Stage de fin de Master : TRaitement de l'Information et Exploitation des Données

Détection des parasites de malaria par microscopie

Nicolas Gourgue

Réalisé à :

Encadré par :

P. Horain et P.A. Pattanaik

Sommaire

Index des figures

Index des tableaux

Remerciement :

Thanks you D.A. Pattanaik for your help, your support.
Merci aux différentes équipe avec qui j’ai eux la chance de travailler, que ce soit l’équipe de Fontainebleau pour la collaboration dans la recherche de segmentation d’image. Leurs idées et nos échanges ont été d’une aide précieuse. Merci à l’équipe de TRIBVN pour sa disponibilité, son aide dans l’utilisation des données. Merci à l’autre équipe de Télécom sudParis qui nous a offert des échanges constructifs pour améliorer notre travail. Et merci à M.Horain, pour sa disponibilité, son aide dans les différentes tâches et problèmes que j’ai rencontré.

Introduction

1. Contexte

La malaria ou paludisme est une maladie qui est responsable du décès de 445 000 personnes dans le monde en 2017 d'après l'OMS[1]. Cette maladie est due à un parasite, le plasmodium qui contamine les globules rouges et les détruit. Pour la détecter cette maladie, nous utilisons un colorant et un médecin ou expert regarde les échantillons de sang pour dire si le patient est contaminé. On peut aussi détecter la maladie par réaction en chaîne par polymérase (PCR)[16]. Cependant cette méthode est très coûteuse et les principaux pays touchés n'ont pas les moyens de généraliser ce genre de méthode. Nous essaierons donc de développer un algorithme de détection/classification pour aider au diagnostic.

2. Équipes

Ce stage s'inscrit dans une collaboration inter-Carnot entre différentes écoles et une entreprise. L'entreprise travaille avec un hôpital pour obtenir des échantillons sanguins de patient infecté ou non par la malaria, qui contiennent des globules rouges, plaquettes et, le cas échéant, des parasites responsables de la malaria. Dans ce projet Inter-Carnot, nous collaborons avec une autre équipe de Télécom SudParis (Pr. Dorizzi, Pr. Gottesman, M. Bouchama) et une équipe du Centre de Morphologie Mathématique de Mines ParisTech à Fontainebleau (Pr. Angulo, M. Uyttenove) ainsi qu'une entreprise TRIBVN. Ce projet a débuté en janvier et se termine en décembre. Le but ici est de développé un outil d'aide au diagnostique, d'une part pour alléger le travail des médecins et d'autre part pour renforcer le diagnostique. Car si le médecin se trompe l'algorithme pour lui montrer son erreur dans le cas d'un oubli de parasite et si l'algorithme se trompe le médecin peut compenser derrière.

3. Tâches

L'équipe de Fontainebleau avait pour objectif d'automatiser la détection de la zone de bon étalement de l'échantillon sanguin. Lorsqu'on fait un prélèvement sanguin on étale la goutte de sang le long de la lame pour avoir une seule cellule d'épaisseur et ainsi visualiser correctement les globules rouges. Il y a des zones où les globules rouges sont bien étalés et des zones, notamment aux extrémités de la lame, où les érythrocytes sont en paquets. Une fois les mauvais étalements retirées de la base d'image, il faut segmenter les images afin d'entraîner un classificateur. Celui utilisé, à base de réseau de neurones, est ResNet 50 [10]. Cette classification fournit un pré-diagnostic sur la contamination du patient. Si l'algorithme a trouvé une cellule infectée, alors le praticien peut la visualiser et valider ou non le diagnostique de la machine. Dans le cas où l'algorithme ne détectera pas de parasites il faudrait prendre le résultat avec précaution : le patient sera soit sain soit peu infecté.
L'autre équipe de Télécom SudParis avait pour mission de reconstituer une image totalement nette à partir d'images de mise au point imparfaite. À partir d'images dont la distance au plan de netteté varie de – 6 à +6 microns, on reconstruit une image totalement nette, en intensité et en phase. Les images étant grandes, quand le focus est bien réglé au centre de l'image, il y a des zones floues en périphérie. Il existe actuellement une autre méthode mais qui est coûteuse en temps et en calcul. L'avantage dans

ce contexte des réseaux de neurones c'est qu'ils vont être très coûteux en temps de calcul pendant l'apprentissage donc dans la phase de développement, mais ils seront beaucoup plus rapidement que les méthodes traditionnelles dans la phase d'application. Ici le réseau utilisé est un auto-encodeur de type U-net [9].

Ma tâche fut de détecter les cellules et les compter en implémentant une méthode de segmentation, de détecter et compter les parasites, et déterminer leur espèce par classification. Mon stage se déroule du 1 juin au 30 novembre, l'intégralité des tâches n'ont donc pas pu être terminé. Nous ne disposons pas pour le moment de base de données labelliser en fonction de l'espèce de parasite.

4. Données

Dans un premier temps nous voulons détecter la maladie. Une utilisation de machine learning semble une bonne idée cependant nous nous heurtons à plusieurs problèmes de grande taille. Pour qu'un réseau de neurone apprenne à trouver et extraire l'information utile il lui faut d'autant plus d'exemples que l'image est grande. Nous disposons d'images de taille 2456 sur 2054 de cellules sanguines au grossissement x20, avec une résolution de 0,75 micron, acquises avec un éclairage de diode électroluminescente (DEL).

Nous avons un microscope avec un plateau de DEL en dessous. Les DELs se répartissent en une diode centrale et 4 cercles contenant respectivement 6, 12, 16 et 24 diodes. L'entreprise nous fournis des images prises avec la diode centrale allumé ainsi que le premier cercle et le troisième, ces images sont appelées br*ight field 11* (BF-11), un autre type d'images est celui avec juste le troisième cercle allumé, ces images sont appelées *dark field 8* (DF-8). Et nous avons aussi pour chaque image un fichier avec 35 images correspondant au 35 premières diodes allumées séparément, ces images sont appelées RAW n°DEL. Ces images sont fournies séparer par canaux (rouge, vert et bleu).

