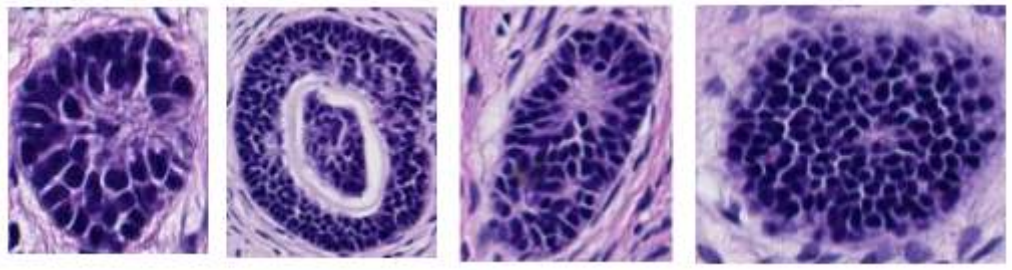


FIGURE 2.7: Follicules secondaires sans noyau

2.3.6 Follicule secondaire avec noyau

Le noyau de l'ovocyte est plus petit au centre du follicule.

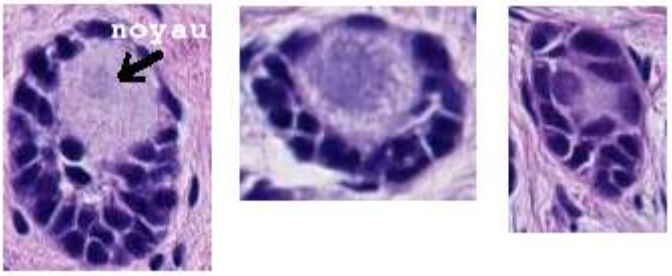


FIGURE 2.8: Follicules secondaires avec noyau

Rien qu'un simple coup d'oeil sur le jeu de données fourni par les experts pour cette étude, il n'est pas difficile de remarquer à la fois des ressemblances inter-classes entre les follicules et des différences notables (forme et texture) au sein d'un même type de follicules. A titre d'exemple, voici deux images de follicules appartenant à la même classe (primordial sans noyau) (figure 2.9):

Et voici, deux images (figure 2.10) de deux follicules qui se ressemblent mais appartenant à deux classes différentes (primordial sans noyau et primaire sans noyau respectivement):

FIGURE 2.9: deux follicules appartenant à la même classe 'primordial sans noyau'

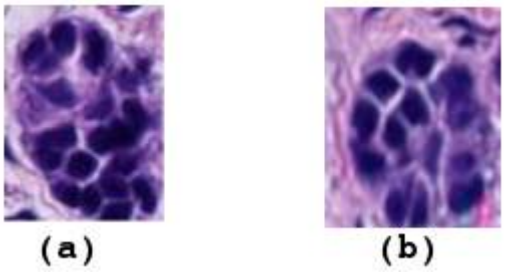


FIGURE 2.10: deux follicules appartenant à deux classes différentes: (a) primaire sans noyau (b) primordial sans noyau

Le nombre de follicules fourni par classe est résumé dans le tableau 2.1. Logiquement, on s'attend à avoir un nombre de follicules primordiaux plus élevé que les follicules primiares et secondaires, ceci est principalement dû au fait que le nombre de follicules qui vont atteindre le stade mature est en constante diminution.

Bien évidemment, les deux critères qu'on vient de lister: le nombre de follicules par classes, ainsi que les ressemblances inter-classes et les différences intra-classes vont influencer largement les résultats de la classification qu'on va établir avec notre méthode (voir chapitre résultats).

Une autre information qui peut être visualisée dans le tableau 2.1 est la taille moyenne (hauteur x largeur) en pixels des follicules (annotations) par classe. Comme on le constate, la taille d'un follicule devient de plus en plus importante au fur et à mesure qu'il passe par les différents stades de maturité.

2.4 Autres entités

Les vaisseaux constituent une autre entité qui est fortement présente dans les lames. On peut distinguer les vaisseaux par la présence d'une lumière (un trou blanc) en leur sein. Ils sont de différentes tailles. Tandis que la région du cortex

		Taille (pixels)	
Classe	Nombre	hauteur	largeur
Primordial sans noyau	175	62.58	62.97
Primordial avec noyau	248	70.10	66.87
Primaire sans noyau	120	86.17	84.17
Primaire avec noyau	146	92.58	90.96
Secondaire sans noyau	09	220.78	192.56
Secondaire avec noyau	03	110.33	93.33

TABLE 2.1: Statistiques sur le jeu de données

ovarien (plus sombre) plus épaisse est très riche en follicules, les vaisseaux quant à eux occupent la majeure partie de la medula ovarienne (appelée aussi zone vasculaire) située au centre.

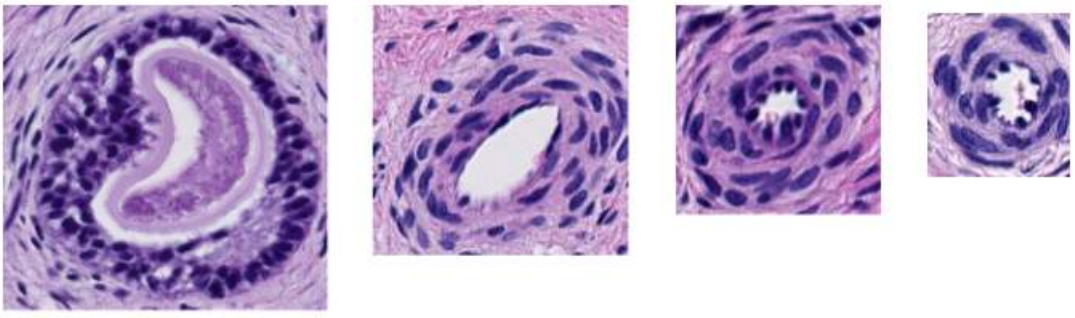


FIGURE 2.11: Les différents types de vaisseaux.

Chapitre 3

Contexte et Objectifs

Quantifier et dresser une classification de la réserve en follicules de l'ovaire est important pour résoudre les problèmes de fertilité. L'imagerie des follicules ovariens fournit des informations sur le vieillissement de l'ovaire i.e. nombre de follicules, taille, position, réponse aux stimulations hormonales [13]. Aujourd'hui, le suivi de follicules dans l'ovaire est une procédure non automatisée, et se fait avec une intervention d'un expert. L'analyse manuelle des follicules est une tâche très laborieuse et potentiellement sujette à des erreurs. Pour obtenir des résultats crédibles, un médecin doit examiner plus de 30 femmes par jour pendant toute la durée de leur cycle. Cet examen consiste en: imagerie échographique de l'ovaire dans la bonne position, mesurer chaque follicule de l'ovaire manuellement, sauvegarder les résultats, répéter ce processus pour les deux ovaires (gauche et droite) [14]. L'analyse de la zone corticale de l'ovaire qui implique la quantification et la classification des follicules afin de calculer la densité folliculaire dans l'ovaire se fait de manière manuelle [15].

3.1 Etat de l'art

Dans la littérature, les sujets traitant l'application des approches assistées par ordinateur à l'analyse des images de follicules sont rares [13]. En outre, le peu de sujets qu'on a trouvé porte sur l'analyse des images ultrasons (échographie). Dans [14], les auteurs ont proposé un algortihme qui segmente des images échographiques

de follicules par application d'une méthode de seuillage optimal après estimation grossière des limites de l'ovaire[1]. Cependant, cette méthode n'a pas donné des résultats optimaux. Les mêmes auteurs ont amélioré la méthode à l'aide des contours actifs et, par conséquent, la qualité des follicules reconnus est considérablement améliorée.

Une méthode de segmentation basée sur la détection des contours (*edge based method for segmentation*) a été utilsée dans [16] pour la détection des follicules. Cette méthode utilise un filtre passe bàs gaussien pour la suppression du bruit dans les images ultrasons d'ovaires comme étape de prétraitement. Des opérations de dilatations et d'érosions morphologiques sont ensuite appliquées aux résultats. Dans [17], une méthode de segmentation à base des bassins versants est proposée. Malheureusement, à cause du bruit de speckle[2] qui prévale sur les images ultrasons (échographie), la détection des coutours de l'objet est difficile et conduit ainsi à une mauvaise segmentation.

3.2 Détection et quantification des objets

La détection suivie de la quantification des objets cibles présents dans des lames histologiques est une tâche dure. L'application d'une technique d'apprentissage directement sur les images sans aucun prétraitement de celles ci (construction du modèle, et prédiction des nouveaux exemples) afin de réaliser la détection est rarement appliquée dans la littérature, mais semble donner de bon résultats. Dans ce contexte, *DanC.Ciresan* et *al.* dans [18] ont appliqué la technique des réseaux de neurones DNN directement sur des images histologiques afin de détecter et quantifier les mitoses. Les DNN sont utilisés comme des classifieurs de pixels en (mitose/non mitose). Cette approche a remporté la compétition $ICPR$ 2012 *mitosis detection*[3] en réalisant des scores de (précision, sensitivit, F-measure) = (0.88, 0.70, 0.782) [18]. Dans [1], les auteurs ont appliqué la technique des machines à vecteur de support SVM sur des images histologiques dans le but de détecter et quantifier les noyaux des cellules. La méthode utilise les valeurs

[1] après quoi, on aura une sous-image contenant uniquement l'ovaire dans laquelle on cherche les follicules

[2] Le speckle est un bruit particulier que l'on retrouve dans toutes les images sonar

[3] $ICPR$: est une compétition internationale pour la reconnaissance des formes organisée sous forme de challenge. Les organisateurs de l'édition 2012 sont: IPAL UMI CNRS - TRIBVN - Pitié-Salpêtrière Hospital - The Ohio State University . http://ipal.cnrs.fr/ICPR2012/

des pixels et les degrés d'inclusions des contours (*edge value*)[4] des noyaux pour entrainer le classifieur. Les résultats montrent une détection remarquable et une bonne performance de généralisation. La précision atteinte est de 90% [1].

3.3 Objectifs

Toutes les méthodes de détection des follicules citées ci-dessus ont opéré sur des images ultrasons échographiques (à l'exception de [15]). Notre travail consiste en développement d'un outil qui opére sur des images de lames histologiques de l'ovaire dans l'objectif de détecter, quantifier, et de classifier automatiquement les follicules dans les différentes classes exposées précédemment. L'outil se base sur la théorie de l'apprentissage supervisé, en l'occurence la méthode des $Extra-trees$ avec extraction aléatoire des sous-fenêtres exposée dans [2].

Les problèmes de quantification et de classification des objets (follicules) dans les images se ramènent au problème de détection: comment localiser un follicule dans l'image ? Cette localisation devient plus problématique dans le cas des objets agglutinés/touchés qui exige une séparation afin de les quantifier [1].

Dans un premier temps, nous essayerons de faire une détection et quantification automatiques de ces objets, avant de faire une classification. La méthode doit parcourir la lame et repèrer les follicules (ce qui implique leurs quantification), et ensuite les classer.

La difficulté de la tâche de détection (quantification) réside dans le positionnement des follicules dans les images. En effet, dans l'ovaire, les follicules sont positionnés soit individuellement (follicules individuels espacés), soit en amas (des follicules positionnés les uns aux cotés des autres pour former des amas) (figure 3.1). Le dernier cas, est majoritaire. Pour la classification, la grande variabilité des follicules au sein d'une même classe, ainsi que la grande ressemblance des follicules appartenant à différentes classes constituent les deux grands obstacles de cette partie.

Ce travail s'inscrit dans le contexte de développement de la plateforme logicielle du projet Cytomine[5]. Le projet Cytomine est une platefome multifonctions utilisant

[4] Cette valeur est calculé par l'opérateur de Laplace

[5] Cytomine est un projet financé par la Wallonie (programme WIST3 de la DGO6). http://www.cytomine.be/

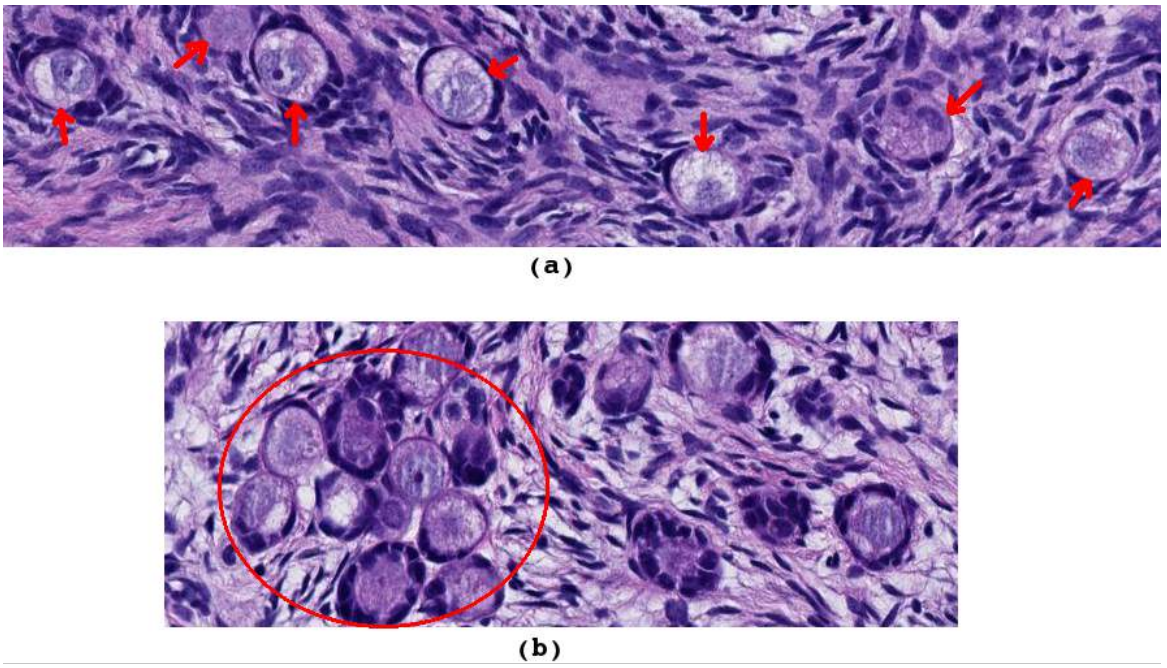


FIGURE 3.1: positionnement des follicules dans une image: (a) follicules individuels espacés (flèches) (b) amas de follicules entourés par le cercle rouge

les nouvelles technolgies web et regroupant différents outils open-source, ainsi que des algorithmes génériques. Le projet a pour but la visualisation et l'annotation collaborative des lames digitales. L'annotation consistera à définir une région de la lame et à déterminer son appartenance à une classe [19].

Les annotations (exemple d'annotations dans la figure 3.2) réalisées dans le cadre de ce projet sont faites par les chercheurs[6]. Elles ont été réalisées sur quelques images parmi un ensemble de 24 images[7] téléchargées sur le serveur de la plateforme. Ces images sont multidimensionnelles dont les largeurs varient de [65280-88320] pixels et les hauteurs entre [38400-43520] pixels, et dont la résolution est de 0.45 μm/*pixel*.

[6]les échantillons ont été collectés, les lames ont été préparées, les annotations ont été réalisées par Fransolet Maité -Laboratoire de Biologie des Tumeurs et du Développement- LBTD Université de Liège

[7]les images au format *.ndpi* ont été acquises par le microscope Hammamatsu, elles sont ensuite converties au format *.svs*

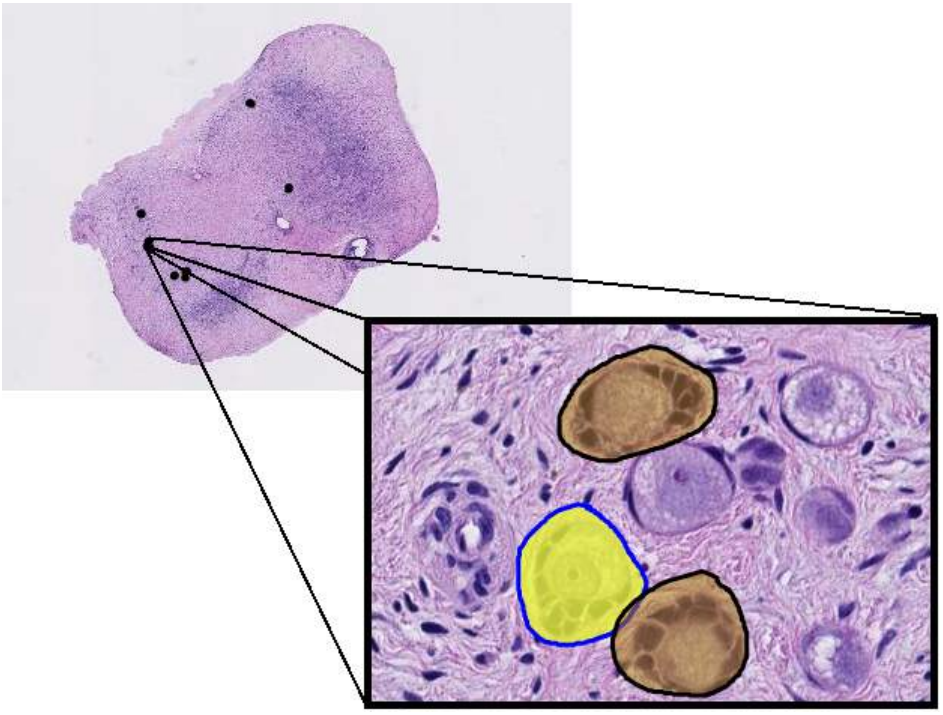


FIGURE 3.2: Exemple d'annotations d'expert sur une lame

Chapitre 4

L'ordinateur peut apprendre

Le travail qu'on va réaliser s'inscrit dans le cadre de l'apprentissage automatique appliqué dans le cas des images de microscopes. C'est pourquoi, on préfère commencer par donner des notions de représentation des images. Nous parlerons ensuite des concepts et techniques de l'apprentissage supervisé, incluant les arbres de décision et les ensembles d'arbres. La présentation de la méthode appliquée dans ce travail en l'occurence la méthode des arbres extrêmement aléatoires ($Extra-trees$) avec extraction des sous-fenêtres aléatoires expliquée dans [2],[7] et [3] est exposée dans le chapitre 5. L'objectif étant de construire un modèle à base des annotations (images) fournies suivant cette méthode , et ensuite de classer de nouvelles images.

4.1 Une image est une matrice de pixels

Les images traitées dans ce travail sont de formes numériques. Du point de vue informatique, une image numérique est un tableau de valeurs. Chaque case de ce tableau, qui stocke une valeur se nomme un pixel. En notant n le nombre de lignes et p le nombre de colonnes de l'image, on manipule ainsi un tableau (ou une matrice) de n x p pixels. Les valeurs des pixels sont enregistrées dans l'ordinateur sur un certain nombre de bits sous forme de nombres entiers, par exemple entre 0 et 255 dans le cas des images en niveaux de gris 8 bits, ce qui fait 256 valeurs possibles pour chaque pixel. Dans le cas des images en niveau de gris, la valeur d'un pixel représente la luminance: La valeur 0 correspond au noir, et la valeur 255 correspond au blanc, tandis que les valeurs intermédiaires correspondent à des niveaux de gris allant du noir au blanc. Une image couleur RGB est en réalité

composée de trois canaux afin de représenter le rouge, le vert, et le bleu. Chaque pixel d'une image couleur comporte ainsi trois nombres (r,g,b), chacun étant un nombre entier entre 0 et 255. Si le pixel est égal à (r,g,b)=(255,0,0), il ne contient que de l'information rouge, et est affiché comme du rouge. De façon similaire, les pixels valant (0,255,0) et (0,0,255) sont respectivement affichés vert et bleu.

4.2 Apprentissage automatique supervisé

L'apprentissage automatique (*machine learning*), qui est l'un des sous-domaines de l'intelligence artificielle, a pour objectif d'extraire et d'exploiter automatiquement l'information présente dans un jeu de données (images, textes, paroles, etc.). Il consiste en l'élaboration d'algortihmes et de méthodes qui ont la faculté d'apprendre à effectuer une tâche à partir de l'observation d'un environnement. Appliqué à des problèmes de classification, il s'agit d'apprendre à distinguer entre eux divers élements de cet environnement par l'observation d'exemples.

Dans ce sens, les algorithmes d'apprentissage automatique sont couramment utilisés pour construire des classifieurs permettant de distinguer les exemples. Le défi considérable de l'apprentissage automatique est de développer une méthode capable de résoudre éfficacement des problèmes de classification de natures diverses. La qualité d'une méthode d'apprentissage dépend donc de sa capacité à construire un classifieur qui généralise le phénomène observé. L'apprentissage est dit supervisé s'il utilise des exemples étiquetés ou classés. Ces étiquettes ou ces classes peuvent être vues comme fournies par un expert ou un superviseur. Le but de l'apprentissage est alors de produire une fonction de classification, appelée hypothèse, permettant de déterminer la classe d'un exemple.

4.2.1 La classification par apprentissage supervisé

En apprentissage automatique, un problème de classification se présente sous la forme d'un ensemble de données, contenant des exemples issus de l'observation d'un phénomène, cet ensemble est appelé ensemble d'apprentissage. Chaque exemple de cet ensemble est constitué d'une description (attributs) et d'une étiquette (classe). Un algorithme d'apprentissage analyse ces données afin de construire un

modèle (classifieur). C'est à ce modèle que revient ensuite la tâche d'étiqueter de nouveaux exemples à partir de leur description.

Soit un problème de classification avec un ensemble d'apprentissage de N exemples, et un ensemble de M classes: si X^i $(i = 1, ..., N)$ est un vecteur d'attributs d'entrée décrivant l'exemple i, et $Y^i \in \{1, ..., M\}$ sa classe, on dénote l'ensemble d'apprentissage par:

$$LS = (X^i, Y^i), i = 1, ..., N$$

Dans notre cas, les éléments ${X_k}^i$ $(k = 1, ..., m)$ de X^i sont des valeurs numériques appartenant à $0, ...255$.

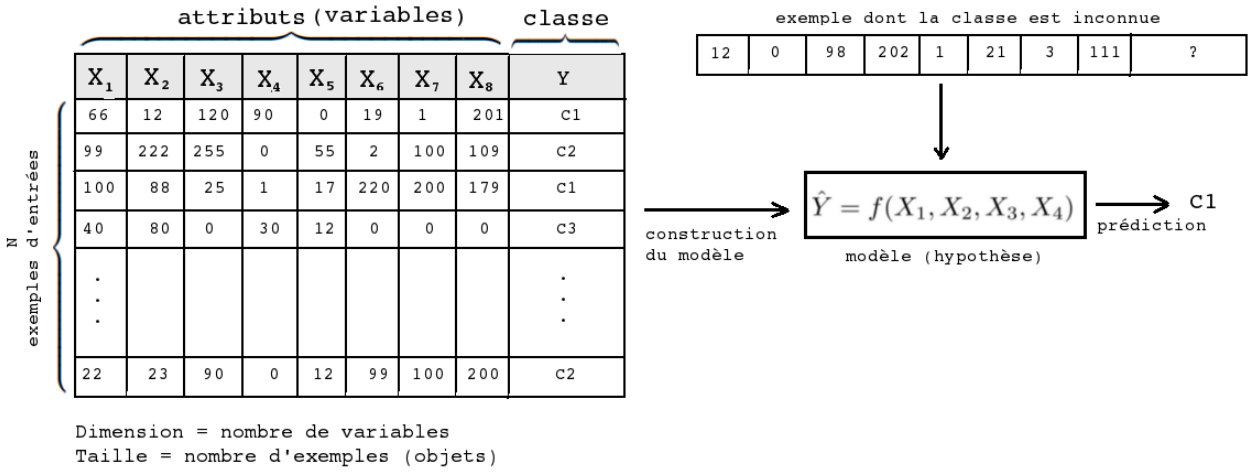


FIGURE 4.1: problème de classification par apprentissage supervisé à 3 classes: les exemples de l'ensemble d'apprentissage sont décrits par 8 attributs à valeurs numériques et une classe de sortie. Après la construction du modèle (hypothèse), on désire prédire la classe de l'exemple dont la classe est inconnue.

Après la construction du modèle, Une méthode pour vérifier la généralisation du modèle au delà de LS est l'utilisation d'un échantillon de test contenant des exemples distincts de ceux de LS. La précision du modèle est ainsi exprimée par son taux d'erreur sur cet échantillon de test.

4.2.2 Sous-apprentissage et sur-apprentissage

La précision d'un modèle peut être influencée par deux sources d'erreur. D'une part, un modèle doit être suffisamment représentatif des cas d'apprentissage et exploiter au mieux l'information contenues dans ces données [6]. On parle de *sous* −

apprentissage si le modèle construit est trop simple i.e. la fonction considérée par l'algorithme d'apprentissage n'est pas assez riche pour pouvoir décrire la diversité présente dans les données (figure 4.2.a). On exprime l'erreur dans ce cas par le biais[1]. D'autre part, quand on cherche à trop "coller" aux donneés d'entraînement LS (apprentissage par coeur), on se retrouve dans le cas de $sur - apprentissage$ (figure 4.2.b). Dans ce cas, il s'agit d'un apprentissage trop précis qui peut ne pas être généralisable. L'erreur dans ce cas est exprimée par la variance[2].

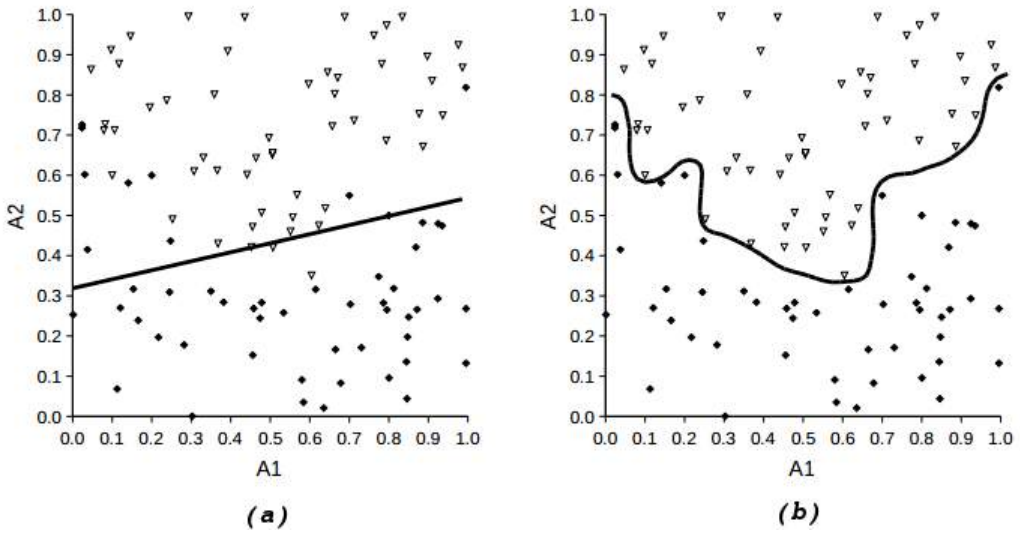


FIGURE 4.2: sous-apprentissage et sur-apprentissage: (a) modèle trop simple (b) modèle trop complexe. Figure tirée de [5]

Un bon compromis biais/variance s'impose pour un modèle d'apprentissage idéal [5]. Il reste à signaler que le biais est une fonction décroissante, tandis que la variance est généralement une fonction croissante de la compléxité (par exemple le nombre de noeuds d'un arbre de décision) du modèle comme montrée sur la figure 4.3.

Pour éviter ce genre de problèmes, on peut par exemple procéder par:

- Sélectionner ou choisir les attributs des objets qui seront utilisés
- Choisir la méthode (algorithme d'apprentissage) qui sera utilisée: arbres de décision, KNN, SVM, etc.
- Evaluer les performances du modèle obtenu, pour celà on utilise la validation croisée par exemple.

[1]Le biais est erreur systématique commise par une méthode indépendamment de l'échantillon
[2]la variance: erreur due à la variabilité du modèle (dépend de l'échantillon)

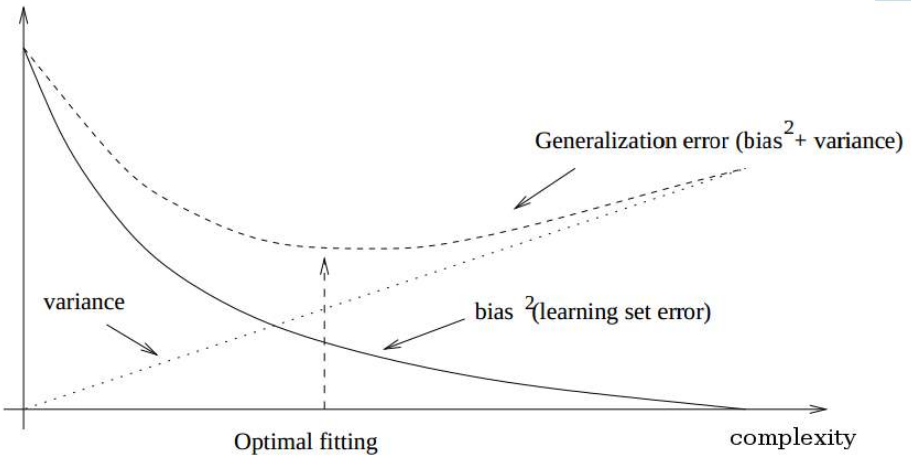


FIGURE 4.3: biais et variance en fonction de la compléxité du modèle. Figure tirée de [5]

Chacune des méthodes utilisées propose un ensemble de paramètres. Il faudra alors trouver une bonne combinaison des valeurs des paramètres pour dénicher le modèle optimal.

Dans notre cas, l'optimisation des paramètres de la méthode appliquée est montrée dans le chapitre 6.

4.3 Arbres de décision

La popularité de l'utilisation de la méthode des arbres de décision dans l'apprentissage supervisé repose en grande partie sur sa simplicité. En effet, à partir de l'échantillon d'apprentissage, une collection de règles de type "*si...alors*" est construite, donnant ainsi une structure d'arbre binaire qui est facilement interprété par un expert humain.

Il s'agit de trouver un partitionnement de l'échantillon d'apprentissage en sous-échantillons disjoints et les plus homogènes possibles du point de vue de la variable de sortie (même classe dans les problèmes de classification) pour donner une structure arborescente (figure 4.4). Un noeud de cette structure réalise simplement un test "*si*" sur un attribut d'entrée, l'issue (*alors*) de ce test est une séparation de l'échantillon d'objets en deux sous-ensembles. Les noeuds terminaux de l'arbre (feuilles) sont étiquetés soit par la classe majoritaire des objets qui ont atteints

cette feuille, soit par une distribution de probabilités des classes estimées par la fréquence de ces objets dans chaque classe [6].

En gros, un arbre de décision est un arbre où:

- chaque noeud intérieur teste un attribut,
- chaque branche correspond à la valeur d'un attribut, et
- chaque feuille est étiquetée par une classe.

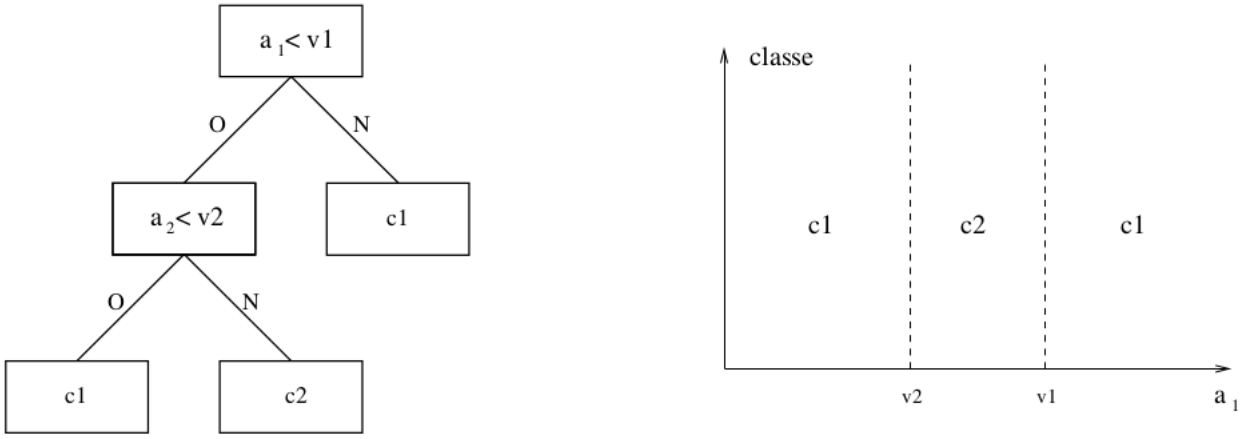


FIGURE 4.4: Arbre de décision. Figure tirée de [6]

4.3.1 Création d'un arbre de classification

La construction de l'arbre de classification à partir de l'ensemble d'apprentissage LS utilise l'approche $Top-down\ induction$ (Algorithmes 1 et 2[3]) [20]. Au départ, le noeud racine contient tous les objets de LS en entrée. Au niveau de chaque noeud de l'arbre, un test est appliqué, séparant ainsi les objets du sous-échantillon courant de façon optimale. Les échantillons résultant de cette séparation se retouvent en entrée des noeuds successeurs de ce noeud. L'algortithme continue de manière récursive sur les noeuds fils développant ainsi l'arbre. Le développement de l'arbre se poursuit jusquà ce que les feuilles de l'arbres contiennent des sous-échantillons d'objets qui ont une même classe [6].

[3]Algorithme d'apprentissage d'un arbre de décision inspiré de [6]

Algorithm 1 *learn_decision_tree*(LS)

-**Si** LS contient des objets de la même classe **Alors**:
renvoyer une feuille étiquetée avec cette classe
FIN
- **Sinon**
i- Faire $[a_k < a_{th} = test\_choice(LS)]$
ii- Diviser LS en LS_{left} et LS_{right} selon le test $[a_k < a_{th}]$, et
- construire les sous-arbres $\tau_{left} = learn\_decision\_tree(LS_{left})$
- construire les sous-arbres $\tau_{right} = learn\_decision\_tree(LS_{right})$
iii- Créer un noeud avec le test $[a_k < a_{th}]$, attacher les sous-arbres τ_{left} et
τ_{right} comme successeurs de ce noeud et renoyer l'arbre résultant.

Algorithm 2 *test_choice*(LS)

- Sélectionner un attribut a_k et un threshold a_{th} qui maximise la mesure de score calculée sur LS (meilleur attribut)
- FIN

La mesure de score utilisée devra favoriser la séparation en classes afin de réduire la profondeur de l'arbre. Une mesure commune est l'impureté qui devra être:

- minimale, si l'ensemble des objets est homogène (i.e. de même classe)
- maximale, si les objets sont uniformément répartis entre les classes

Exemples de mesures d'impureté (p_j: les probabilités de classes):

- l'entropie de *Shanon*: $H(LS) = \sum_j p_j log2p_j$. Elle mesure l'incertitude
- l'indice de *Gini*: $I(LS) = \sum_j p_j(1 - p_j)$

La meilleure séparation sera celle qui maximise la différence entre l'impureté de l'ensemble initial et la somme des impuretés de ces partitions pondérées par les tailles des partitions [21].

4.3.2 Prédictions

Une fois le modèle construit, on peut inférer la classe d'un nouvel objet en le faisant propager à travers les tests (neouds) de l'arbre de haut en bas. À chaque test, l'objet se trouve dirigé vers un des successeurs du neoud courant jusqu'à ce qu'il atteint un noeud terminal. La classe attribuée à cet objet sera dès lors celle qui est majoritairement attribuée à cette feuille lors de la phase de l'apprentissage.

4.3.3 Les critères d'arrêt

Dans l'algorithme ci-dessus, on voit que le critère d'arrêt correspond au fait que tous les objets de LS appartiennent à la même classe, ceci favorise la construction des arbres ayant des feuilles avec peu d'objets (arbre profond) et diminue la fiabilité de l'arbre lors de la classification. Par conséquent le taux d'erreurs augmente. On conclut que ce critère n'aide pas à construire des arbres plus simples.

D'autres critères d'arrêt peuvent être utilisés:

- La profondeur de l'arbre atteint une limite fixée (=nombre d'attributs utilisés)
- Le nombre de feuilles atteint un maximum fixé
- Le nombre d'instances par noeud est inférieur à un seuil fixé
- Le gain d'information maximum obtenu est inférieur à un seuil fixé
- La qualité de l'arbre n'augmente plus de façon sensible. Par exemple, aucun attribut n'améliore la qualité de l'arbre

Avec ces critères, la construction de l'arbre est stopée lorsqu'un seuil est atteint, Ces critères se situent dans le cadre de l'élagage et plus précisément dans le pré-élagage des arbres.

4.3.4 Le sur-apprentissage et l'élagage des arbres de décision

Lors de la construction d'un arbre, la taille de l'arbre grandit de manière linéaire avec la taille de l'ensemble d'apprentissage LS. De plus, un sur-apprentissage peut se produire lorsque cet ensemble contient des données bruitées, ou qu'il ne contient pas certains exemples importants, ou encore lorsque les exemples sont trop spécifiques.

Pour éviter ces problèmes, l'élagage de l'arbre s'impose comme solution.

L'élagage d'un arbre consiste à simplifier l'arbre par suppression de quelques branches. On distingues deux approches: pré-élagage et post-élagage:

- **Pré-élagage**: il a pour but d'arrêter la construction de l'arbre de décision à l'avance même si les feuilles ne sont pas pures

- **Post-élagage**: Dans cette approche, l'arbre de décision est d'abords construit puis simplifié en supprimant un ou plusieurs de ses sous-arbres et en les remplaçant par des feuilles

4.4 Ensemble d'arbres

L'inconvénient des arbres de décision est leur instabilité. En effet, lors de la construction, il suffit de choisir un attribut plutôt qu'un autre (surtout s'il est proche de la racine) pour avoir une autre construction. La conséquence de cette instabilité est que les algorithmes d'apprentissage par arbres de décision ont une variance importante (figure 4.5) qui nuit à la qualité d'apprentissage.

L'idée derrière les méthodes d'ensemble est de construire plusieurs modèles (arbres différents) à partir du même ensemble d'apprentissage. Lors de la prédiction, l'objet est propagé à travers les différents arbres. La prédiction d'un objet est obtenue par une procédure de vote, ou par une moyenne des prédictions.

La prédiction résultante de l'agrégation est nettement plus stable, i.e. de variance plus faible (figure 4.5) que la prédiction d'un seul modèle.

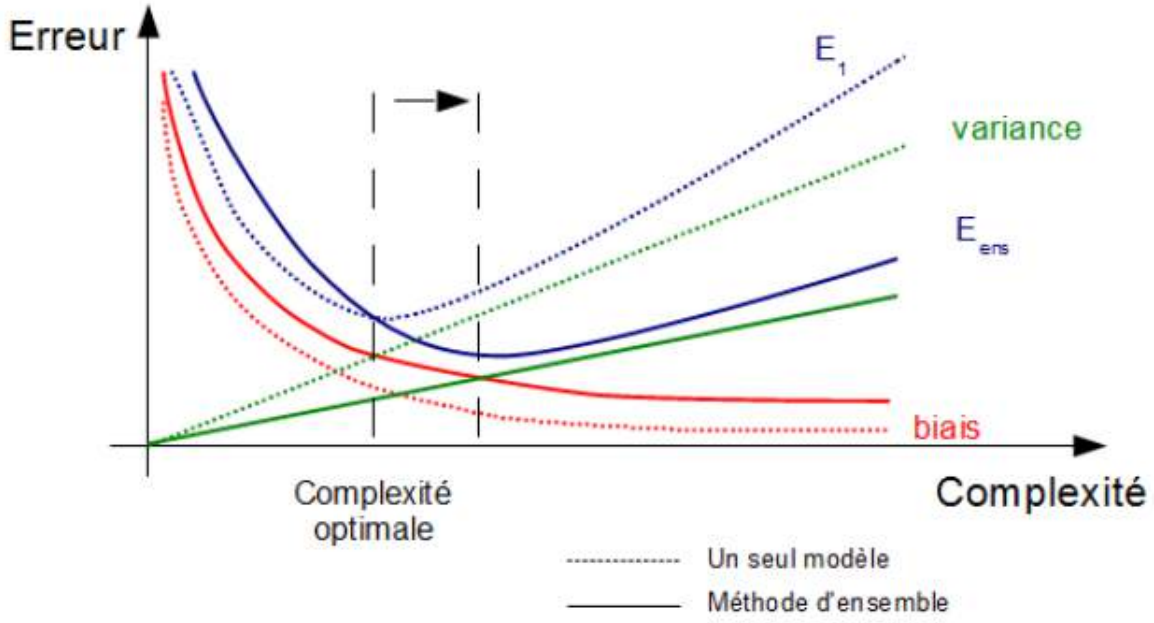


FIGURE 4.5: comparaison variance et erreur entre un seul modèle et un ensemble de modèles

4.5 Les machines à vecteurs de support SVM [1]

Il s'agit de l'une des techniques de l'apprentissage automatique la plus utilisée et la plus performante. Pour les problèmes de classification, un classifieur SVM procède par déterminer le meilleur hyperplan de séparation (figure 4.6) par maximisation de la distance de l'hyperplan aux points les plus proches parmis ceux de l'échantillon d'entraînement (on parle de marge maximale). Ces points sont les vecteurs de support, d'où l'appellation.

La fonction de classification d'un classifieur SVM est de la forme:

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b)$$

où:
- x est le point à classer
- x_i sont les vecteurs de support
- N le nombre de vecteurs de support
- b: une constante
- $y_i \in \{-1, 1\}$ est la classe du vecteur de support x_i
- les coéfficients α_i sont la solutin d'un problème d'optimisation quadratique
- K: c'est le noyau (*kernel*) du classifieur. Il existe plusieurs types de noyaux: linéaire, polynomial, sigmoid, *etc.*

La marge qui est la distance de l'hyperplan aux vecteurs de support est donnée par:

$$M = \frac{1}{\sqrt{\sum_{i=1}^{N} \alpha_i}}$$

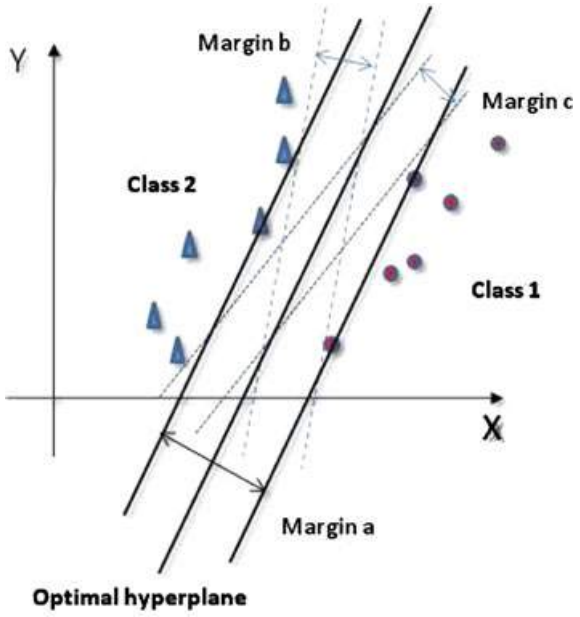


FIGURE 4.6: Hyperplan optimal pour un classifieur SVM

Chapitre 5

Les $Extra-trees$ avec tirage aléatoire de sous-fenêtres

Dans ce chapitre, nous détaillerons la construction d'ensemble d'$Extra-trees$ (les arbres extrêmement "randomisés"). Les arbres sont construits sur tout l'ensemble d'apprentissage. La randomisation proposée est la suivante [7]:

- Lors du développement d'un noeud, on génère k tests aléatoirement, et on prend le meilleur test parmi ceux-ci.
- pour générer un test aléatoire, on choisit un attribut aléatoirement: s'il est numérique, on choisit un seuil de discrétisation au hasard (entre le min et le max de l'attribut dans l'échantillon), s'il est symbolique, on choisit un sous-ensemble aléatoire des valeurs possibles de l'attribut
- on développe l'arbre complètement

Le paramètre k controle le degré de randomisation.