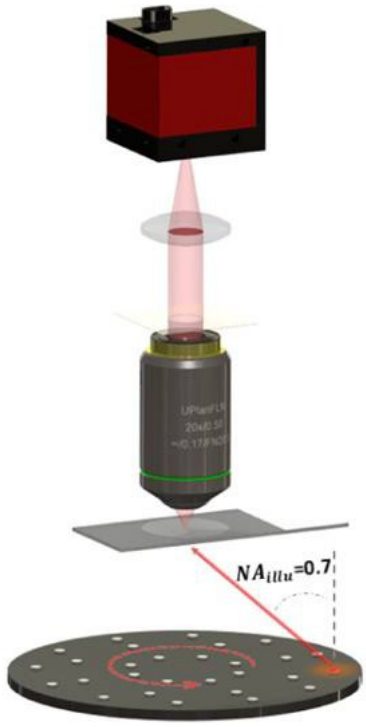

Figure 1: schéma des prises des échantillons

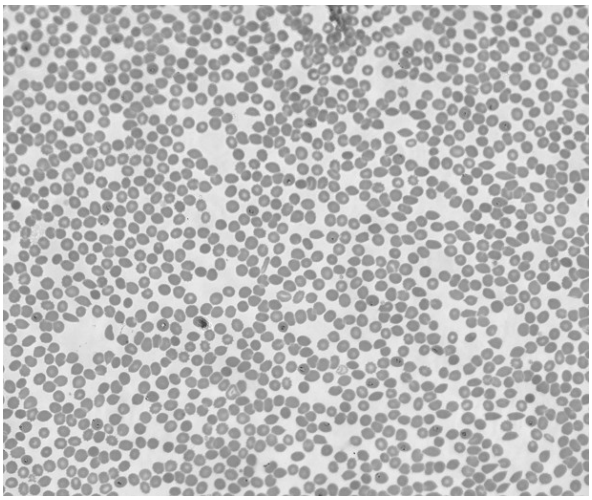

Figure 3: image canal vert, BF-11

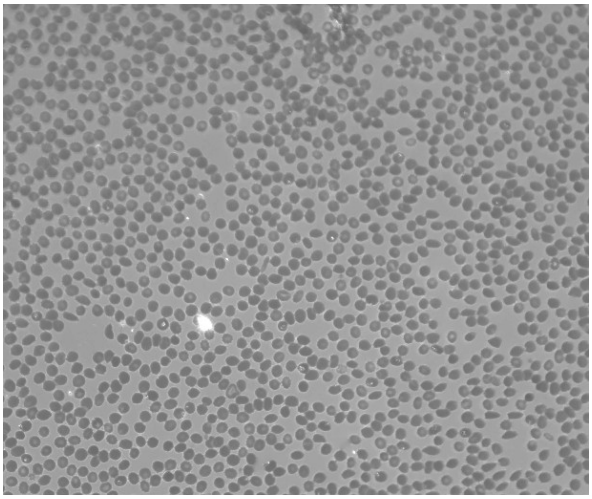

Figure 2: image canal vert, DF-8

Les parasites se trouvant presque toujours à l'intérieur de cellules, il est intéressant de segmenter les cellules pour diriger la détection.

Pour l'apprentissage, nous devons ensuite étiqueter (labelliser) chaque cellule comme malade ou saine. Plusieurs possibilités sont proposées dans la suite du rapport : une labellisation automatique, semi-automatique ou manuelle. Un algorithme de labellisation automatique utilise des critères propres à l'image. Une labellisation semi-automatique serait une labellisation automatique suivie d'un passage humain pour vérifier les images. Cette solution a été envisagée mais n'a pas été poursuivie, car elle s'est avéré chronophage pour des résultats insatisfaisants. Une labellisation manuelle fut donc retenue pour la suite de projet.

Enfin, il y a environs 50 cellules infectées sur 1500 cellules chez un patient malade. Le jeu de données s'en retrouve très déséquilibré. Nous devons donc trouver une solution pour contrebalancer ces écarts. Pour cela, les images de cellules infectées sont multipliés par des rotations et effet miroir. Cela permettra non seulement d'augmenter la robustesse de notre réseau aux variations par rotation mais aussi de rééquilibrer le jeu de données. Nous avons aussi dû nous confronter à la rareté des données car au mois de juin nous n'avions que 4 images. En juillet nous avons obtenons 3 autres images à titre d'exemple pour les différents type de patient :

- CAT01 : patient infecté avec de gros parasites.
- KPJ0 : patient infecté avec de petits parasites.
- DA : patient sain avec beaucoup de plaquettes.
- LE : patient sain avec un taux de plaquettes normal. (fourni plus tard)

I. Labellisation automatique

1. État de l’art

Nous avons dans un premier temps cherché une méthode pour segmenter de manière non supervisée les images, car nous n’avions que 4 grosses images et, même en les subdivisant, nous avons obtenu des résultats qualitatifs insuffisant pour une quelconque classification sur un auto-encodeur variationnel. Pour un réseau supervisé, il faudrait faire une segmentation manuelle pour pouvoir l’entraîner.
Partant de ces différentes contraintes nous avons trouvé un article publié en 2010 [2]. Les codes étant publié sur GitHub depuis 2015 nous nous sommes inspirés de leur approche tout en modernisant les techniques utilisées. Ce code très complet détecte 96 % des cellules et les classe comme infectées ou saines. Il s’est avéré que la détection par transformation de Hough est performante, mais la segmentation est limitée au cercle issu de la transformée de Hough. La classification s’appuie sur un seuillage dans le canal vert, or les images que nous avions au mois de juin était en niveau de gris. Nous avions donc de très mauvais résultats sur nos images. Nous avons donc choisi cet article comme référence en segmentation des hématies sans utiliser les réseaux de neurones.
Concernant les classifieurs, il existe une multitude d’articles ayant utilisé des réseaux de neurones. Nous prendrons pour exemple l’article publié par la Dr. Pattanaik au début de notre stage [15]. Elle a obtenu des performances en spécificité et sensitivité de 95 à 96 %. Nous allons donc utiliser différents réseaux. Pour la suite de ce rapport, j’ai utilisé le ResNet 101 [14] mais d’autres réseaux seront à l’étude d’ici la fin du stage.