5.1 Les $Extra-trees$ avec extraction de sous fenêtres aléatoires [2],[3]

C'est la méthode utilisée dans ce travail[1], c'est pourquoi nous essayons de nous étaler un peu dans sa description. La méthode est basée sur la construction d'un

[1] plus précisément la variante avec la couche SVM

ensemble d'arbres randomisés avec extraction de sous-fenêtres aléatoirement à partir des images. Elle propose également un tas de paramètres contrôlant la méthode qu'on va aborder dans les lignes suivantes.

5.1.1 Phase d'apprentissage

Etant donné un échantillon d'apprentissage d'images étiquetées LS. La première étape de la méthode consiste à extraire des zones rectangulaires appelées $sous - fenetres$ des images de l'échantillon, on aura ainsi construit un nouvel échantillon d'apprentissage LS_w composé de sous-fenêtres. L'ensemble des attributs de chaque sous-fenêtre est constitué de tous les pixels la décrivant.

La variable de sortie (classe) d'une sous-fenêtre se voit assignée la classe de l'image à partir de laquelle elle a été extraite. Le modèle de classification est ensuite construit sur la base de cet échantillon.

L'extraction aléatoire des sous-fenêtres est réalisée de manière équilibrée, i.e. pour chaque image de LS, on extrait un total de $N_w = N_{ls}/N_{img}$, où: N_{ls} représente le nombre total de toutes les sous-fenêtres qu'on devra extraire (= taille du nouvel échantillon LS_w), et N_{img} est la taille de l'échantillon d'images LS.

La taille des sous-fenêtres à extraire étant fixée par un paramètre w (après réduction de la taille). On exige que la sous-fenêtre sélectionnée soit complètement incluse à l'intérieur de l'image (i.e. sans coupure aux bords). Les coordonnées d'une sous-fenêtre extraite ne sont pas mémorisées, ce qui implique qu'une même sous-fenêtre peut se présenter en plusieurs copies dans l'échantillon d'apprentissage. La taille des sous-fenêtres, ainsi que le paramètre N_w influencent la zone couverte par l'extraction sur l'image.

Un exemple d'extraction de sous-fenêtres d'une taille 16x16 à partir d'une image d'un follicule est illustré dans la figure 5.1.

L'algorithme d'apprentissage est brièvement le suivant (algorithme 3):

Durant la construction d'un arbre comme elle est décrite dans [2], un test associé à un noeud interne de l'arbre consiste en une simple comparaison de la valeur d'un pixel choisi aléatoirment dans une sous fenêtre à une valeur de coupure. Le

FIGURE 5.1: quelques sous-fenetres de 16x16 extraites à partir d'une annotation d'un follicule secondaire

Algorithm 3 *learn_Extra_trees*(LS_w)

1- **Initialisation**:

T < − nombre d'arbres aléatoires

w < − taille des sous-fenêtres

N_w < − nombre total de sous-fenêtres à extraire (taille de LS_w)

N_{img} < − nombre d'images de LS

min_size (%) < − taille minimale de la sous-fenêtre (portion de l'image)

max_size (%) < − taille maximale de la sous-fenêtre (portion de l'image)

2- **Pour** chaque image de LS **faire:**

i- extraire N_w/N_{img} sous-fenêtres aléatoirement dont la taille de chacune

$\in [min\_size, max\_size]$ de $min(largeur\_image, hauteur\_image)$

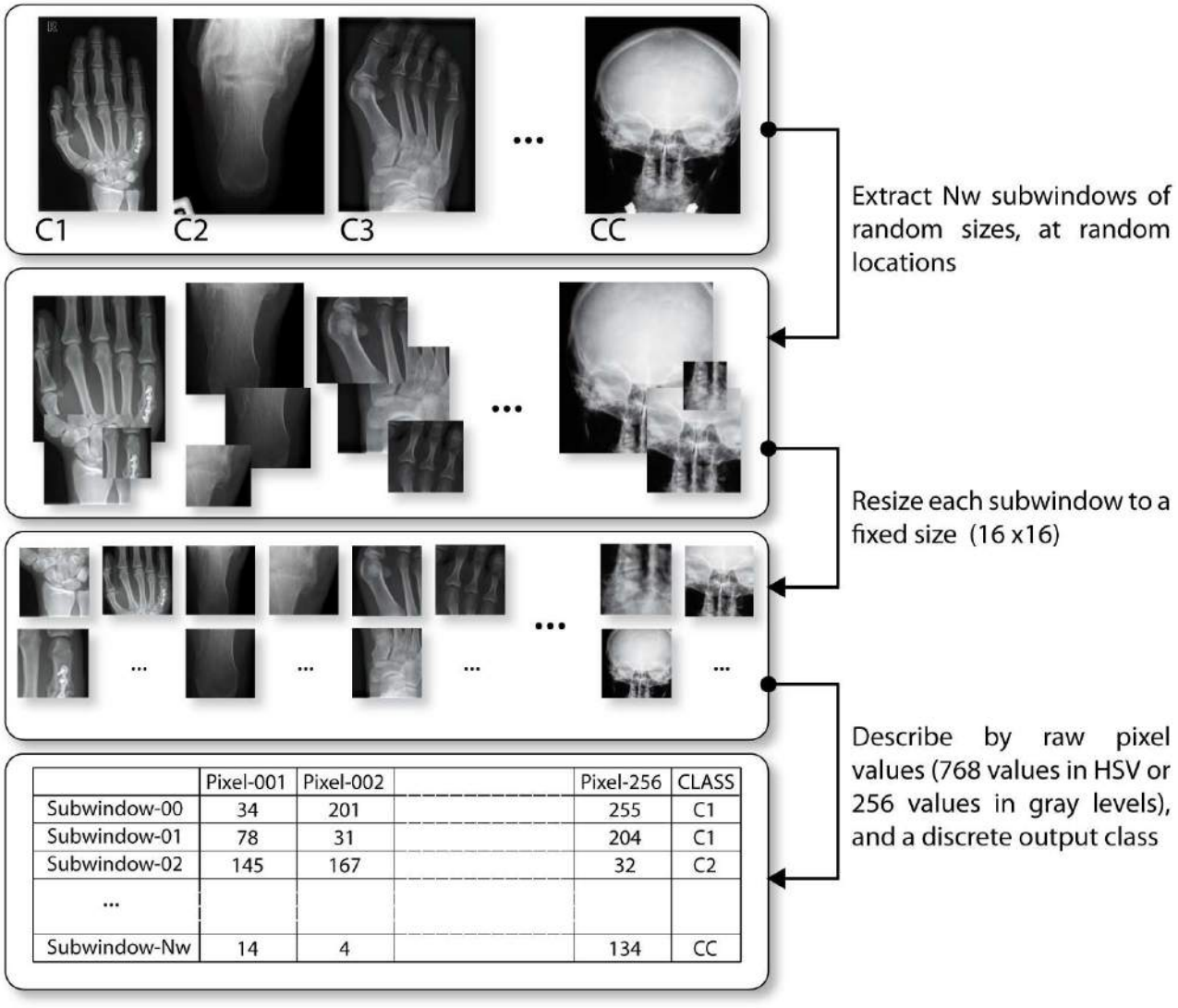
ii- réduire la taille de chaque sous-fenêtre à w x w

iii- attribuer à chaque sous-fenêtre la classe de l'image

3- Construire un ensemble de T arbres aléatoires à partir du nouvel échantillon LS_w

développement d'un noeud est stoppé dès que le nombre de sous-fenêtres atteignant ce noeud est plus petit qu'une valeur d'un seuil prédéfini (n_{min}). À ce stade, le noeud devient une feuille de l'arbre. Dès lors, un vecteur d'une taille de M (M: nombre de classes) contenant des fréquences de classes est alors calculé en fonction des sous-fenêtres qui ont atteint ce noeud (figure 5.3.a).

La complexité de l'algorithme de construction d'un arbre aléatoire est liée au nombre d'images (ici de sous-fenêtres) dans l'échantillon d'apprentissage, elle est de l'ordre de $O(N_w log N_w)$ [6].



	Pixel-001	Pixel-002		Pixel-256	CLASS
Subwindow-00	34	201		255	C1
Subwindow-01	78	31		204	C1
Subwindow-02	145	167		32	C2
...					
Subwindow-Nw	14	4		134	CC

FIGURE 5.2: Extraction aléatoire des sous-fenêtres et construction de la matrice d'apprentissage. Figure tirée de [7]

5.1.2 Phase de prédiction

Après l'extraction de sous-fenêtres d'une taille w x w soit par un parcours exhaustif avec décalage d'un pixel horizontalement et verticalement pour avoir toutes les fenêtres possibles (comme dans [22]), soit aléatoirement (dans [7]) pour avoir un nombre N_{ts} de sous-fenêtres, ces dernières sont propagées dans le modèle construit. Les prédictions sont combinées pour avoir à la fin la prédiction de l'image initiale. Chaque sous-fenêtre se voit attribuer un ensemble de T (nombre d'arbres) vecteurs de probabilité de classe, donc TxM (M: nombre de classe). Les prédictions des sous-fenêtres sont ensuite moyennées, et la classe correspondant à la plus grande estimation de probabilité agrégée est affectée à l'image de test[2] (figure 5.3.b).

Avec un arbre aléatoire, la complexité de prédiction d'une fenêtre est de l'ordre de $logN_w$ [6].

Les étapes de base de la phase de prédiction sont montrées dans l'algorithme 4.

Algorithm 4 *predict*(*image*)

1- on extrait un nombre de fenêtres de taille w x w aléatoirement à partir de l'image
2- on propage chaque sous-fenêtre dans le modèle
3- on combine les classes attribuées aux sous-fenêtres par le modèle, et on associe la classe résultante à l'image.

5.2 Variante de la méthode incluant les machines à vecteurs de support SVM [2]

À la place de construire des vecteurs de probabilités de classes dans chaque noeud terminal (feuille) de l'arbre, dans cette variante chacun de ces noeuds va constituer un attribut d'une nouvelle matrice d'apprentissage. Le but d'utiliser ici les extra-trees est de construire cette nouvelle matrice d'apprentissage (figure 5.4.a). Cette dernière est de dimension N_I x $\#leaf$, et elle est constituée de toutes les images de l'echantillon d'apprentissage initial disposées en lignes, et en colonnes tous les noeuds terminaux issus des T arbres utilisés. Suivant ce schéma, chaque image est donc décrite par un vecteur d'attributs. Les éléments de cette matrice sont soit:

[2]dans [6], on parle aussi d'addition des vecteurs de probabiltés à M classes, et ensuite choisir la maximum du vecteur résultat

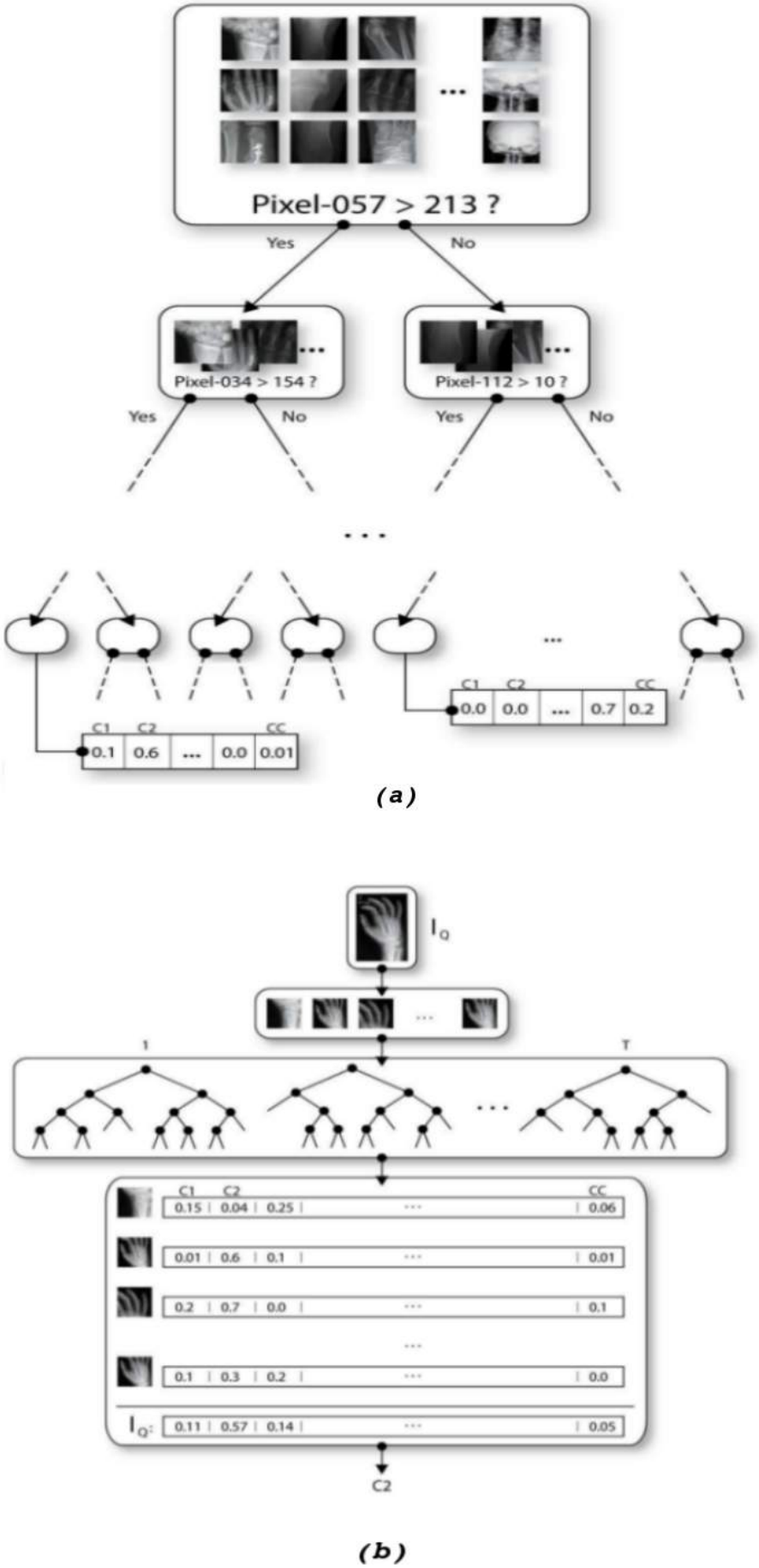


FIGURE 5.3: (a). Construction d'un arbre (b). Prédiction au travers d'un ensemble d'arbres. Figure tirée de [2] (supplementary data)

1. des valeurs binaires (0,1)
2. des fréquences

indiquant respectivement, la présence d'au moins une sous-fenêtre issue de l'image $image_i$ dans le noeud ($leaf_j$), et la proportion de sous-fenêtres issues de l'image $image_i$ atteignant le noeud $leaf_j$.

De là, on peut comprendre que dans cette approche, on utilise les Extra-trees uniquement pour produire une autre représentation des images de l'échantillon initial.

Cette matrice d'apprentissage est utilisée pour entrainer un classifieur SVM (une couche SVM est ajoutée au modèle des extra-trees initial).

Pour la prédiction (figure 5.4.b) de la classe d'une nouvelle image, les sous-fenêtres extraites de cette image sont propagées aux travers des arbres du modèle des Extra-trees pour produire le vecteur d'attributs de l'image, ce dernier est ensuite soumis au classifieur SVM pour prédire la classe de l'image.

Cette variante est celle utilisée dans ce travail.

5.3 Paramètres contrôlant la méthode [2]

Les paramètres les plus pertinents de la méthode sont en nombre de cinq. Leurs influences sur les résultats obtenus mesurés par l'erreur de classification sont montrées sur les graphiques de la figure 5.5. Ces graphiques ont été réalisés sur un ensemble de 80 bases de données publiques contenant des images différentes. Ils nous permettent de dégager la meilleure combinaison de ces paramètres pour construire un modèle optimal.

Le paramètre le plus important est l'intervalle des tailles (en %) des sous-fenêtres extraites à partir de l'image. Le choix de ce paramètre dépend du problème traité. Une petite taille des sous-fenêtres permet de capturer les détails fins, et c'est ce qui est recommendé dans le cas des images à texture répétée. Le problème de classification des images histologiques s'inscrit dans ce cas.

La taille du vecteur descripteur de 16x16 représente les attributs des sous-fenêtres de la matrice d'apprentissage en pixels (figure 5.2). La valeur de 16x16 (dans le

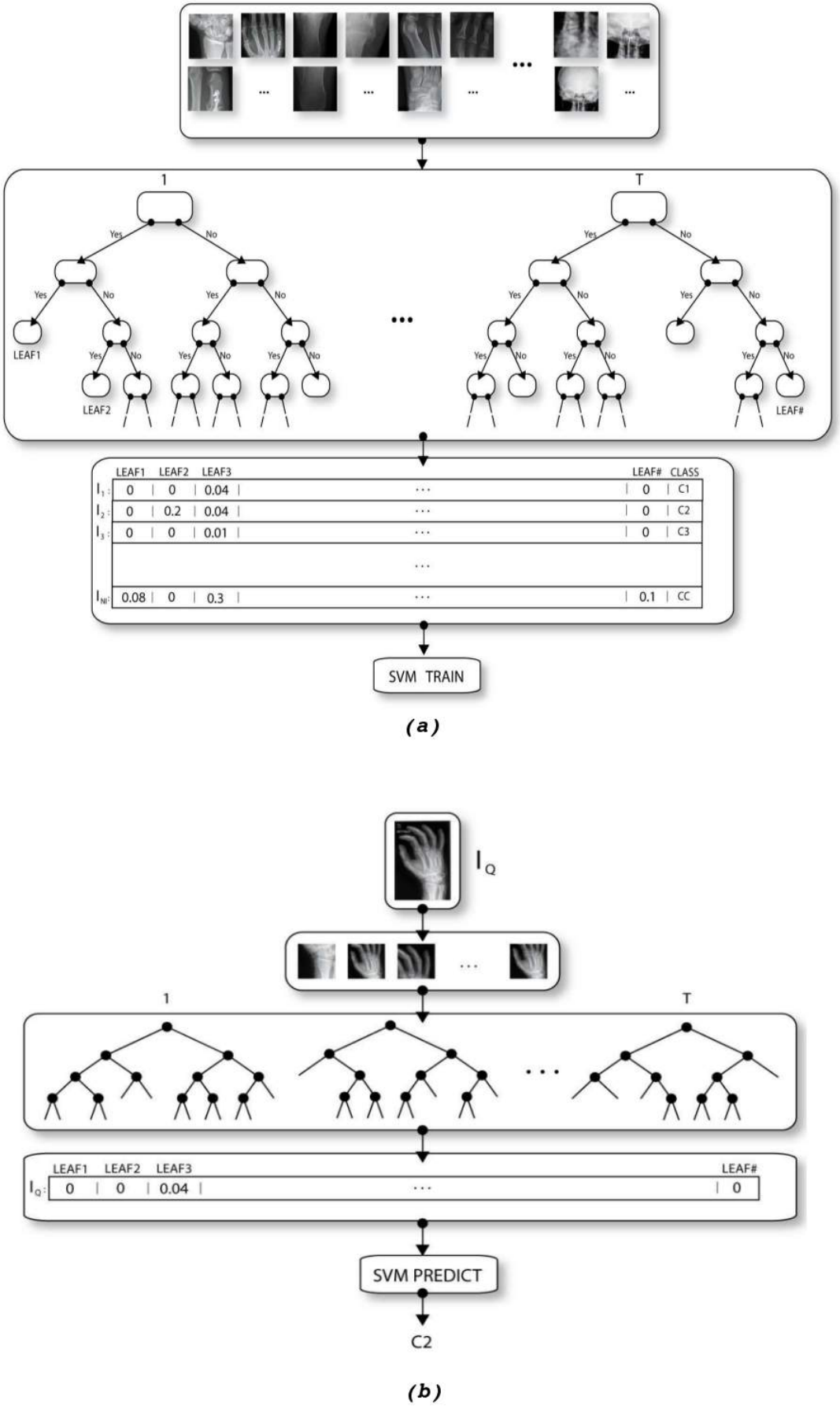


FIGURE 5.4: Méthode incluant la couche SVM (a). Construction des vecteurs d'attributs pour entrainer le classifieur SVM. (b). Prédiction d'une nouvelle image. Figure tirée de [2]

mode couleur RGB, cette valeur nous donne un vecteur descripteur d'une taille de 16x16x3 = 768) des sous-fenêtres donne la plus petite erreur, et semble être la meilleure valeur pour ce paramètre (figure 5.5).

Le nombre de sous-fenêtres extraites N_w est un autre paramètre important de cette méthode (pas de graphique). $N_w = 1\ million$ donne de bons résultats [2].

Comme on l'a vu ci-dessus, la variance diminue avec l'utilisation d'un ensemble d'arbres qu'avec un seul arbre, par conséquent le paramètre T doit être bien choisi pour la construction d'un modèle optimal. Sur les graphiques de la figure 5.5, la valeur $T = 40$ semble donner en moyenne le taux d'erreur le plus faible.

Le nombre de tests aléatoires K permet de filtrer les attributs sans importance. Le taux d'erreur présente une allure décroissante en fonction de K. Par conséquent, plus on grandit la valeur de K, plus on obtient de bons résultats.

Le dernier paramètre est $nmin$ qui représente le critère d'arrêt de l'algorithme de construction du modèle i.e. la taille minimale d'un ensemble pour créer un noeud. Ce paramètre doit être choisi grand car les arbres de la couche *Extra-trees* doivent être élagués afin de construire des attributs qui ne soient pas trop "collés" aux images de l'échantillon d'apprentissage (ie. trop spécifiques) [2].

Pour un nombre de sous-fenêtres de 1 million, le choix de $nmin = 500$ semble le meilleur [2].

L'optimisation de ces paramètres ainsi que d'autres qui interviendront dans le construction du modèle dans notre cas sera illustrée dans le chapitre suivant.