2. Segmentation des hématies

Nous optons donc pour rechercher une méthode qui permettrait de segmenter les images en cellules automatiquement. Face à la tâche fastidieuse qui consistera découper à la main chaque cellule pour créer un ensemble d’apprentissage conséquent, sans aucune garantie, car nous avons à ce moment-là uniquement 4 images de tailles 2456×2054. Nous optons pour regarder du côté des méthodes classique de segmentation et de morphologie mathématique.

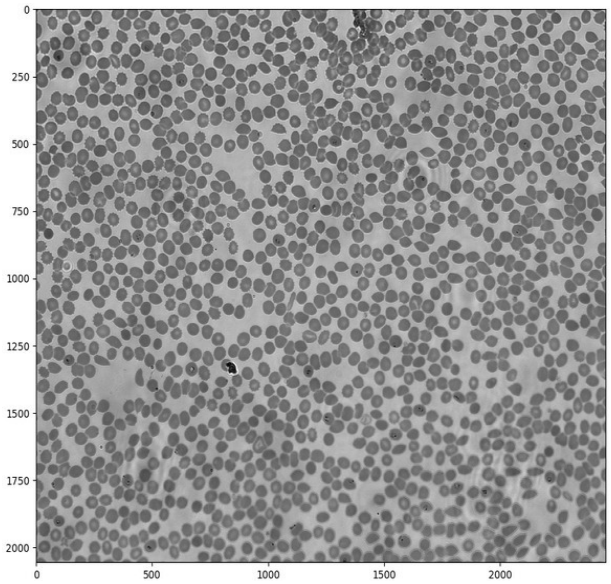

Figure 4: image originale

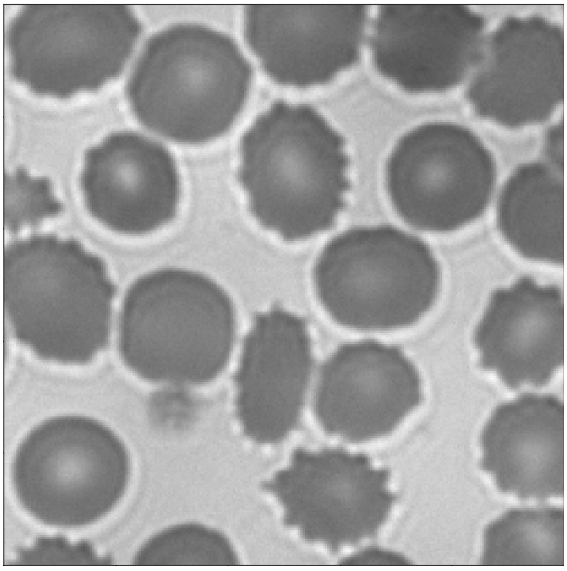

Figure 5: image zoom originale

Après plusieurs tentatives sur différentes images nous conserverons l'algorithme suivant. D'abord nous commençons par calculer le seuillage d'Otsu[6] localement sur des disques de rayon 4 fois celui d'une cellule, cela a pour but de s'ajuster au fond pour seuillée les niveaux de gris correspondant au fond qui peut varier au cours de l'image. Ensuite nous réalisons un bassin versant [3] en prenant les valeurs en dessous du seuil comme des cellules et les valeurs au-dessus comme du fond, cela permet de remplir la majorité des cellules creuses et d'éliminer les bruits du fond de l'image.

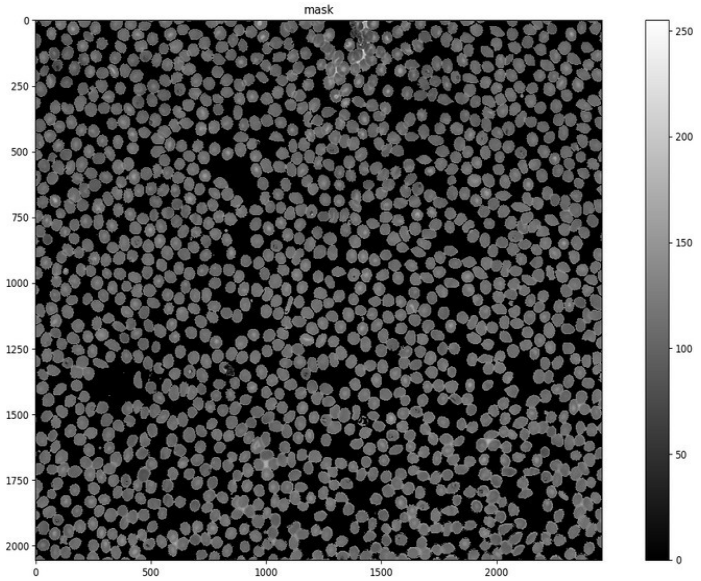

Figure 6: image avec le fond filtré

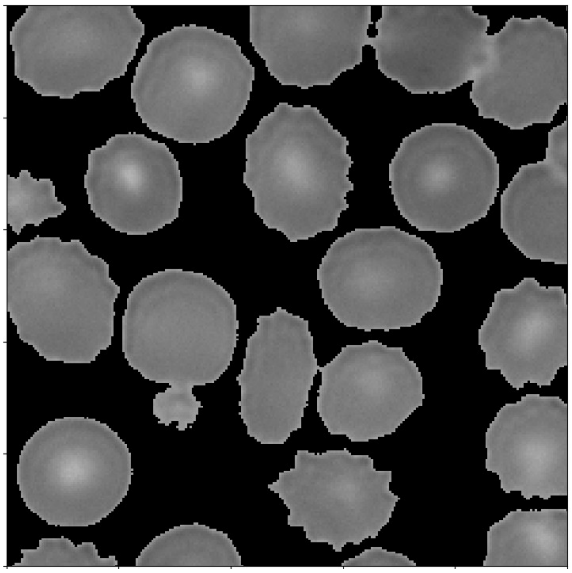

Figure 7: image zoom avec le fond filtré

Nous obtenons ainsi un masque binaire qui indique une grande majorité des pixels du fond. Il peut arriver qu'un îlot de cellule empêche le bassin versant de remplir une petite zone au milieu d'elles, mais cela reste anecdotique. Dans ces cas-là, la segmentation de la zone de fond sera soit répartie entre les cellules soit attribué à l'une d'elles.

Nous calculons ensuite avec la méthode Canny [5] les contours. La méthode Canny à détecter les contours de manière plus efficace qu'un filtre classique comme Sobel. Pour ce faire on s'appuie sur des

formules mathématiques (formule ci-dessous). Après on optimise cette réponse pour obtenir des contours fin. La méthode implémentée en python ajoute à cela un filtrage par hystérésis. Cela consiste à prendre un seuil haut et un seuil bas sur les contours de l'image. On prend tous les points du seuil haut et on prolonge les contours tant qu'il y a continuité et que les points sont au-dessus du seuil bas.

$$\sum(f)=\frac{\left|\int_{-W}^{0} f(x)\,dx\right|}{\sqrt{\int_{-W}^{+W} f'^{2}(x)\,dx}} \qquad \Lambda(f')=\frac{|f'(0)|}{\sqrt{\int_{-W}^{+W} f'^{2}(x)\,dx}} \qquad Maximisation(\Sigma.\Lambda)$$

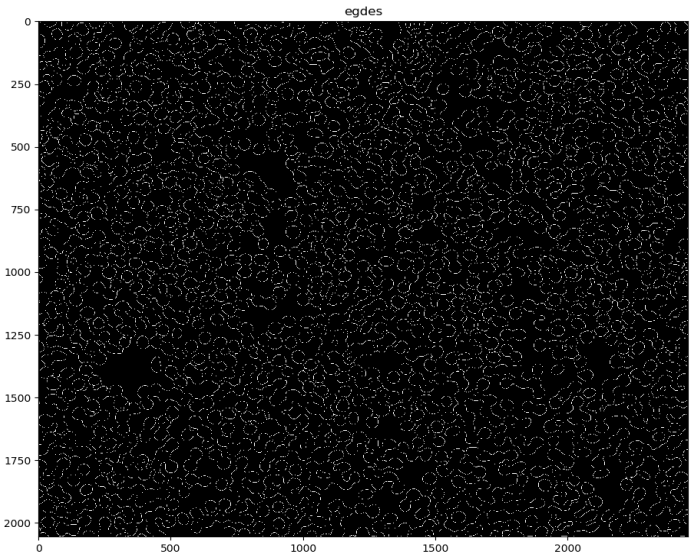

Figure 8: Contour calculé par la méthode Canny

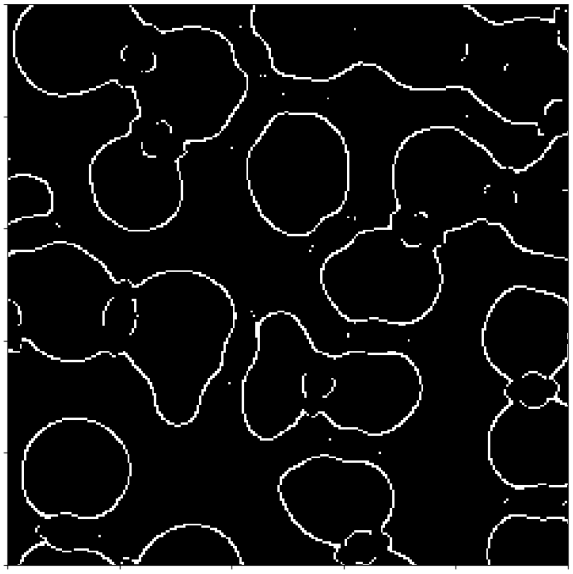

Figure 9: Contour zoom calculé par la méthode Canny

On voit sur la Figure 9 les contours des cellules. L'impression d'enveloppe en point tillé est due au sous-échantillonnage de l'image, mais les cellules sont pour la plupart fermée.

Ensuite nous allons utiliser la transformation circulaire de Hough [4] de façon itérative. Celle-ci permet de repérer les alignements de point dans une image, et a été adapté pour la détection de cercle. Pour rechercher les cercles dans l'image, chaque point de contour vote pour les cercles auxquels il appartient. On conserve/classe ensuite les cercles par leur nombre de votes reçus.

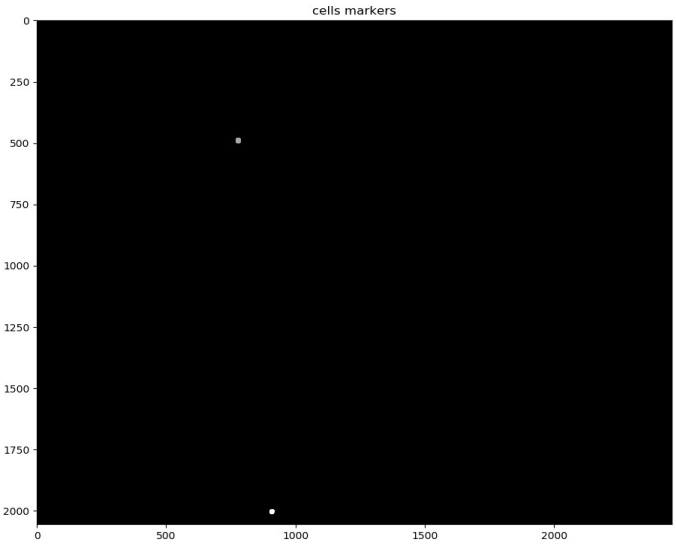

Figure 10: Marqueur placé sur les cellules détectées

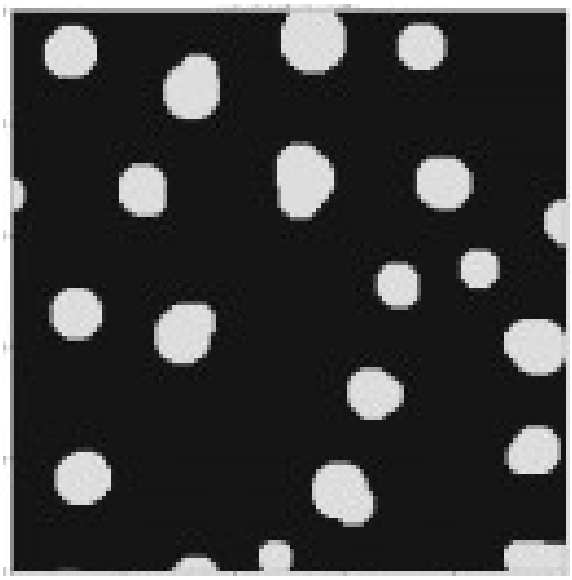

Figure 11: Zoom marqueur placé sur les cellules détectées

Nous sélectionnons les cercles qui ont les valeurs les plus élevées sur l'accumulateur. Cela a pour but de prendre que les cellules les plus sûres. Nous nous servons de cette détection pour placer un marqueur circulaire dans cette cellule de rayon la moitié du cercle détecté par la transformer circulaire de Hough.