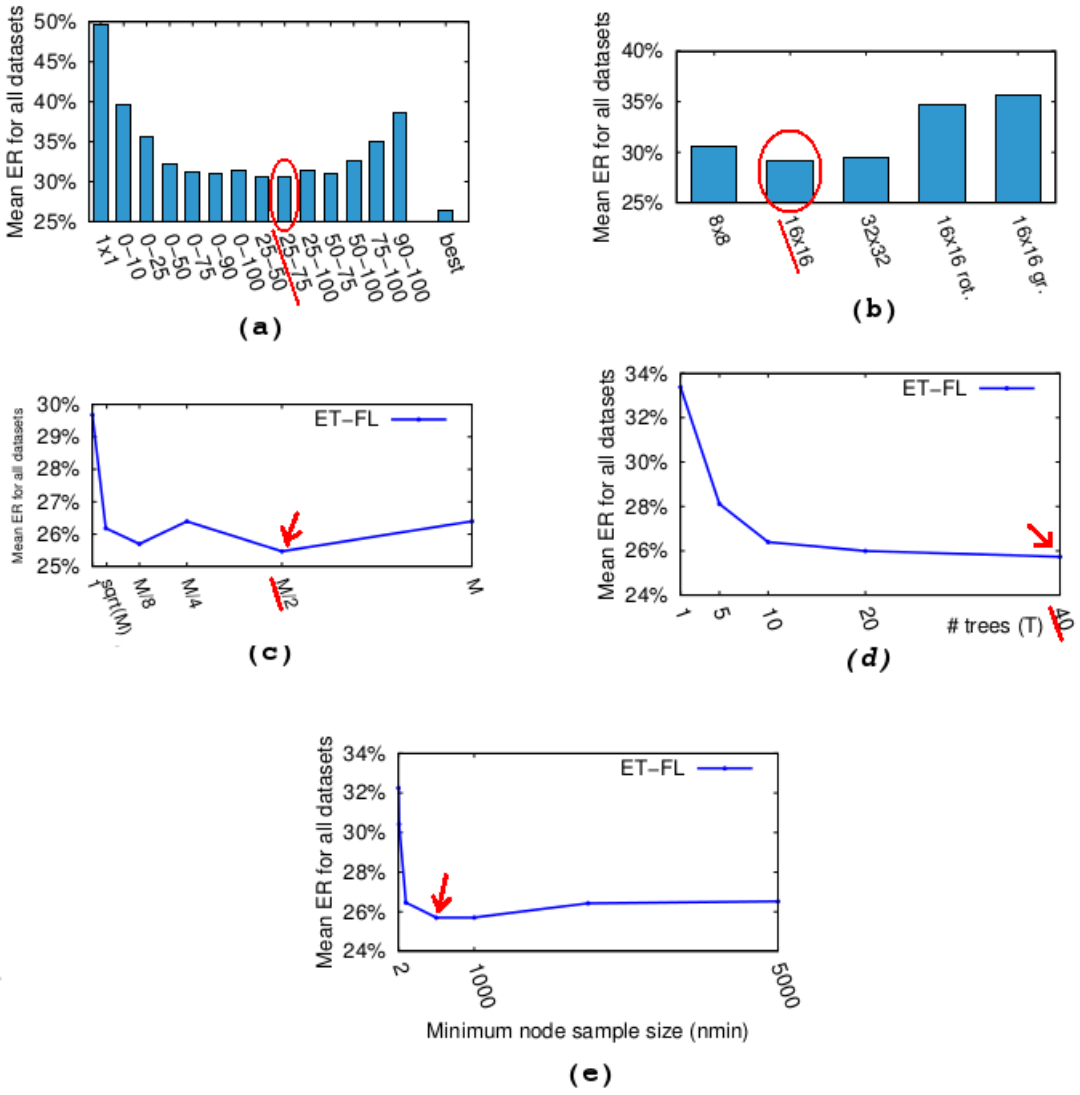


FIGURE 5.5: Variation du taux d'erreur en fonction des paramètres de la mèthode, en fonction de: (a) la taille proportionnelle des sous-fenêtres extraites (en %), (b) la taille finale des sous-fenêtres après redimensionnement, (c) le nombre de tests aléatoires K, (d) le nombre d'arbres dans l'ensemble, (e) le nombre minimum d'objets pour le développement d'un noeud. figure tirée de [2], et modifiée. Le taux d'erreur est calculée au travers de 80 jeux de données.

Chapitre 6

Application et réalisation

6.1 Nomenclature de la plate forme cytomine

Ce travail comme il a été souligné dans les lignes précédentes, s'intègre dans le développement de la plate-forme *Cytomine*. *Cytomine* est basée sur une approche $homme - machine$ hybride utilisée pour l'analyse des bioimages multidimensionnelles de grandes résolutions. L'approche combine les technologies web récentes, les conceptes des bases de données spatiales, les méthodes d'apprentissage automatique, et l'annotation collaborative [8].

Cytomine nous propose une architecture incluant une interface web, une base de données, et une API permettant la réalisation des opérations de base (figure 6.1):

- ajouter une annotation d'une région d'une image en utilisant les formes géométriques tellesque: cercle, polygone, etc, sous forme de layer
- supprimer une annotation
- télécharger une annotation ou un lot d'annotations, etc.

6.1.1 Projet

Un projet cytomine est caractérisé par un identifiant (*id_project*), un nom, un ensemble d'utilisateurs avec des droits d'accès, et une ontologie (ensemble des

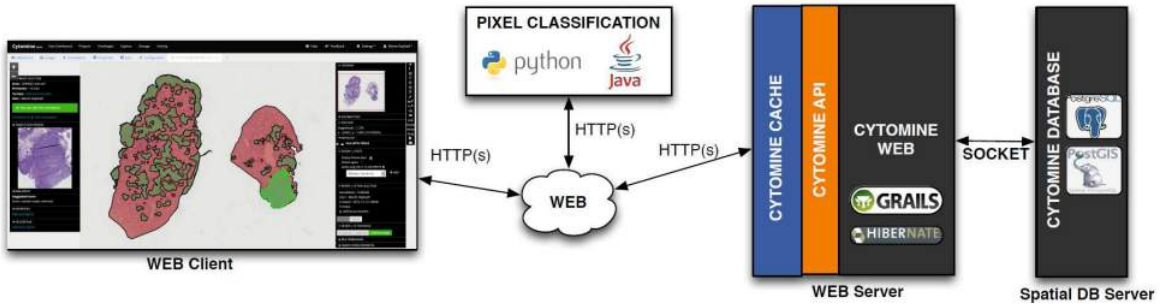


FIGURE 6.1: Architecture de la plateforme Cytomine. Figure tirée de [8]

termes). Une fois crée, on pourra ajouter des images au projet (les "uploader" et les "deployer" pour les associer au projet). Les projets sont stockés dans une base de données spatiale et relationnelle (*PostgreSQL* avec l'extension *PostGIS*), et peuvent être visualisés et édités via l'interface web de la plateforme ou par des programmes ou scripts tiers (*Java*, *Python*) [8].

6.1.2 Image

Les images de la plateforme sont multidimensionnelles et de grande résolution. Une image est caractérisée par un identificateur (*id_image*), et le projet auquel elle est associée. L'ajout d'une image peut se faire soit via l'interface de la plateforme, soit au moyen des scripts (*Python*) ou programmes (*Java*). Les images sont organisées par projet. Une image est divisée en tuiles (petites zones carrées) dans le but de faciliter l'accès à une zone de l'image à une résolution particulière. En spécifiant les coordonnées et le niveau de grossissement, on peut récuperer uniquement les tuiles formant la zone nécessaire.

6.1.3 Annotation

Une annotation forme l'entité principale du modèle (base de données relationnelle). Les annotations sont réalisées sur les images. L'ajout d'une annotation revient à délimiter une série de coordonnées qui forment le coutour de la zone désirée et ce en utilisant une des formes géometriques de sélection (polygone, cercle, rectangle, Ellipse) disponibles dans l'interface de la plateforme. Une fois la zone selectionnée, on pourra dès lors lui attribuer une ou plusieurs classes (on parle de termes).

Trois types d'annotation existent:

- **Annotation d'utilisateur**: annotation réalisée manuellement par un humain (expert) utilisée pour l'entraînement du modèle de classification
- **Annotation de job ou de programme**: annotation générée par un programme tiers *Java* ou un script *Python*
- **Annotation révisée**: pour les algorithmes de prédiction et corrigée par un utilisateur

Une annotation est référencée par une URL [8].

6.1.4 Crop d'annotation

Une image rectangulaire d'aire minimale contenant complètement l'annotation est appélée crop d'annotation. Ce crop est muni d'un masque de transparence distinguant l'objet de l'annotation de l'arrière plan. Lors de la récupération des annotations, le masque peut être contrôlé par le paramètre `cytomine_dump_type`.

6.1.5 Ontologie

L'ontologie représente toutes les classes (termes) utilisées dans le projet [21].

6.1.6 Utilisateur

Chaque utilisateur de la plateforme se voit attribuer une paire de clés publique et privée, et un espace de stockage sur le serveur.

6.1.7 Interface web

Pour la visualisation des images à des resolutions multiples dans des clients web (navigateurs), une interface web est implémentée utilisant les technolgies web (*javascript*, *JQuery*, *Backbone.js*, *etc.*). L'interface fournit également des informations statistiques sur le projet sous forme de graphes. Elle comprend également

un module de *review* permettant à un expert de parcourir les annotations générées automatiquement par les programmes et d'apporter une modification à leur classe si nécessaire [21].

En outre, un mécanisme de mémoire cache est implémentée entre le serveur et le client (figure 6.1) afin d'accélerer la récupération des données les plus fréquentes [19].

6.2 API et langage d'implémentation

Comme on peut le voir sur la figure 6.1, la plateforme fournit une API via des requêtes HTTP. Les différentes entités du modèle BDD de la plateforme sont codés sous forme d'objets. Ainsi, les annotations, les utilisateurs, les images, les jobs,... sont des objets. Ces derniers peuvent êtres manipulés directement par un programme tiers. Par exemple, on pourra écrire un script python permettant d'effectuer des annotations automatiques d'images.

Notre choix du langage d'implémentation s'est porté sur le langage Python.

Python, un langage interprété, et à la fois un langage de script et de programmation objet. Ce langage est utilisé sur la plupart des plateformes informatiques. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser [*wikipedia*]. Il est, en outre, très facile d'étendre les fonctionnalités existantes, il existe des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées.

Les bibliothèques Python qui nous intéressent dans notre travail sont les suivantes:

6.2.1 PIL (Python Imaging Library)

Cette librairie contient des classes pour le traitement d'images. Elle prend en charge de nombreux formats d'images. Le module *Image* de cette librairie fournit des fonctions incluant la chargement d'images à partir d'un fichier, et la création de nouvelles images.

6.2.2 pickle

Le module *pickle* implémente les fonctions pour vider les données de l'instance de classe dans un ficher et charger les données "picklées" pour les rendre utilisables. Dans notre cas, on l'utilise pour sérialiser un modèle crée (*.pkl*), et le "depickler" au moment de l'utilisation.

6.2.3 numpy

Extension du langage Python pour la manipulation des vecteurs et matrices de grandes dimensions.

6.2.4 scikit-learn

Librairie pour l'apprentissage automatique. Elle fournit des algorithmes de classification, de régression, de clustering incluant les ensembles d'arbres aléatoires, les SVM, $K-means$, le *boosting*, etc.[1]

[1]Le site en ligne: *http://scikit-learn.org/*

6.3 Grandes lignes de la méthode

La méthode qu'on va décrire repose sur la classification d'images sans aucun traitement préalable des images utilisées pour la construction du modèle, ni d'ailleurs pour les images à prédire. Il s'agit d'une utilisation des images à l'état "brut" sans passer par une phase de prétraitement. Par ailleurs, une étape de post-traitment des prédictions est appliquée. Il consiste en la collection des prédictions considérées jusque là positives par le modèle, et application d'un clustering en termes de coordoonnées, et filtration de la prédiction la plus haute au sein de chaque cluster. Cette étape de post-traitement est similaire à celle appliquée dans [1] pour la détection des noyaux des cellules.

Comme il est déja dit dans les sections précédentes, on commence par détecter (quantifier) les follicules dans une image de coupe de lame avant leur classification. On construit un modèle destiné uniquement à la détection de follicules dans une première étape. Ensuite dans une deuxième étape, un classifieur de six classes de follicules est construit et destiné à la classification (figure 6.2). Par ailleurs,

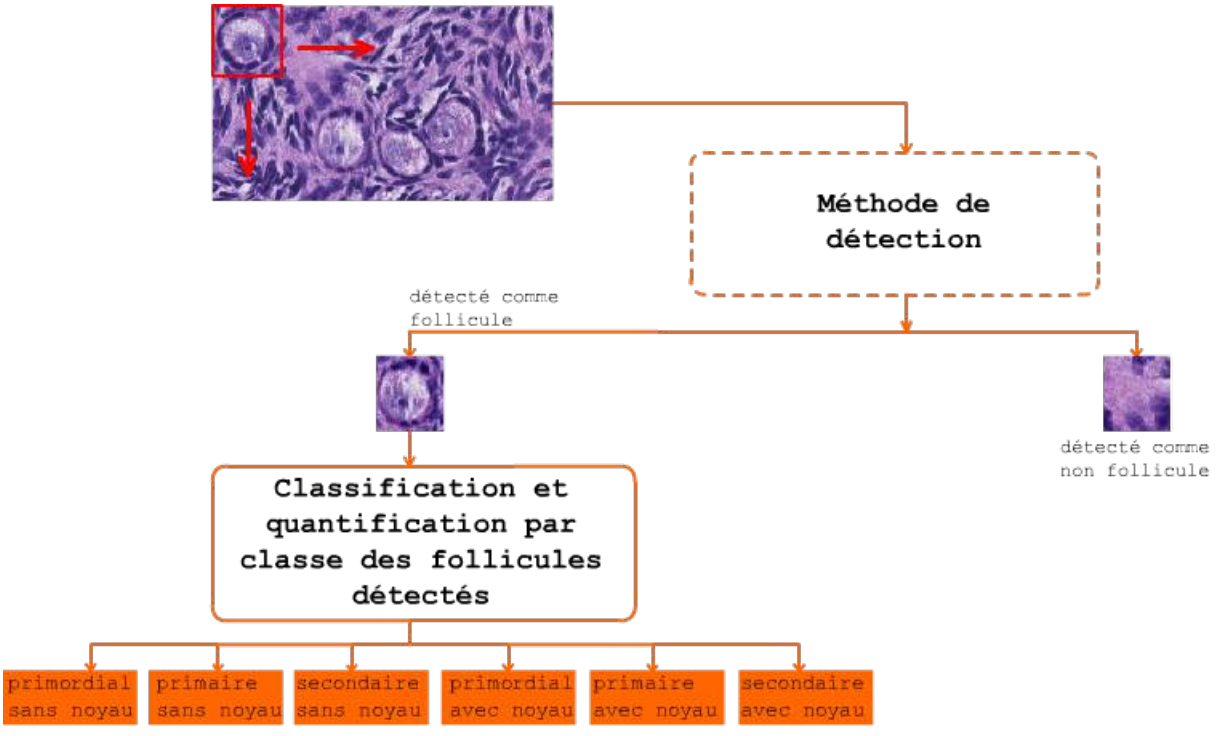


FIGURE 6.2: Le framework général de la méthode.

Une méthode de parcours exhaustive de la lame avec décalage de pixels (20 px) est appliquée. Une étape de post-traitement est ensuite appliquée sur les clusters

constitués des prédictions positives voisines (en termes de coordonnées) afin de dégager la prédiction la plus haute. Après cette étape de détection cruciale, l'objet detécté comme follicule est ensuite soumis à une classification.

6.3.1 Détecter et quantifier les follicules

Après avoir construit le modèle en fixant les paramètres optimaux (table 6.1), une méthode de parcours de la lame est appliquée, suivie d'une étape de post-traitement des follicules positivement prédits (clustering et merging). Par ailleurs, l'utilisation de cette méthode de détection est rendue possible du fait que nous avons observé par application du modèle que la probabilité de détection d'un follicule diminue au fur et à mesure qu'on s'éloigne du follicule. En d'autres termes, le taux de détection d'une image (annotation) contenant le follicule entier est supérieur au taux de détection d'une image contenant une partie du follicule, c'est ce qui est montré sur la figure 6.4.

6.3.2 Construction du modèle pour la détection des follicules

Le modèle est entrainé avec les objets "follicules" et les objets "non-follicules" (négatifs). L'ensemble des objets "follicules" est constitué de toutes les annotations classées comme follicules par les chercheurs, tandis que l'ensemble des objets du deuxième ensemble est constitué des vaisseaux et des annotations en plus qu'on a réalisé dans des régions différentes sur les lames (en dehors des follicules) en nombre de 500[2], et qu'on a ajouté au projet avec l'ontologie "$Negatif$" afin d'égaliser les tailles des deux classes.

Nous avons ajouté au projet l'ontologie suivante: $temp\_foll$, $temp\_non$.

- $temp\_foll$ (taille = 701): contient les follicules de tout type: primordiaux sans/avec noyau, primaire sans/avec noyau, secondaire sans/avec noyau.
- $temp\_non$ (taille = 688): contient les vaisseaux et les annotations négatives.

[2]Au départ, on a réalisé 200 annotations négatives qu'on ajouté à l'ensemble d'apprentissage, et les résultats n'étaient pas bons à cause de la disporportion des tailles des deux classes.

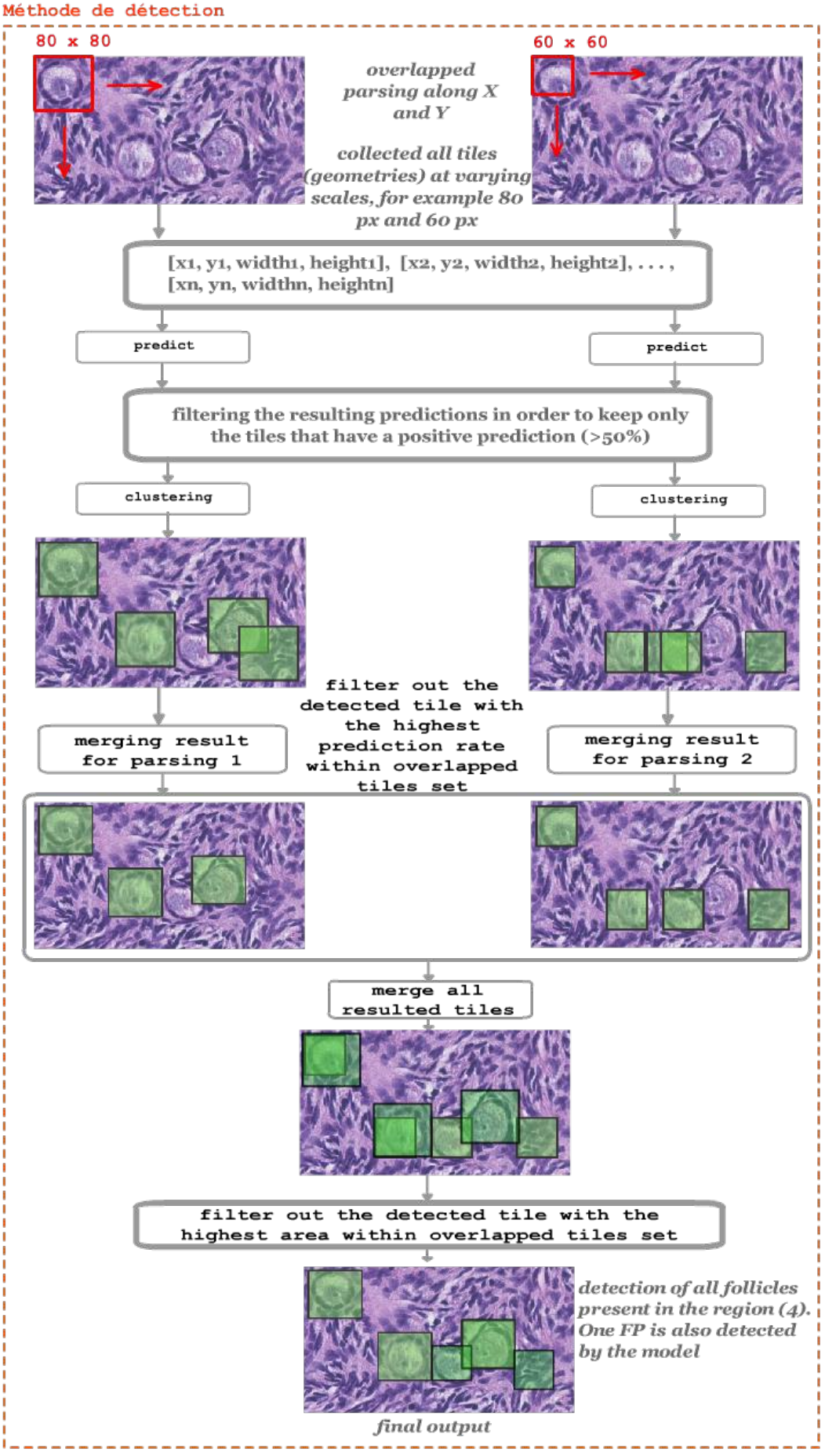


FIGURE 6.3: Le framework de la technique de détection utilisée.