Une fois les marqueurs placés, ils sont étiquetés. C'est-à-dire que chaque marqueur différent porte une étiquette (ici un numéro) différent. Ensuite après avoir éliminé le fond pour éviter les fausses détections, on réalise un bassin versant afin de remplir les cellules qui ne sont pas circulaires.

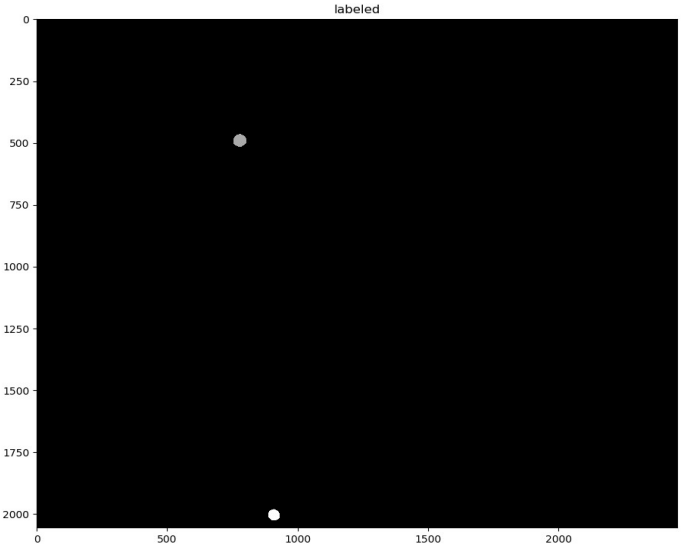

Figure 12: Cellules étiqueté

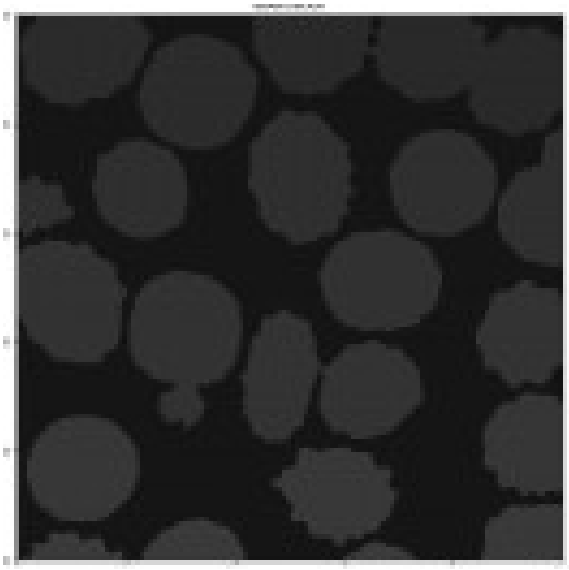

Figure 13: Zoom Cellules étiqueté

Ainsi, il suffit qu'une cellule soit en partie circulaire pour être détectée et marquée, puis le bassin versant complète la forme réelle de la cellule. Le masque de fond permet d'éviter les fuites qui agrégerait les cellules avec l'arrière-plan.

Cependant nous avons régulièrement des groupes de cellules qui apparaissent. Dès que deux cellules sont un peu jointive la transformation de Hough va en marquer une et le bassin versant va remplir les deux.

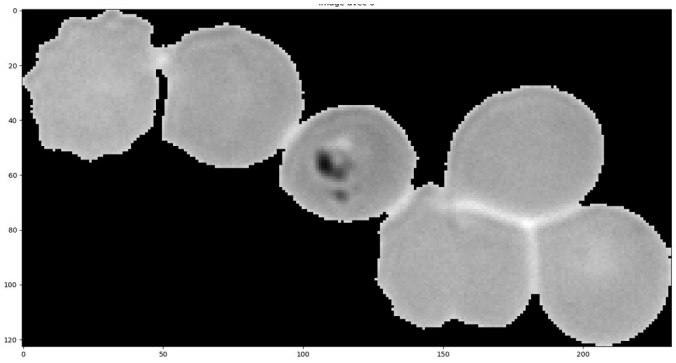

Figure 14: Amas de cellules

Pour corriger ce problème, nous utilisons un test pour savoir si dans les objets étiqueter certains sont des amas de cellules. Si la taille de l'axe principal (le grand d'axe de l'ellipse d'inertie) ou l'aire de la zone est plus grande que des valeurs de référence, la zone est considérée comme un amas de cellules.

Nous lui appliquons alors une méthode pour séparer les cellules, consistant à calculer sur la boîte englobante de l'amas (le plus petit rectangle droit contenant la forme) les contours par la méthode de Canny.

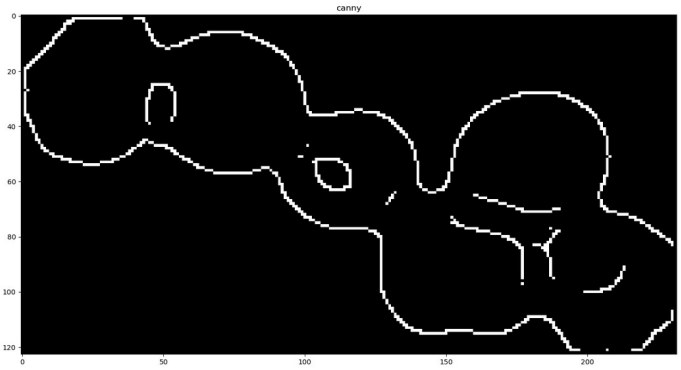

Figure 15: Détection de contour par Canny

Ensuite nous recalculons les transformations circulaires de Hough avec toujours une condition sur l'accumulateur mais cette fois nous n'utilisons pas de seuil pour choisir les cercles à conserver. Nous choisissons un nombre de cercle. Il est fixé de manière proportionnelle à l'aire de l'amas de cellule, avec en moyenne 5 cercles par cellules. Cela permet d'éviter d'avoir deux centres dans la cellule ronde et 0 dans la cellule elliptique. Nous fixons ensuite les marqueurs comme 0,2 rayon de cercle détecté par la transformée de Hough.