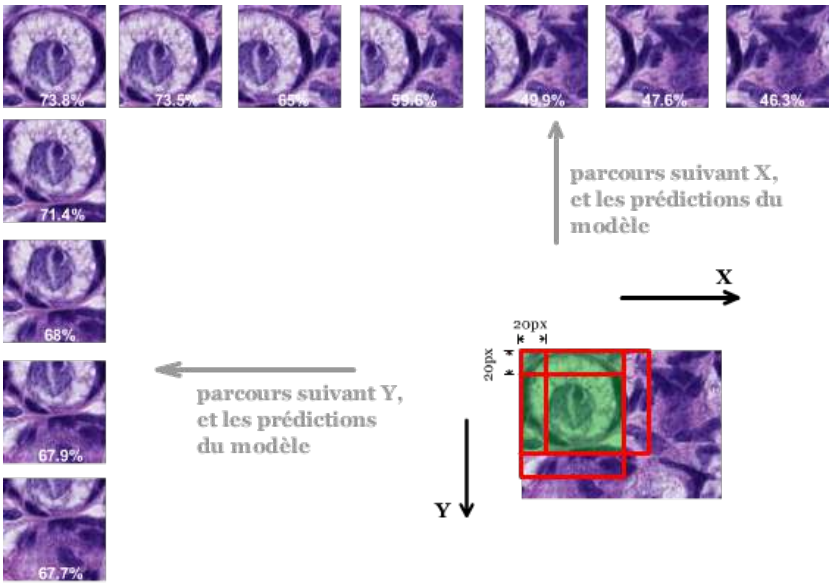


FIGURE 6.4: Parcours avec décalage de $20px$ et Probabilités de détection suivant les axes de décalages X et Y. On remarque une diminution des prédictions plus on s'éloigne du follicule. L'image du follicule sur la figure sur laquelle on a appliqué le modèle ne fait pas partie des données d'entraînement (LS) du modèle

Le classifieur détermine si un nouvel objet est un follicule (classe: $temp\_foll$) ou non (classe: $temp\_non$).

Le tableau 6.1 montre les paramètres pour la construction du modèle. Les paramètres optimaux surlignés en jaune sur le tableau sont obtenus en testant les combinaisons figurant dans le même tableau parmi l'ensemble des paramètres suivants:

`-pyxit_n_subwindows`: nombre de sous-fenêtres extraites par image
`-pyxit_min_size`: la taille minimale de sous-fenêtres extraites
`-pyxit_max_size`: la taille maximale des sous-fenêtres extraites
`-pyxit_target_width`: largeur des sous-fenêtres redimensionnées
`-pyxit_target_height`: hauteur des sous-fenêtres redimensionnées
`-forest_n_estimators`: nombre d'arbres utilisés
`-forest_max_features`: le nombre maximun d'attributs parmi lesquels on choisit le test de séparation
`-forest_min_samples_split (n_min)`: taille minimale d'ensemble pour créer un noeud
`-svm`: un flag pour l'utilisation de la couche SVM.

En ajoutant la couche SVM au modèle, les temps mis pour dénicher les bonnes valeurs des paramètres ci-dessus pour chaque combinaison deviennent plus importants. Ceci revient au fait qu'on teste ces paramètres pour deux modèles en réalité: les Extra-trees aléatoires pour l'apprentissage des paramètres, et le classifieur SVM. Du coup, les temps des deux tests sont additionnés, et les matrices de confusions générées sont des matrices cumulées (voir tableau 6.2).

La précision est calculée à partir de la matrice de confusion de la table (6.2). L'équation du calcul est la suivante:

$$precision = \frac{TP}{TP+FP}$$

- des paramètres en plus sont utilisés:

`-cv_k_folds`: un paramètre pour la validation croisée (voir ci-dessous)
`-pyxit_colorspace`: spécifie le mode couleur dans lequel on travaille: 2 mode RGB, 3 en niveau de gris.
`-cytomine_dump_type`: spécifie le type de crop d'annotation: 1: crop d'annotation incluant le backgroud, 2: crop d'annotation en mode alpha (sans le background).

Par comparaison aux valeurs des paramètres pour un modèle optimal de la figure 5.5 et exposés dans [2], on remarque que dans notre cas les valeurs des parmètres `pyxit_min_size`, `pyxit_max_size`, `forest_n_estimators`, et `forest_min_samples_split` sont légèrement différentes.

Le temps nécessaire pour l'entraînement et la construction du modèle de détection avec les paramètres optimaux du tableau 6.1 est de 69.7 min[3], ceci inclut la construction des *Extra-trees*, l'apprentissage des paramètres pour le classifieur SVM (Kernel linéaire), et l'entraînement de ce dernier.

6.3.2.1 La validation croisée k-fold

Cette procédure divise l'ensemble des données en k sous-ensembles. Les k–1 sous-ensembles sont utilisés pour entrainer le modèle, tandisque le k^{eme} sous-ensemble

[3]Les temps sont mesurés en exécutant les scripts sous le système UBUNTU 12.10 tournant sur un serveur de 30 processeurs, une fréquence de 2099.998 MHZ chacun

k_folds	n_subw	target_width target_height	min_size max_size	n_est T	max_features K	n_min	SVM	Erreur(FP/FN) (%/%)	précision (%)
3	100	32x32	[0.25-0.50]	40	M/2=1536	500	1	19/11	82.40
3	100	32x32	[0.75-0.95]	40	M/2=1536	500	1	18/11	83.17
3	200	32x32	[0.75-0.95]	40	M/2=1536	500	1	18/11	83.17
3	100	16x16	[0.75-0.95]	40	M/2=384	500	1	27/15	75.89
6	100	32x32	[0.75-0.95]	10	M/2=1536	500	1	12/8	88.46
6	100	32x32	[0.75-0.95]	10	M/8=384	500	1	11/9	89.21
6	100	32x32	[0.75-0.95]	10	M/8=384	200	1	10/9	90.10
6	300	32x32	[0.75-0.95]	10	M/8=384	200	1	10/10	90.00
6	200	32x32	[0.75-0.95]	10	M/2=1536	500	1	10/10	90.00
6	200	32x32	[0.75-0.95]	10	M/2=1536	200	1	11/9	89.21
6	100	32x32	[0.75-0.95]	10	$\sqrt{M} = 56$	100	1	18/8	83.63
6	100	32x32	[0.75-0.95]	10	M/4=768	10	1	14/10	86.53

TABLE 6.1: Extraction empirique des paramètres optimaux pour le modèle de détection

	$Non\_follicule$	$Follicule$	Total
$Non\_follicule$	544	58	90%
$Follicule$	60	641	91%

TABLE 6.2: Matrice de confusion du modèle de détection optimal (surligné sur le tableau 6.1)

est utilisé comme ensemble de test. Le procédé est répété jusqu'à ce que chaque sous-ensemble est utilisé comme un ensemble de test [1]. Dans notre cas, cinq sous-ensembles d'annotations sont utilisés à chaque fois (6 fois) pour construire le modèle, et le 6ème sous-ensemble est utilisé comme ensemble de test. Le meilleur taux d'erreur parmi les k taux d'erreur est ensuite utilisé pour le calcul de la précision du modèle suivant l'équation précédente. Rappelons que la validation croisée est utilisée uniquement pour les tests sur les modèles, et ainsi trouver les bons paramètres. La construction du modèle se fait ensuite dans une autre étape en utilisant ces paramètres[4] sur l'ensemble des données d'apprentissage.

6.3.2.2 Parcours de la lame

Deux parcours exhaustifs de la lame sont réalisés. Les parcours sont faits avec décalage de 20 pixels suivant les deux axes X et Y comme montré sur la figure 6.4. Un parcours a pour but de récupérer les tuiles[5] d'une taille donnée avant de les soumettre au classifieur. Deux tailles des "*readers*" sont utilisées pour le parcours:
- `80 X 80` pour la detection des grands objets, et
- `60 X 60` pour les petits objets.

A noter sur ce point que la variante utilisée par les auteurs dans [1] pour le parcours afin de détecter les noyaux des cellules utilise plusieurs tailles car les objets à detecter sont de différentes tailles.

[4] on utilise deux scripts python différents pour ces deux taches: trouver les paramètres, et construire le modèle

[5] Une tuile est une forme géométrique incluant les coordonnées (x, y) et les dimensions

Les tuiles ainsi récupérées après le parcours sont ensuite soumises à une classification avec le modèle construit auparavant. La méthode ne garde que les tuiles positivement prédites (follicules) avec un taux $\geq 50\%$[6].

À remarquer également que l'utilité du parcours exhaustif avec décalage est que ce dernier nous permet de localiser totalement le follicule entier et de nous éviter de tomber dans des cas comme celui de la figure (6.5.b) obtenu avec un parcours exhaustif simple sans décalage.

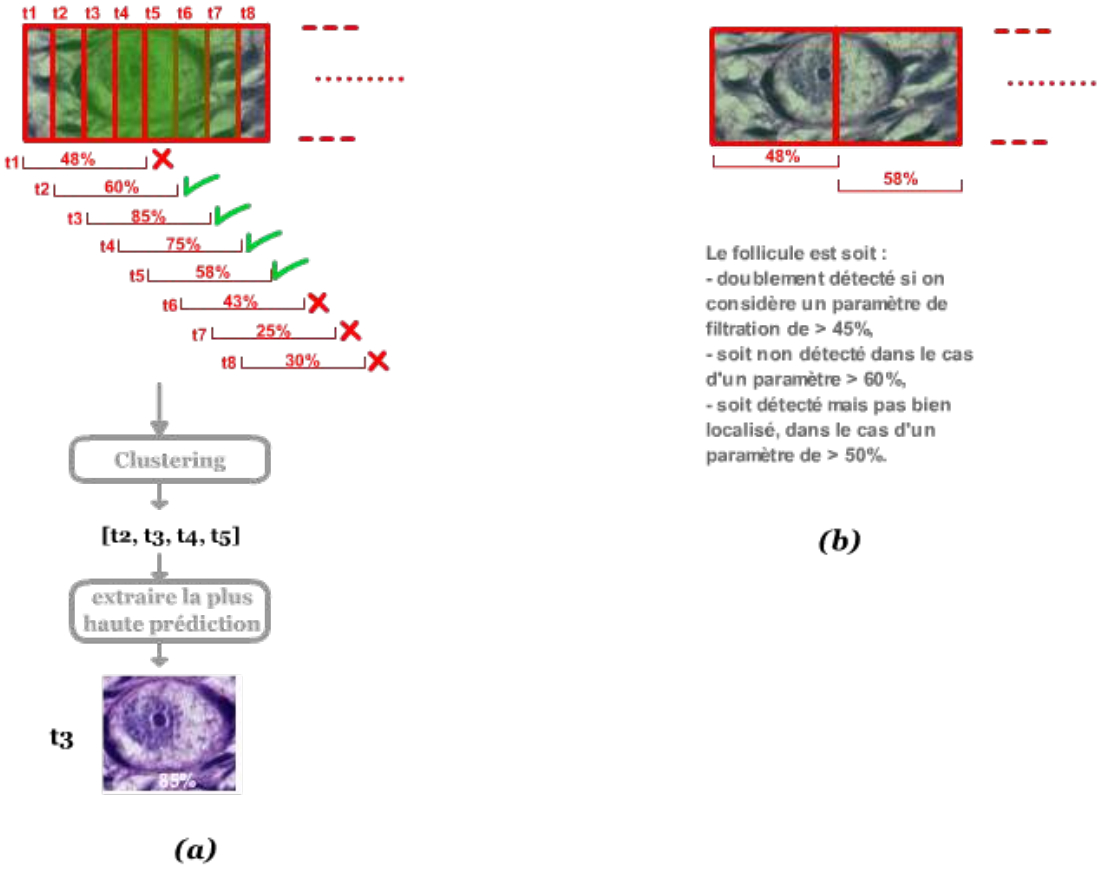


FIGURE 6.5: Exemple d'application de la méthode: (a) parcours avec décalage et application du clustering donne un seul follicule détecté et entièrement localisé. (b) parcours exhaustif sans décalage

[6]Logiquement, vu qu'il s'agit d'un classifieur binaire. Par ailleurs, ce seuil est ajustable et des résultats avec un seuil de plus 60% sont parfois meilleurs que ceux obtenus avec un seuil de 50%

6.3.2.3 Clustering

Un simple clustering (Algorithmes 5, 6) est ensuite appliqué sur l'ensemble des tuiles positives ($\geq 50\%$) afin de ne garder que la meilleure tuile parmi plusieurs issues du parcours du même follicule. Par exemple, l'application du clustering sur les tuiles de la figure 6.4 nous ne donne que la tuile contenant le follicule entier. Les clusters sont formés en fonctions des coordonnées des tuiles positives: deux tuiles positives dont les coordonnées sont espacées de moins de 80 pixels (ou 60 pixels pour le deuxième parcours) suivant les deux axes X et Y sont groupées dans le même cluster.

Algorithm 5 *clustering*$(Liste, Liste\_clusters)$

- **Si** *Liste* est vide **alors**:
  - renvoyer $Liste\_clusters$
- **Sinon**:
  - Soit t_1 la tuile en tête de la liste *Liste*
  - Soit $cluster\_courant = [\ ]$
  - ajouter t_1 à $cluster\_courant$
  - **Pour** toute tuile $t_i \in Liste$ **faire**:
    - **Si** $closest(t_1, t_i) = vrai$ **alors**:
      - ajouter t_i à $cluster\_courant$
    - **Fsi**
  - **fait**
  - ajouter $cluster\_courant$ à $Liste\_clusters$
  - $Liste = Liste$ – les éléments de $cluster\_courant$
  - *clustering*$(Liste)$

Algorithm 6 *closest*(t_1, t_2)

- $distance\_max = 80px$ (ou $60px$)
- soient $t_1 = [x_1, y_1, width_1, height_1]$ et
  $t_2 = [x_2, y_2, width_2, height_2]$ deux géométries (tuiles)
- **Si** $|x_1 – x_2| \leq distance\_max$ et $|y_1 – y_2| \leq distance\_max$ **alors**:
  - Renvoyer $Vrai$
- **Sinon**:
  - Renvoyer $Faux$
- **Fsi**

6.3.2.4 Merging

Il arrive parfois que deux détections positives se chevauchent même après l'application du clustering[7], pour cela de telles détections sont fusionnées en une seule et unique détection. Ceci est réalisé en gardant la tuile de la plus haute detection parmi elles.

Un merging final (Algorithme 7) est ensuite appliqué pour la fusion des deux résultats issus des deux analyses avec des tailles 80x80 et 60x60. Ce merging permet également de fusionner les tuiles, mais cette fois-ci en gardant la tuile dont l'aire est la plus grande. Nous dirons que deux tuiles détectent le même objet si l'intersection de leurs surfaces est supérieure ou égale à $1600px^2$.

Algorithm 7 $merging(t_1, t_2)$

- $inter\_surface = 1600px^2$ pour 40x40 pixels
- soient $aire_1 = \text{surface}(t_1)$ et
  $aire_2 = \text{surface}(t_2)$
- **Si** $aire_1 \cap aire_2 \geq inter\_surface$ **alors**:
  garder uniquement une tuile dont l'aire est la plus grande
- **Fsi**

6.3.3 Révision et correction

Cytomine dispose d'un module de révision/correction afin de corriger les classifications ratées. Ceci nous permettra d'avoir un ensemble d'apprentissage plus grand, plus complet, et plus adapté au problème de détection. Nous avons procédé au raffinement du modèle de détection en révisant quelques 600 annotations qu'on ajouté par la suite à l'ensemble d'apprentissage pour reconstruire un modèle plus performant.

[7] Dans le cas de deux meilleures tuiles positivement prédites issues de deux clusters voisins, et dont les surfaces se chevauchent

	primo	prim	sec	primo_SN	prim_SN	sec_SN	Total
primo	75	5	0	4	0	0	89%
prim	8	34	0	4	4	0	68%
sec	0	0	0	0	2	0	0%
primo_SN	13	3	0	37	6	0	62%
prim_SN	2	7	0	14	17	0	42%
sec_SN	0	2	0	2	0	0	0%

TABLE 6.3: Matrice de confusion du modèle de classification. Légende: *primo*: primordial avec noyau, *prim*: primaire avec noyau, *sec*: secondaire avec noyau, *primo_SN*: primordial sans noyau, *prim_SN*: primaire sans noyau, *sec_SN*: secondaire sans noyau

6.3.4 Classification des follicules

Pour construire le modèle destiné à la classification des follicules dans les six classes, nous avons adopté les mêmes valeurs de paramètres que le modèle de détection à l'exception de `n_subw` = `300`[8].

Vu le peu d'annotations de follicules secondaires (sans noyau (9) et avec noyau (3)), le modèle ne saura pas distinguer (il ne va pas "apprendre" les caractéristiques de ces types de follicules) ces deux types parmi les autres types. La disporpotion flagrante des tailles des classes (voir tableau 2.11) rend la classificaton des follicules en six classes difficile. Aucune chance d'avoir un follicule classé comme secondaire (sans/avec noyau) avec un modèle construit avec ces données.

À première vue, le modèle construit ne sera pas performant à cause de la disporpotion des tailles des classes, ainsi que la ressemblance des follicules issues de deux classes différentes, comme on l'a déja souligné dans le chapitre 2. Ces deux paramètres vont influencer négativement les résultats obtenus. La matrice de confusion obtenue pour le modèle (table 6.3) confirme ces difficultés. On remarque que le modèle n'arrive pas bien à classifier dans les deux classes de follicules secondaires à cause de manque d'informations caractérisant ces types de follicules. Le modèle n'arrive pas également à distinguer entre les deux classes "*primaire sans noyau*" et "*primordial sans noyau*" à cause de la ressemblance de leurs éléments.

[8]Nous avons procédé à l'extraction des paramètres optimaux pour le modèle de classification basés sur les six classes en testant quelques combinaisons de paramètres en validation croisée k = 6. Ces valeurs semblent donner le meilleur taux de classification moyen par classe

Il est à noter que le modèle de classification est construit en utilisant les données des six classes fournies. L'ensemble d'apprentissage est constitué de 701 exemples répartis dans six classes (tableau 2.1).

Les résultats de cette classification sont montrés dans le chapitre suivant.

Chapitre 7

Tests et Résultats

Dans ce chapitre, nous allons exposer les tests effectués et les résultats obtenus sur d'autres régions de lames. Nous commençons par les résultats de la détection, et dans un deuxième temps nous exposerons les résultats de la classification.

7.1 Résultats pour la détection et la quantification

Pour évaluer les performances de la méthode de détection, nous l'avons appliquée à quatres régions issues des lames différentes. Les mesures de performances sont basées sur les métriques de détection appliquées dans les challenges de reconnaissance des formes. Ces mesures sont:

- Le nombre d'objets (follicules) détectés D.
- *True Positive* (TP): le nombre d'objets (follicules) qui sont vraiment de la classe "follicule" parmi les D objets détectés.
- *False Positive* (FP): le nombre d'objets détectés comme follicules parmi les D objets alors qu'en réalité ils ne sont pas des follicules.
- *False Negative* (FN): le nombre d'objets "follicules" qui sont dans la région, mais qu'ils ne sont pas détectés par la méthode.
- recall (sensibilité): il s'agit d'une mesure qui peut être calculée comme suit:

$$recall = \frac{TP}{TP+FN}$$

- $precision = \frac{TP}{TP+FP}$ comme on l'a déja vu dans le chapitre 6.
- *F-measure*: est une mesure de score. Les meilleurs scores sont ceux proches de 1 pour cette métrique. Elle est calculée comme suit:

$$F\text{-}measure = 2 * \frac{precision*recall}{precision+recall}$$

Seul un expert sera à même de déterminer la qualité de la classification, et intervenir dans la mesure de ces métriques. Mais, nous essayerons quand même de donner ces mesures basées sur notre propre observation.

7.1.1 Détection dans quatre régions différentes

Nous avons choisi de faire des tests sur quatre régions issues de différentes images (lames). Les régions choisies sont de différentes tailles, et contiennent un nombre différents de follicules qui sont disposés soit en amas, soit individuellement. Pour chaque région, nous donnons les résultats visuels obtenus. Nous montrons également les probabilités de détection. Pour la première région, nous exposerons en outre l'application de la méthode en passant par les différentes étapes visuellement. Par ailleurs, un résumé des résultats de détection et les temps requis est synthétisé dans le tableau 7.1. Les mesures des métriques exposées précédemment sont quant à elles montrées dans le tableau 7.2, et ce pour les quatres régions.

Les régions analysées sont issues de coupes de lames ne contenant pas d'annotations qui ont servi à l'entraînement du modèle.

7.1.1.1 Région 1

Dans la figure (figure 7.1) suivante on pourra voir la coupe dont elle issue la région analysée. La coupe de la région ne contient aucune annotation.

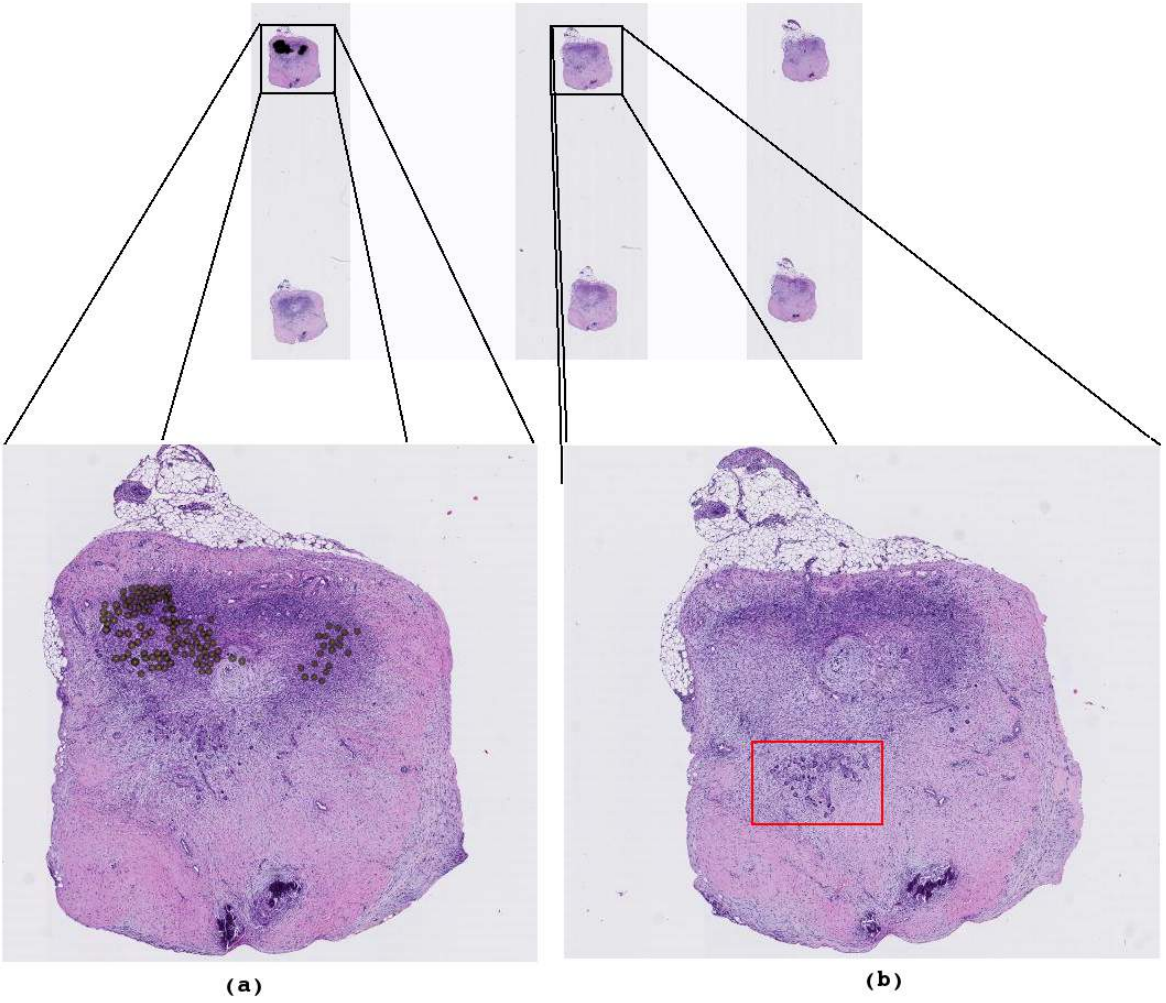


FIGURE 7.1: La région 1: (a) annotations négatives faites à posteriori et dont on a parlé dans le chapitre 6. (b) région d'analyse entourée en rouge

Les différents stades de détection de la méthode appliquée à la région 1 sont montrées dans la figure 7.2.