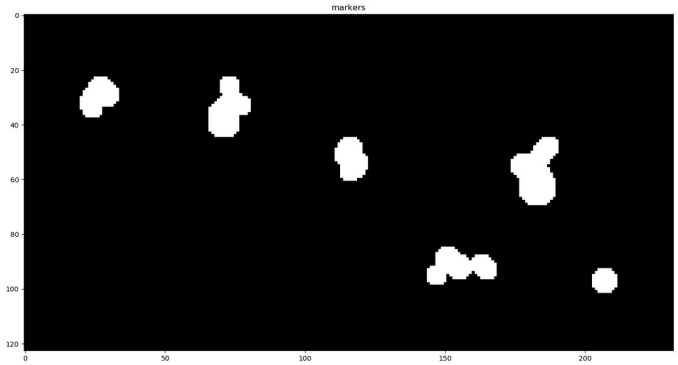

Figure 17: Marqueur des centres de détection

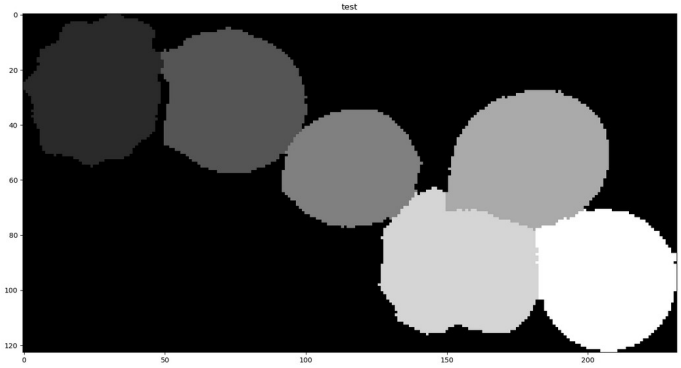

Figure 16: Bassin versant et étiquetage

Nous utilisons ce marqueur pour faire un bassin versant sur l'amas initial. Si nous arrivons à séparer convenablement les cellules, c'est-à-dire qu'elles passent toutes le test pour vérifier : qu'elles ne sont pas des amas et qu'elles ne sont pas mal découpées, c'est-à-dire qu'aucune cellule est coupé en petit bout. On conserve le découpage.

Si malheureusement le découpage se fait mal et que l'une des cellules déborde ou remplit partiellement une autre. Dans la Figure 18 cicontre on ne voit pas très bien les nuances de gris, car les labels ont des valeurs élevées mais proches (46 à 51 ci-dessus). Néanmoins ici les cellules 1 et 2 sont un amas qui n'a pas pu être séparé du reste.
Alors on supprime le découpage et on recommence avec des marqueurs de rayon 0.4, on recommencera une dernière fois en cas d'échec avec un rayon de 0.8. Si après toutes ses tentatives l'amas de cellule n'est pas détruit alors on l'efface et on continue l'algorithme. La plupart du temps l'amas se détruit à l'itération suivante.

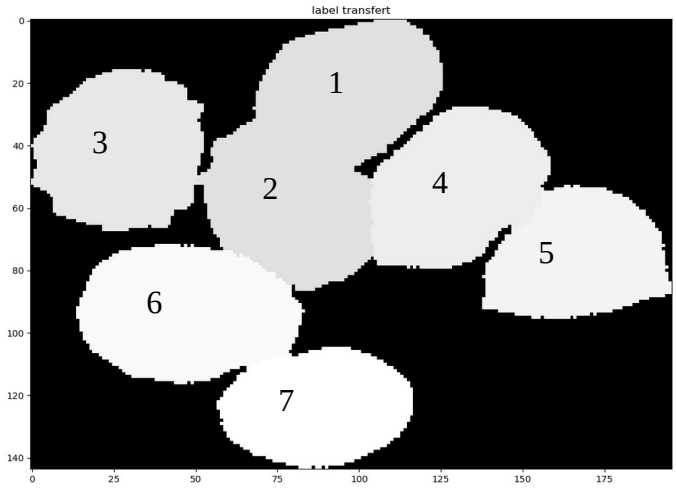

Figure 18: Découpage et étiquetage

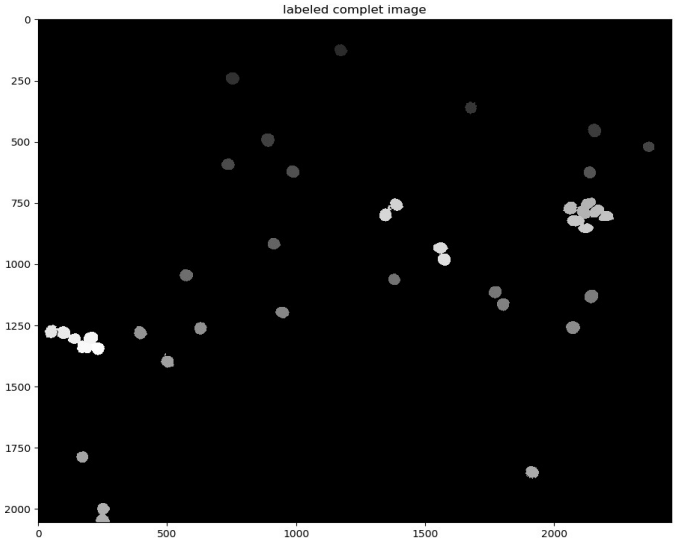

Figure 19: Étiquetage des cellules

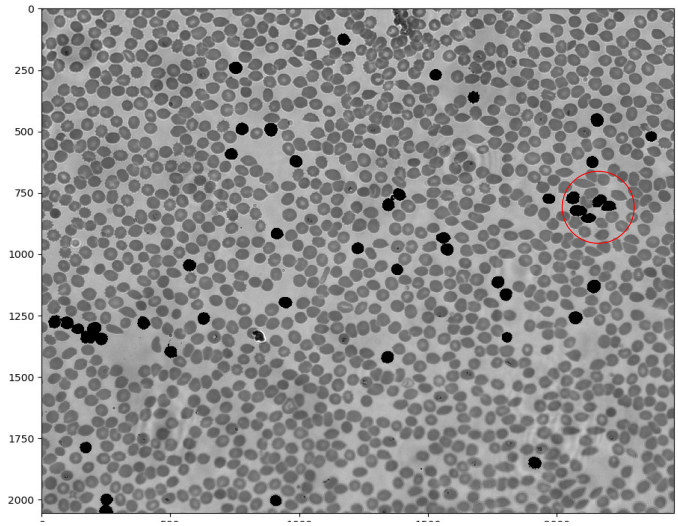

Figure 20: Image d'entrée de l'itération suivante

On remarque dans la Figure 19 que les cellules séparer ont bien été supprimé de l'image pour la détection suivante on remarque que dans notre amas mal découpé dans le cercle rouge, les cellules bien séparées ont été supprimées et celles qui sont restées en amas ont été réintégrées pour une détection future.

Nous obtenons dans performance de détection de cellule de l'ordre de 98 %, pour les patients infectés. Le calcul de cette performance se fait pas 1-la proportion d'erreur. La proportion d'erreur est calculé à partir du nombre de cellule détecté par l'algorithme et le nombre d'erreur est la somme de l'ensemble des cellules manquées (croix noir), des cellules mal découpées (croix bleu), des amas de cellules non séparées (croix rouges). Nous avons une légère tolérance pour les amas au bord car si l'une des cellules est coupé il est difficile de la détecter. Mais il nous a semblé important d'inclure un maximum de cellule, notamment car certaines au bord sont parasitées.

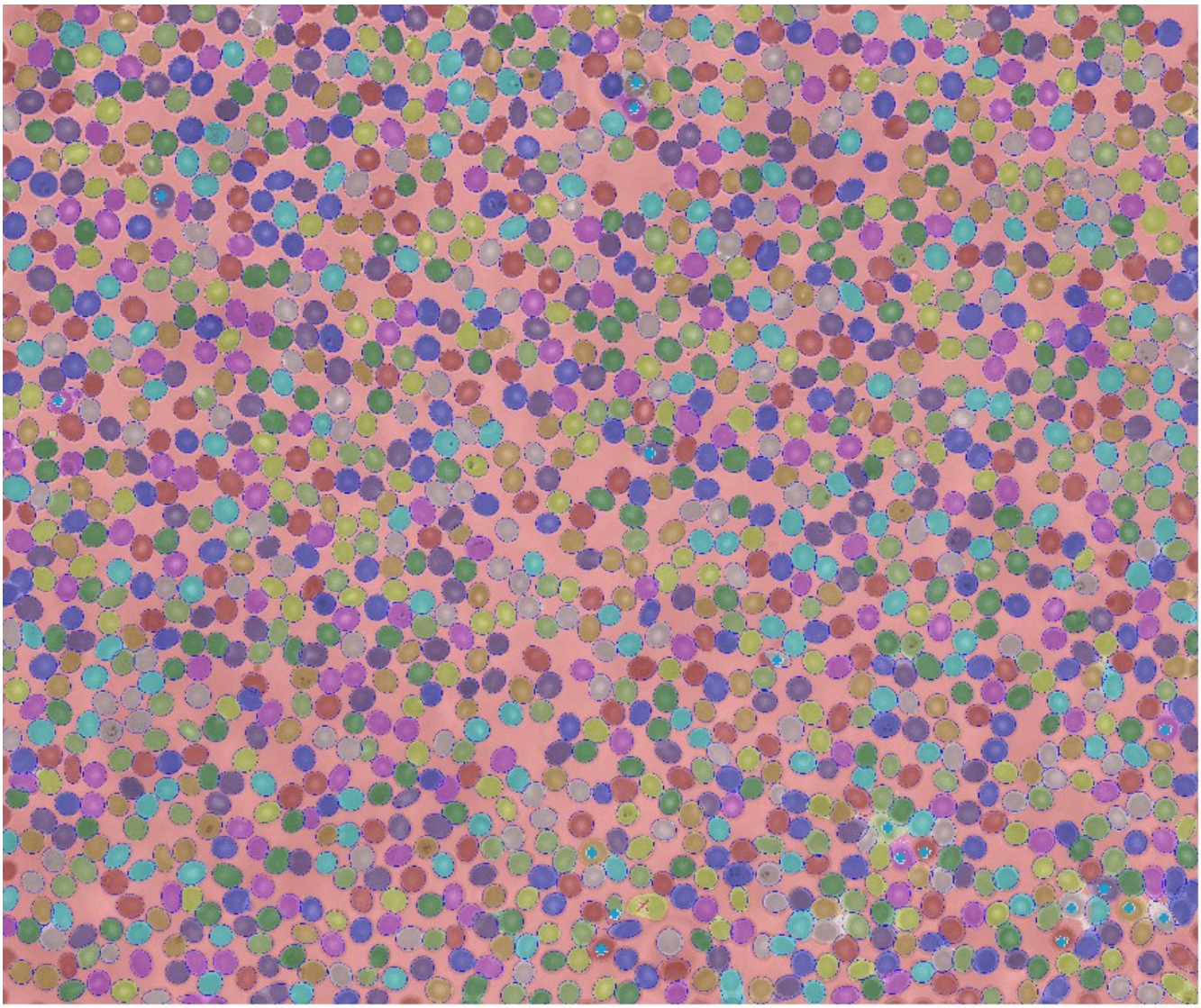

Figure 21: estimation des performances

Sur la Figure 21 nous avons fait des croix bleues pour les cellules mal découpées, croix noires (ici il n'y a pas de cellule manquée) pour les cellules manquées et croix rouges pour les amas non segmentés.