Les probabilités de détection pour cette région peuvent être visualisées sur la plateforme *Cytomine* (figure 7.3).

Num région	coordonnées(*)	id_image $(x_1,y_1),(x_2,y_2)$	taille $(widthxheight)$	#follicules(**) dans la région	#détections(†)	runtime (secondes)
1	(37427,7253) (38669,8047)	86519895	1242x794	±70	68	990
2	(71856,5396) (73256,6588)	86519643	1400x1192	±68	71	1434
3	(56726,29397) (58054,30351)	86502020	672x954	±27	24	1110
4	(8346,5178) (9660,5700)	86520268	1314x522	±21	22	618

TABLE 7.1: Résumé des détections dans 4 régions différentes: (*): coordonnées de début et de fin de la région. (x_1,y_1): le premier point en haut à gauche , (x_2,y_2): le dernier point en bàs à droite. L'*id_image* est l'image dont elle est issue la région analysée. (**): le comptage est fait à la main. (†): le nombre de détections considérées positives par le modèle.

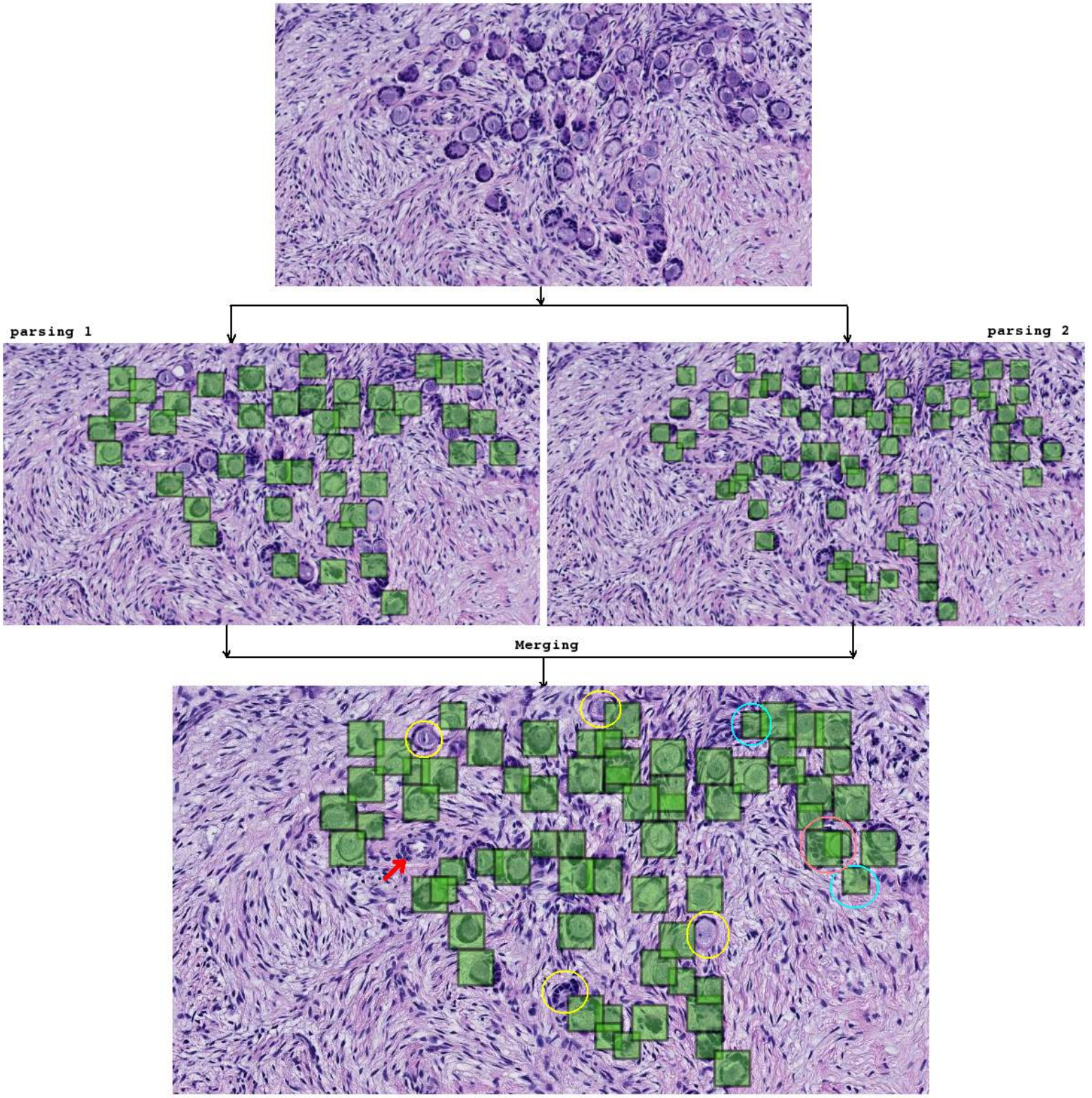


FIGURE 7.2: Les différents stades de détection. Sur l'image du résultat final: Entourés par le cercle jaune, la méthode n'arrive pas à détecter ces follicules (c'est des exemples de FN). Deux exemples de faux positifs sont entourés en cyan. On remarque aussi que la méthode arrive à descriminer le vaisseau (flèche rouge). La méthode arrive à détecter deux follicules collés entre eux (cercle rose sur la figure)

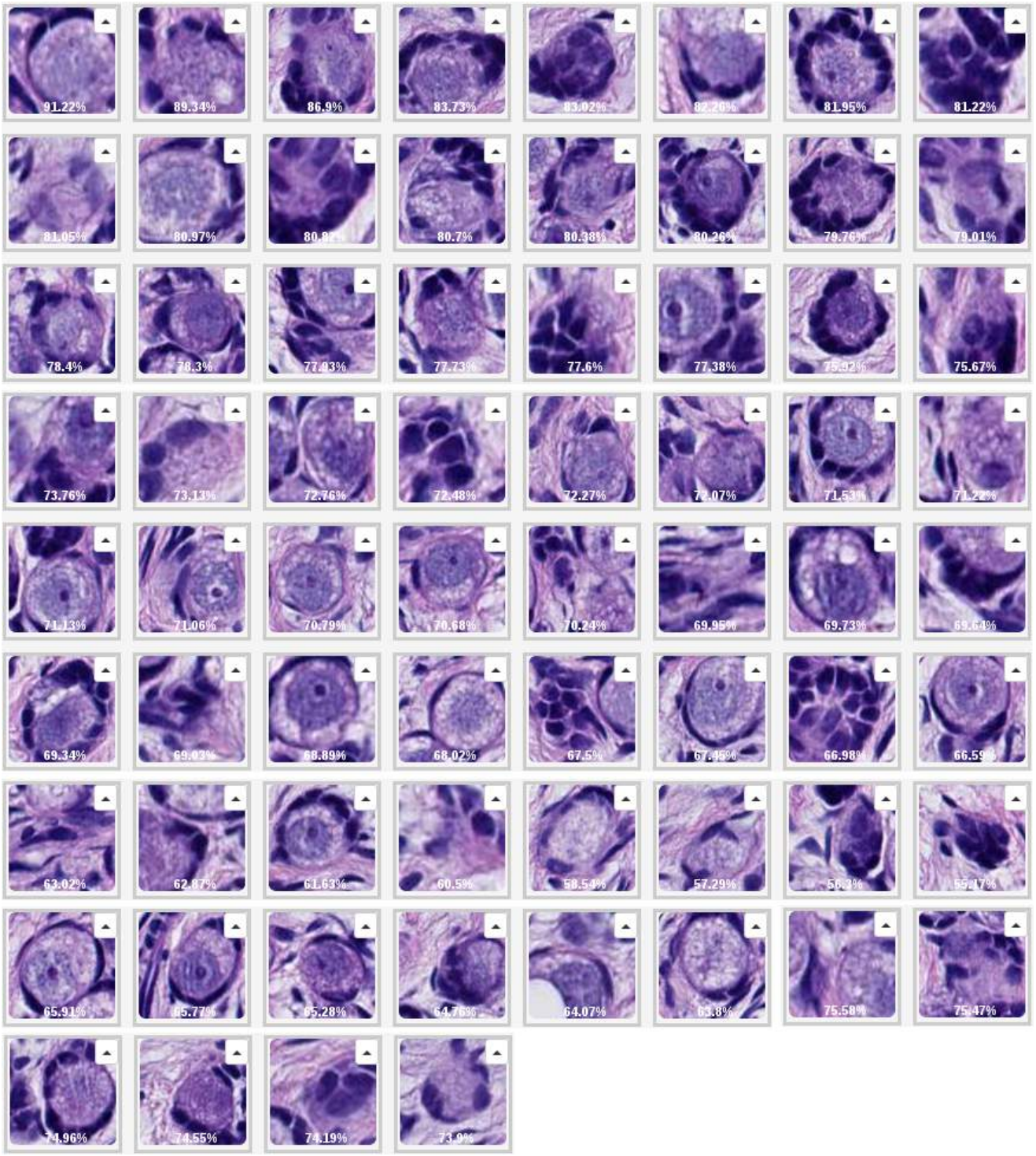


FIGURE 7.3: Les probabilités des détections positives de la région 1

région	D	TP	FP	FN	recall(%)	precison	F-measure(%)
1	68	64	04	05	92.75	94.11	93.54
2	71	65	06	04	94.20	91.54	93.22
3	24	22	02	06	78.57	91.66	84.72
4	22	16	07	05	76.19	69.56	75.17

TABLE 7.2: Mesures des différentes métriques de détection pour les 4 régions

7.1.1.2 Région 2

La région analysée, la lame dont elle est issue, ainsi que le résultat de l'application de la méthode sur cette région sont montrés dans la figure 7.4. Les probabilités des détections positives sont montrées sur la figure 7.5.

Pour tester la capacité de généralisation de la méthode, deux autres régions sont ajoutées aux tests: les deux régions 3 et 4. La robustesse d'une méthode consiste à la tester sur différentes images. Une méthode robuste ne va pas se laisser influencer par des petits changements. Dans notre cas, la robustesse de notre méthode est liée directement à la qualité du modèle construit.

7.1.1.3 Région 3

On remarque immédiatement l'apparition des faux positifs et des faux négatifs, et ceci est dû principalement à la qualité du modèle qui à son tour liée à la qualité des données d'entraînement (figure 7.6). Les probabilités des prédictions de cette région sont affichées sur la figure 7.7. Elles sont plus faibles que celles obtenues pour les régions 1 et 2, et ceci à cause de la couleur plus foncée de la région 3.

7.1.1.4 Région 4

Les mêmes constatations que précédemment sont à remarquer sur la détection dans la région 4. Les résultats sont sur la figure 7.8. On remarque cette fois ci que la méthode n'a pas arrivé à bien détecter les follicules dans la région entourée en jaune sur la figure comme elle le fait dans la zone entourée en vert.

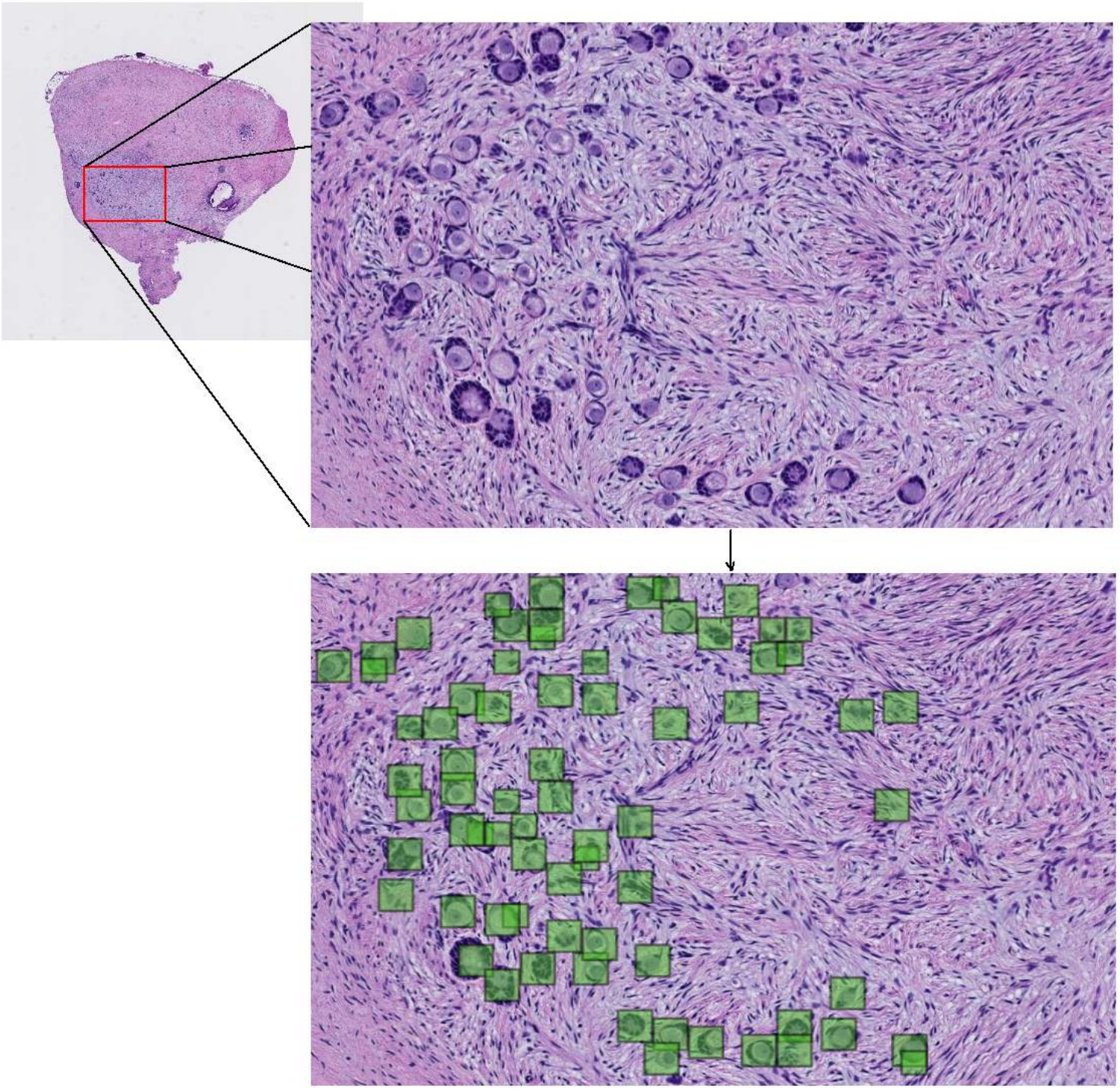


FIGURE 7.4: Détection dans la région 2

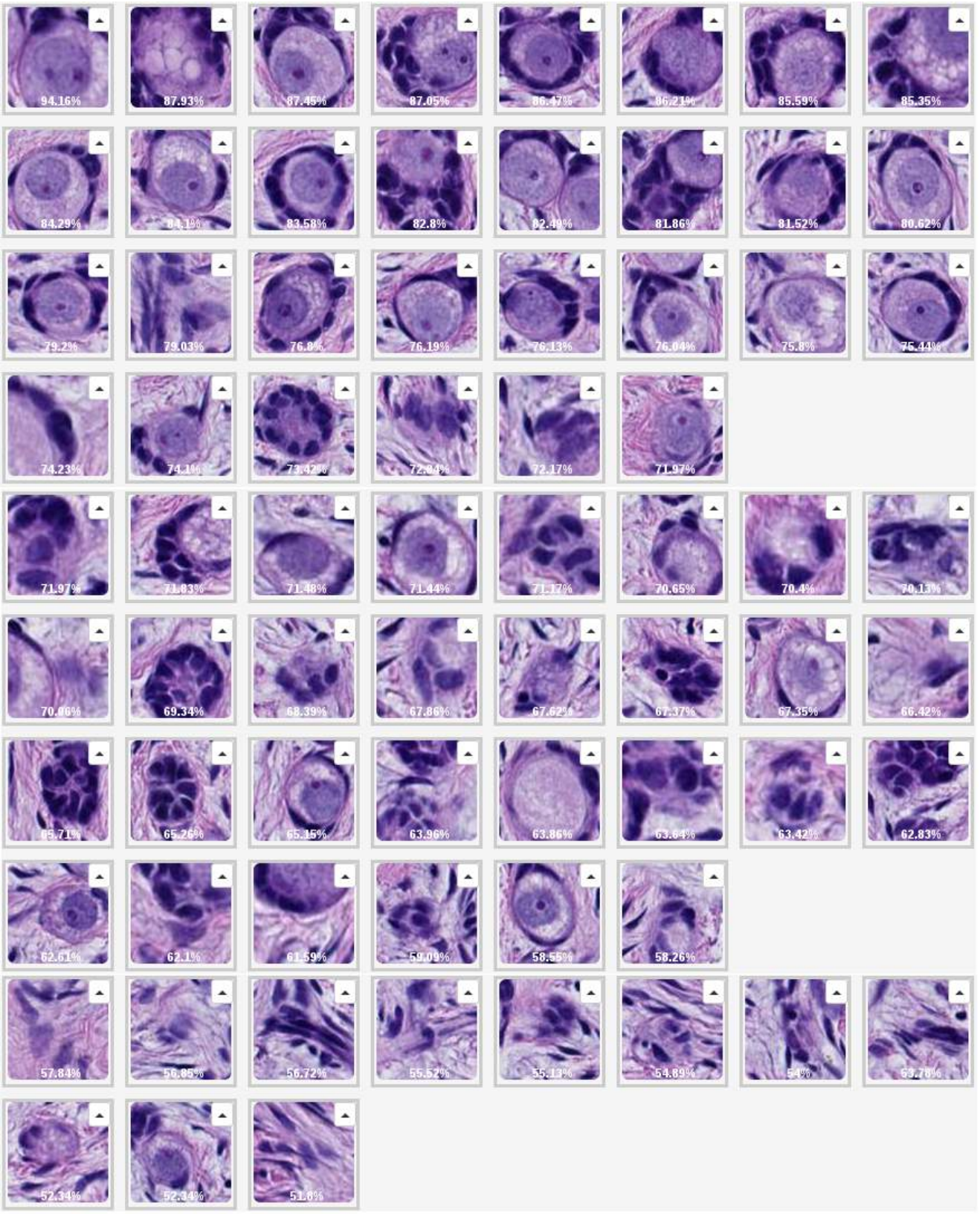


FIGURE 7.5: Les probabilités des détections positives de la région 2

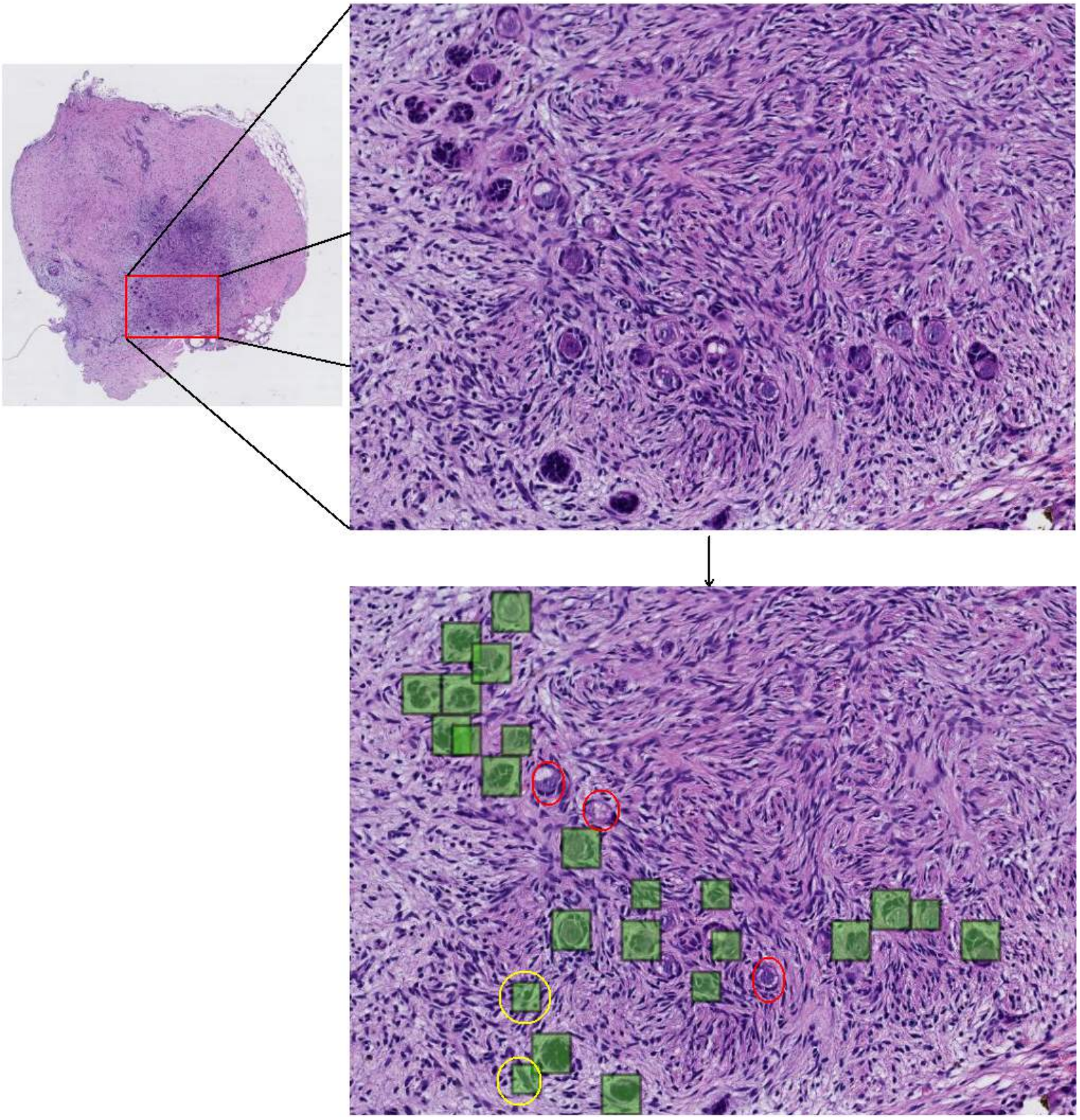


FIGURE 7.6: Détection dans la région 3: follicules non détectés entourés en rouge (FN), et en jaune des exemples de faux positifs

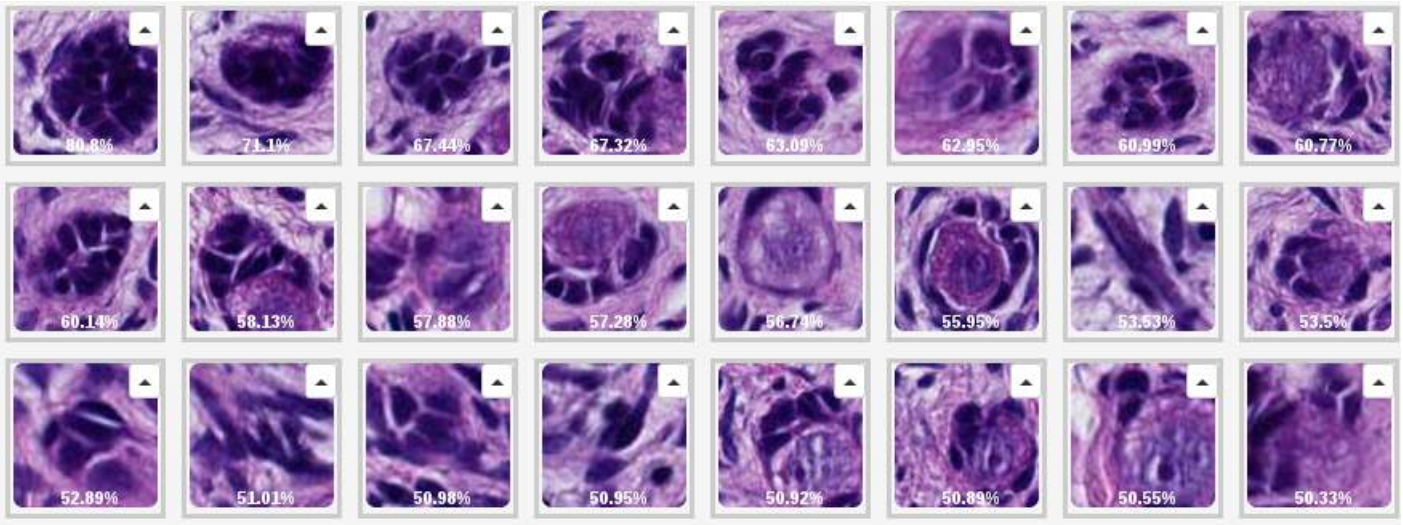


FIGURE 7.7: Les probabilités des détections positives de la région 3

En gros, sur les trois régions 1, 2, et 3, la qualité de la prédiction peut être qualifiée de bonne, tandis que sur la région 4, elle est moyenne.

7.1.2 Détection dans une grande région de la lame

Nous effectuons un test de détection dans une grande région. La région est d'une taille de 6000x4000 pixels (figure 7.9). La région montre une zone sombre dans laquelle on remarque la présence de follicules. Le résultat visuel de la détection est montré dans la figure 7.10. Les follicules sont détectés, néanmoins des faux positifs sont également détectés. Le temps nécessaire pour scanner une telle région est de 89 minutes[1].

[1]Le temps est la somme des temps de téléchargements des tuiles, des temps des prédictions, du clustering et du merging, et de la pubication des résultats sur *Cytomine*.

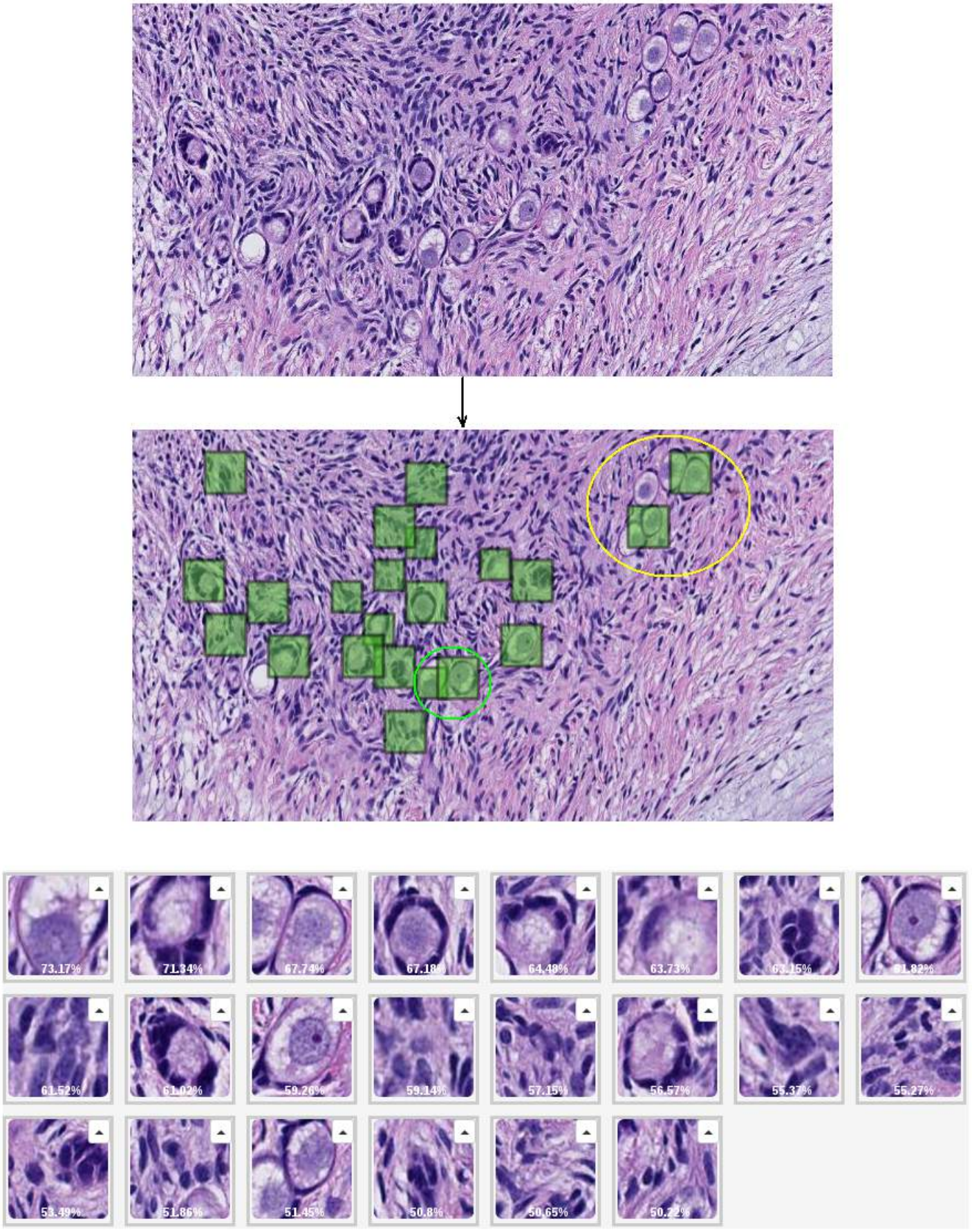


FIGURE 7.8: Détection dans la région 4, et les probabilités des détections positives

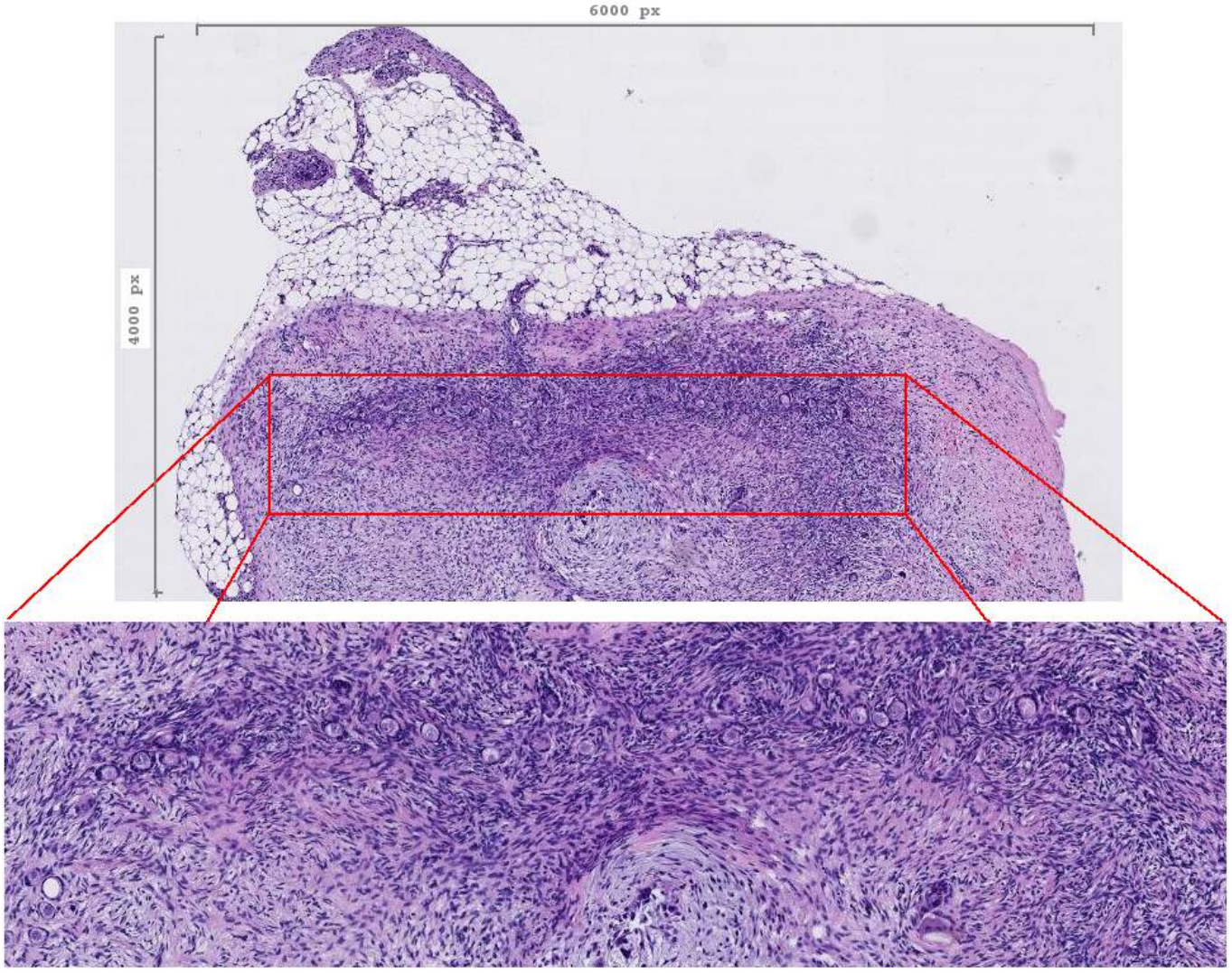


FIGURE 7.9: Application à une grande image: en haut la grande région à analyser. En bàs, un zoom de la région contenant les follicules. La méthode devra détecter la présence des follicules dans cette zone

7.2 Résultats de la classification

Dans le cadre de l'évaluation des capacités de la méthode appliquée à la classification des follicules en utilisant un modèle à six classes, des tests ont été effectués sur les follicules détectés dans les régions de test précédentes. Tout follicule détecté se voit attribuer une classe parmi les six classes par le modèle de classification. Nous nous contentons de faire la classification des follicules détectés dans la région 1 et 2.

Les résultats des tests effectuées sur la région 1 pour la classification des follicules détectés sont montrés sur la figure 7.11. La répartition de ces follicules par classe

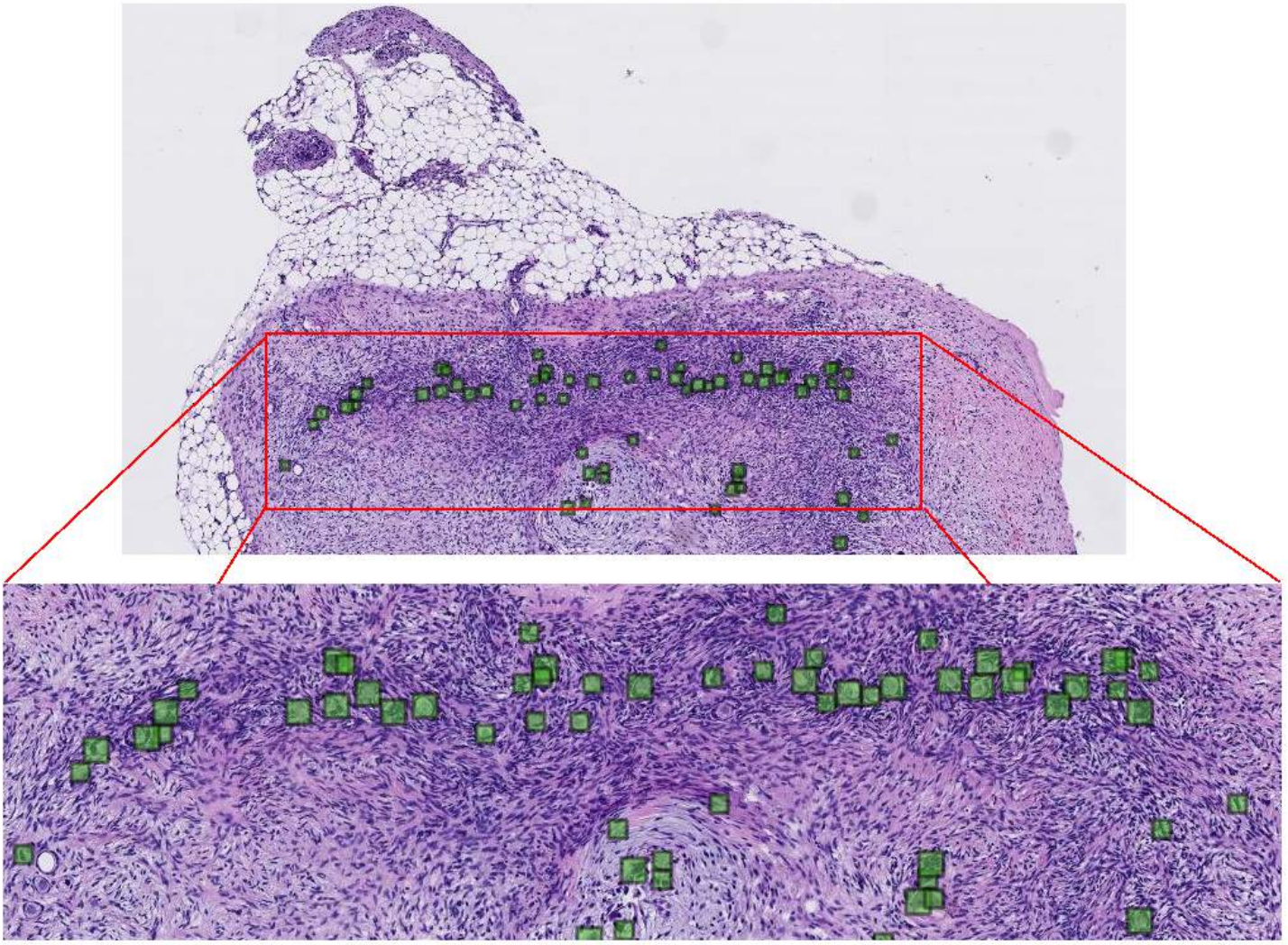


FIGURE 7.10: Résultat: La méthode détecte les follicules dans la région sombre. Les faux positifs sont toujours présents

est donnée dans le tableau 7.3. On remarque que le modèle détecte bien les cellules avec noyau: la plupart des follicules avec noyau sont classés comme tels par le modèle. Cependant, on remarque que deux follicules avec noyau sont classés dans la classe des follicules primordiaux sans noyau. La classification des follicules de cette région peut être jugée comme bonne, et ce malgré les probabilités qui sont un peu faibles. Ce fait est dû à la ressemblance des données des six classes sur lesquelles est construit le modèle, ainsi que le nombre de classes.

La classification des follicules de la région 2 est moins bonne que la précédente: des faux positifs sont à remarquer dans la classe des follicules primordiaux avec noyau.

région	Total (D)	primo_SN	prim_SN	sec_SN	primo	prim	sec
1	68	11	04	00	29	24	00
2	71	09	05	00	42	15	00

TABLE 7.3: Répartition par classe des follicules détectés dans les deux régions 1 et 2. Même légende utilisée que dans la tableau 6.3

Les résultats des tests de la région 2 sont montrés sur la figure 7.12.

Comme attendu, aucun des follicules détectés n'est classé dans les classes des follicules secondaires (sans/avec noyau).

On peut visualiser les résultats des deux opérations de détection et classification sur la figure 7.14, les follicules sont colorés suivant les colorations données à l'ontologie du projet dans la plateforme *Cytomine*.

7.2.1 Discussions

Dans le cadre de l'amélioration des résultats de la classification, on a essayé de contourner le problème des tailles des classes de l'ensemble d'apprentissage en dupliquant les exemples des petites classes d'une manière à avoir des tailles de classes identiques d'environs 250 exemples chacune. Les résultats obtenus avec cette solution (figure 7.13) ne sont pas encourageants vu le nombre de follicules classés comme "*secondaire sans noyau*" dans une région dont la surface ne dépasse pas 1242x794 px^2, ce qui est biologiquement anormal. Cette piste a été abandonnée.

Une autre solution envisageable -non testée-, c'est d'essayer d'augmenter le nombre de sous-fenêtres à extraire dans les images appartenant aux classes de petites tailles dans le but d'avoir un ensemble d'apprentissage plus homogène.

7.3 Temps d'exécution

Les temps exposés dans les tableaux précédents pour la détection sont les temps d'exécution incluant les temps des deux parcours et des téléchargements des tuiles depuis le serveur (plusieurs requêtes HTTP générées), les temps de prédiction,

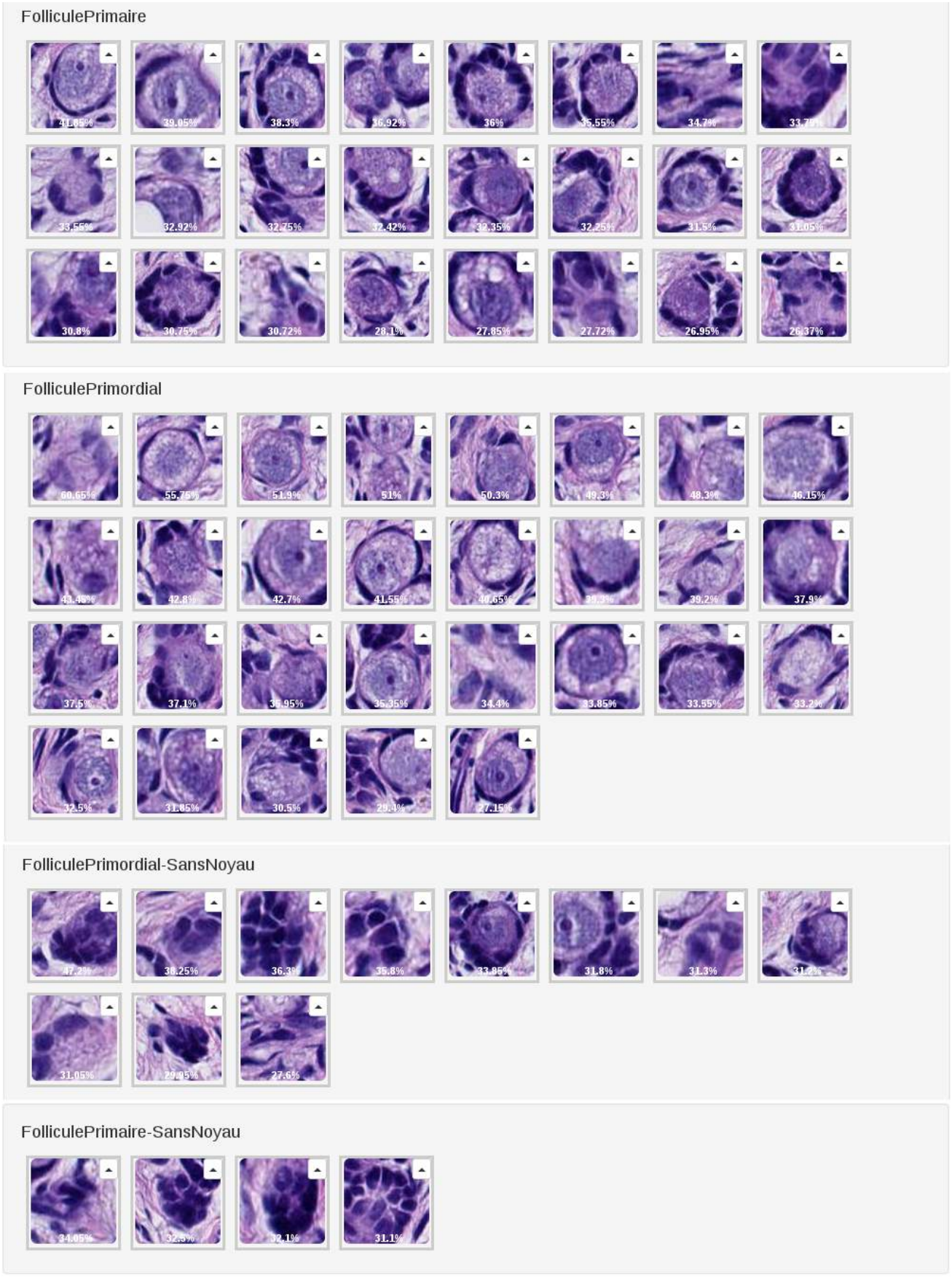


FIGURE 7.11: Classification des follicules de la région 1.

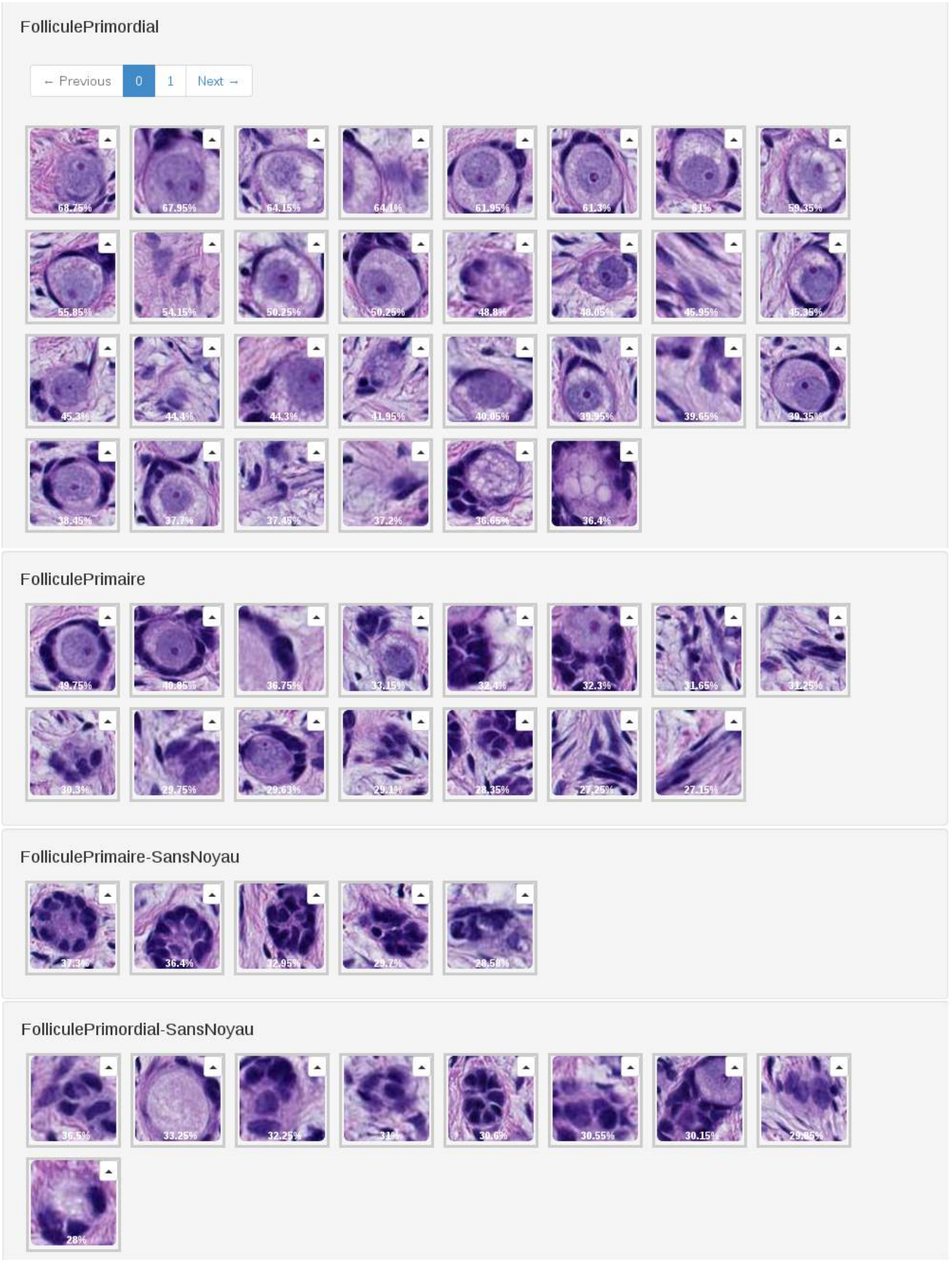


FIGURE 7.12: Classification des follicules de la région 2.

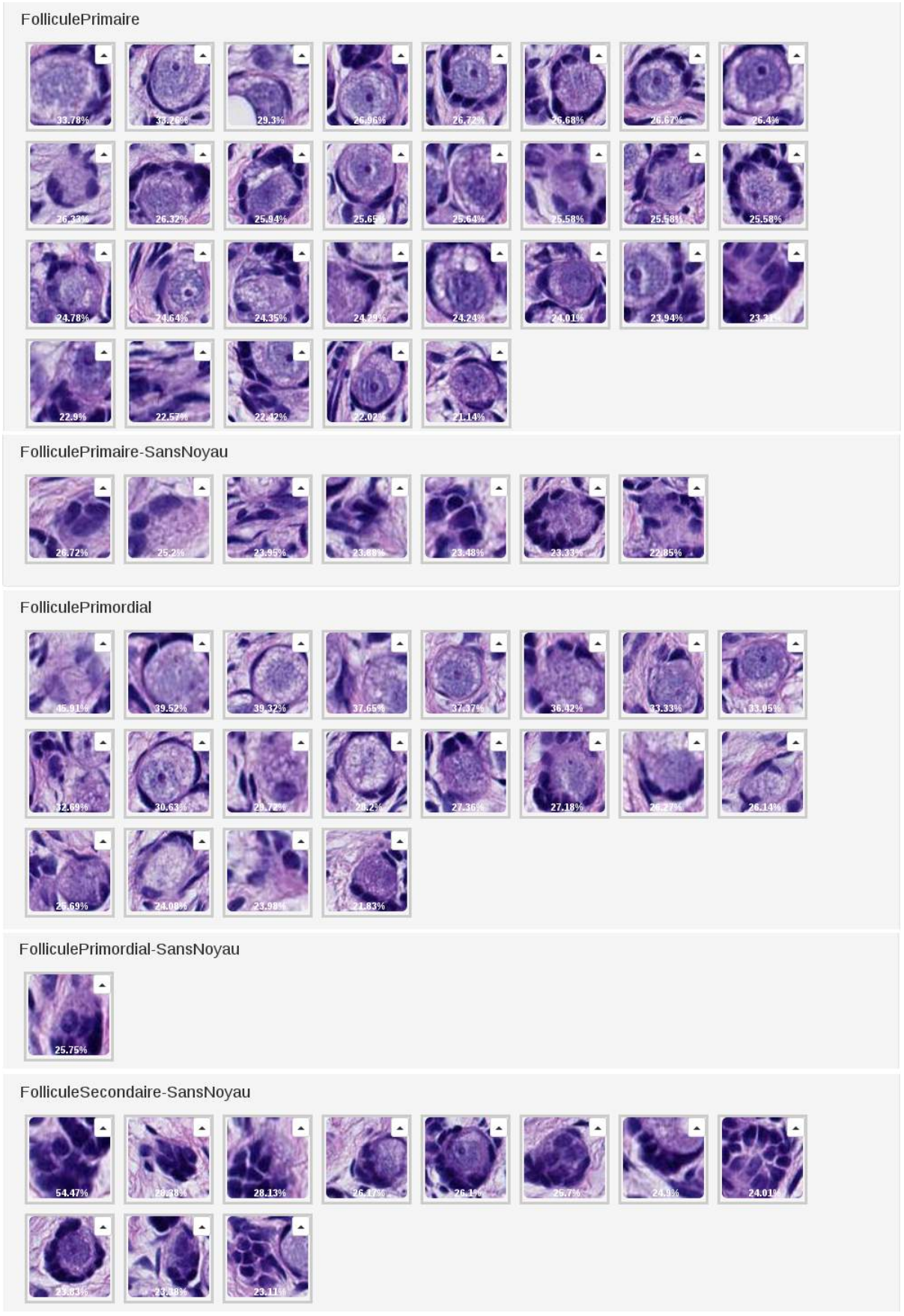


FIGURE 7.13: Résultats obtenus avec un modèle de classification construit sur base de classes de tailles identiques. Le modèle est construit avec les mêmes paramètres que le premier modèle de classification -section 6.3.4-, il est ensuite appliqué à la région 1.

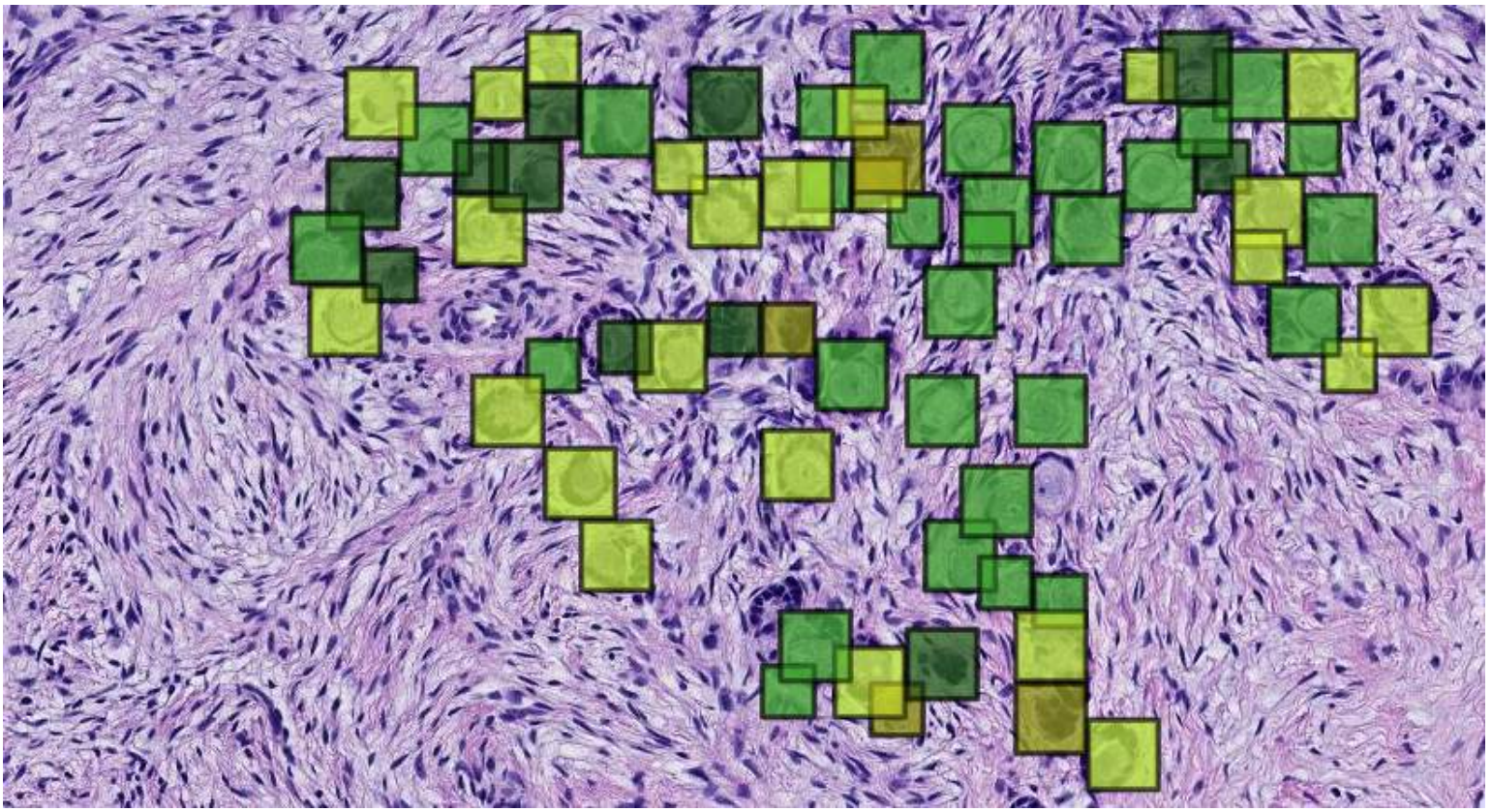


FIGURE 7.14: Détection et classification des follicules dans la région 1. jaune: follicules primaires avec noyau, vert: follicules primordiaux avec noyau, marron: follicules primaires sans noyau, et en vert foncé: follicules primordiaux sans noyau.

les temps de formation des clusters (récursion sur des listes), les temps de la procédure de merging appliquée aux résultats des deux parcours, et enfin les temps de l'*uploading* sur le serveur. La procédure de parcours ne se fait pas parallèlement, et la multitude des requêtes HTTP envoyés et les réponses reçues rendent ces temps considérables.

On peut améliorer ces temps en créant deux threads, chacun aurait la tâche d'un seul parcours, et les deux vont s'exécuter en parallèle. Pour les requêtes HTTP, on pourrait télécharger en une seule requête toute la région d'intérêt qu'on veut traiter (une grande tuile), et de procéder aux traitements en local.

Chapitre 8

Conclusion et perspectives

Deux problématiques de l'apprentissage supervisé ont été traitées dans ce travail: la détection des objets dans de grandes images histologiques, et leurs classification. Les données sur lesquelles on a travaillé sont des images d'annotations de follicules réalisées sur des lames histologiques des tissus ovariens. Ces données sont réparties en six classes, chaque classe représente un type de follicules qu'on pourra rencontrer au sein de l'ovaire.

L'objectif principal du projet est de mettre en place une technique de détection des follicules au sein des régions de coupes histologiques. Comme second objetif, on a essayé de procéder à la classification des follicules détectés. La méthode de détection mise en place s'applique sur des images sans aucun traitement préalable apporté à ces images comme c'est le cas dans la plupart des méthodes de détection.

Les objectifs fixés ont été partiellement atteints. La détection des follicules est encourageante. En revanche, pour la classification malgré les résultats obtenus jugés jusque là bons -sur deux régions de test-, les probabilités d'appartenance à une classe restent faibles dans la plupart des cas. La grande variabilité des annotations au sein d'une même classe, et la forte ressemblance des annotations de follicules appartenant à différentes classes, ainsi que la disporportion des tailles des classes sont les principaux facteurs influençant la qualité du modèle de classification, et du coûp les résultats obtenus. L'intervention d'un expert d'une part, pour ajouter des annotations pour les classes de petite taille à l'instar des follicules secondaires (sans/avec noyau), et d'autre part, la révision des annotations après une étape de classification afin de les corriger constituent une des solutions pour améliorer

la qualité du modèle de classification. Un bon modèle de classification donne des grandes probabiltés d'appartenance sans appel de plus de 90% parfois.

L'approche récursive du clustering utilisé pour former les groupes de tuiles dans l'étape de détection est contraignante en termes du temps et de mémoire consommée. Dans ce sens, les défauts de l'approche apparaissent dès son application à des grandes régions de plus de (6000x6000 *pixels*).

Les images histologiques fournies pour la plupart contiennent des régions sombres et des régions claires. Les régions sombres sont riches en follicules. À partir de ce constat, on pourrait améliorer les temps de détection dans une lame entière en ne parcourant que ce type de région. Ceci peut être réalisable, en ajoutant une étape de pré-détection qui consiste à détecter les régions sombres par application d'une segmentation par région par exemple, ou de construire un modèle binaire à partir d'un ensemble d'apprentissage contenant des images claires et des images sombres, dans ce cas le modèle aura pour têche de localiser des régions sombres. Une fois que la région sombre à parser est bien localisée, alors on pourra lui appliquer notre méthode.

Une de nos perspectives que nous envisagerons de faire dans le cadre de la généralisation de notre méthode à d'autres détections d'objets au sein des lames histologiques, c'est de l'appliquer au problème de détection des mitoses qui constitue un des sujets phares de l'application des techniques de l'apprentissage automatique dans ce genre de problème.

Enfin, nous pensons que l'amélioration de la méthode est envisageable. Le code source est disponible sur le site du projet à l'adresse:
`http://www.student.montefiore.ulg.ac.be/~debit/projet.php`.

Bibliographie

[1] J. W. Han, T. P. Breckon, D. A. Randell, and G. Landini. The application of support vector machine classification to detect cell nuclei for automated microscopy. *Machine Vision and Applications*, May 2010.

[2] R. Maree, P. Geurts, and L. Wehenkel. Towards generic image classication, an extensive empirical study. University of Liege, January 2013.

[3] R. Marée, P. Geurts, and L. Wehenkel. Extremely randomized trees and random subwindows for image classification, annotation, and retrieval. *Advances in Computer Vision and Pattern Recognition*, pages 125–141, 2013.

[4] M. Myers, K. L. Britt, N. G. M. Wreford, F. J. P. Ebling, and J. B. Kerr. Methods for quantifying follicular numbers within the mouse ovary. *Reproduction*, 172:569–580, 2004. URL http://www.reproduction-online.org/content/127/5/569.long.

[5] P. Geurts. Contributions to decision tree induction: bias/variance tradeoff and times series classification. University of Liege, May 2002. URL http://www.montefiore.ulg.ac.be/~geurts/thesis.html.

[6] R. Marée. Classification automatique d'images par arbres de décision. Université de Liège, 2005.

[7] P. Geurts. Ensembles d'arbres extrêmement aléatoires, application à la classification d'images. Université de Liège, Février 2004. URL http://www.montefiore.ulg.ac.be/~geurts/publications/Slides/geurts-seminaire-lip6.ppt.

[8] R. Maree, L. Rollus, B. Stevens, G. Louppe, O. Caubo, N. Rocks, S. Bekaert, D. Cataldo, and L. Wehenkel. A hybrid human-computer approach for large-scale image-based measurements using web services and machine learning. *Proceedings IEEE International Symposium on Biomedical Imaging*, 2014.

[9] J. S. Charleston, K. R. Hansen, A. C. Thyer, L. B. Charleston, A. Gougeon, J. R. Siebert, M. R. Soules, and N. A. Klein. Estimating human ovarian non-growing follicle number: the application of modern stereology techniques to an old problem. *Oxford journals*, 2007.

[10] P. B. Miller, J. S. Charleston, D. E. Battaglia, N. A. Klein, and M. R. Soules. An accurate, simple method for unbiased determination of primordial follicle number in the primate ovary. *Soc Study Reprod.*, 1997.

[11] D. Monniaux, A. Caraty, F. Clément, R. Dalbiès, J. Dupont, S. Fabre, N. Gérard, P. Mermillod, P. Monget, and S. Uzbekova. Développement folliculaire ovarien et ovulation chez les mammifères. *Inra Prod. Anim*, 22:59–76, 2009.

[12] F. Pellestor. Génétique, reproduction, et développement - histologie de l'appareil génital. *Faculté de Médecine Montpellier-Nîmes*, octobre 2006.

[13] J. R Tegnoor. Automated ovarian classification in digital ultrasound images using svm. *International Journal of Engineering Research and Technology (IJERT)*, 1, August 2012.

[14] B. Potocnik, B. Cigale, and D. Zazula. Automatic detection of follicles in ultrasound images of ovaries using active contours. *International Journal of Service Computing And Computational Intelligence*, 1:26–30, August 2011. URL http://storm.uni-mb.si/XUltra/.

[15] M. Fransolet, S. Labied, L. Henry, M. C. Masereel, E. Rozet, N. Kirschvink, M. Nisolle, and C. Munaut. Strategies for using the sheep ovarian cortex as a model in reproductive medecine. *PLoS ONE*, 2014.

[16] P. S. Hiremath and J. R. Tegnoor. Automatic detection of follicles in ultrasound images of ovaries. *Proc. 2nd International conference on Cognition and Recognition –(ICCR08), Mysore, India*, pages 468–473, 2008.

[17] P. S. Hiremath and J. R. Tegnoor. Automatic detection of follicles in ultrasound images of ovaries by watershed segmentation. *Proc. of International Conference on Systemics, Cybernetics and Informatics- (ICSCI09), Hyderabad, India*, pages 327–330, 2009.

[18] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, 8150, September 2013.

[19] R. Maree, B. Stevens, L. Rollus, N. Rocks, X. M. Lopez, I. Salmon, D. Cataldo, and L. Wehenkel. A rich internet application for remote visualization and collaborative annotation of digital slide images in histology and cytology. *BMC Diagnostic Pathology*, September 2013.

[20] P. Geurts. An introduction to machine learning. University of Liege -GIGA-, 2013. URL http://www.montefiore.ulg.ac.be/~lwh/AIA/.

[21] A. Deblire. Travail de fin d'études: Segmentation et classification automatiques de cytoponctions de la thyroide. Université de Liège, 2013.

[22] M. S. Hoque and M. C. Fairhurst. A moving window classifier for off-line character recognition. *Electronic Letters*, pages 628–630, March 2000.