Cette méthode donne donc des performances satisfaisantes, car elles ratent peu de cellules. Cependant sont temps de calcul est long, de l'ordre de 4 minutes par image de 2456 × 2054. Nous verrons donc

s’il est possible de trouver une solution alternative pour pouvoir en faire une application industrielle (par exemple le mask R-CNN).

Par comparaison avec l’algorithme original cité en introduction nous avons de bien meilleures performances, car nous sommes pour les patients testés entre 98 % et 96 % de bonnes segmentations alors que « auto-count » lui était a 96 % de comptage mais ne segmentait pas correctement les cellules non-ronde.

Nous revenons ensuite au problème de labellisation comment définir si une cellule est infectée ou saines ? Nous avons pris la décision d’annoter à la main les images. Cette décision résulte du fait qu’il allait être long de mettre en place une technique dont les résultats devaient de toute manière être vérifiés à la main. Cependant l’entreprise pense que dans un projet avec plus de temps il pourrait être judicieux de développer un tel programme, pour faire un prétraitement et gagner du temps sur les parasites les plus évident lors de la labellisation.

3. Jeu de données

Nous avons donc segmenté avec cette méthode les 10 images du patient infecté avec de gros parasites (CAT01) et les 40 images du patient infecté avec de petits parasites (KPJ0), les 30 images du patient sains avec beaucoup de plaquette (DA), le patient sain avec un nombre normal de plaquette (LE), soient 110 images. Dans le tableau 1, nous constatons que nous avons près de 130 000 cellules dont seulement 3 100 sont infectées.

Patient	Canal	Champ	infectée	saine	ratio
KPJ0	Green, Blue, Red	BF, DF, RAW	2671	51688	19,35
CAT01			465	13278	28,55
DA			0	32149	NA
LE			0	30924	NA
TOTAL			3136	128039	40,83

Tableau 1: nombre de cellules

Nous voulons maintenant entraîner un réseau de neurones pour classer les cellules parasitées. Cependant nos données sont très déséquilibrées : nous avons 20 fois plus de cellules saines que de cellules infectées pour le patient KPJ0 et 30 fois plus de cellules saines qu’infectées pour le patient CAT01.
Plusieurs solutions s’offrent à nous. Dans un premier temps nous nous contenterons d’augmenter le nombre d’images de cellules infectées par symétrie et rotations. 15 pour CAT01 et 10 pour KPJ0. Nous avons aussi une différence d’effectif entre les cellules infectées et saines pour le patient CAT01 qui correspond à un patient atteint avec de gros parasites. Nous avons 10 images soient 450 cellules infecter et 13 000 cellules saines et 40 images pour le patient KPJ0 qui correspond à un patient atteint avec de petits parasites. Nous avons ensuite deux patients sains qui ont plus ou moins de plaquettes.
Une autre solution pour répondre au déséquilibre des données, peut être de pondérer la fonction de coût afin de pénaliser les erreurs sur les images parasitées classées comme saines.

II. Entraînement et réseau de neurones

Maintenant nous allons entraîner un classifieur de type Res-Net 101 pour tenter de classer les cellules. Notre réseau choisit est un Resnet-101. Le choix de cette architecture est dû à sa faible complexité. L'avantage des architectures ResNet est que le niveau de complexité est assez faible et qu'elles donnent de bons résultats.

1. Répartition des données d'apprentissage

Dans le but d'évaluer l'impact de l'augmentation de données et la pondération des classes. Pour cela nous allons comparer 2 apprentissages sur le même jeu de données. Nous nous limiterons au patient CAT01 et aux images RAW 0 (correspondant la diode centrale). Cela représente 450 cellules infectées et 13 000 cellules saines. Après augmentation nous obtenons 26 000 images, soient 13 000 d'hématies infectées et 13 000 d'hématies saines. Avec ces images nous constituons 3 ensembles : train, validation, test. Le dernier est séparé des deux autres manuellement avec un rapport de 80 % pour apprentissage+validation et 20 % pour le test. Ensuite on sépare l'ensemble apprentissage et validation dans l'apprentissage en entrant le ratio en paramètre. On prendra 80 % d'apprentissage et 20 % de validation. Soit au total 64 % apprentissage, 16 %, validation, 20 % test.
Nous ferons 2 apprentissages différents. Pour le premier différent, nous séparons d'abord les images en test et apprentissage+validation pour avoir des images de cellules infectées différentes et après nous mélangeons les données au sein des ensembles. Pour le troisième, les images sont mélangées et pondérées de façon inversement proportionnelle au ratio des classes.
Le réseau est déjà pré-appris sur la base ImageNet [7], nous avons donc une contrainte sur le type d'image en entrée, c'est-à-dire sur la dimension du tenseur. Nous pouvons choisir la taille d'image a 84×84 pixels que nous avons généré, mais nous devons fournir une image à 3 canaux. Nous donnerons dans cette phase de test 3 fois le canal vert ce qui constituera une image en niveau de gris. Le choix de ce canal résulte de l'a priori des experts pour la détection des parasites. Dans un premier temps nous lançons 100 époques.

		Matrice de confusion			
Nombre d'époque		100			
ensemble		different		ponderer	
		prédit		prédit	
% / classe		sains	malade	sains	malade
réel	sains	100,00 %	0,00 %	98,10 %	1,90 %
	malade	0,26 %	99,74 %	8,14 %	91,86 %
conditions		Dataset = CAT01, diode centrale, image G 3 canaux			

Tableau 2: performances sur 100 époques par ensemble

Nous obtenons des résultats satisfaisant avec plus de 99 % de bonnes classifications dans le cas de la répartition différente.

Les faux positifs (en orange) sont des erreurs bien moins graves que les faux négatifs (en rouge). Nous mettons en jaune les vrais positifs et vrais négatifs, car ils ne présentent pas d'intérêt particulier. Nous affichons les proportions totales et par classe.

Pour la première répartition des données d'apprentissage, les résultats sont assez encourageants. Nous avons peu d'erreur et le taux de faux négatifs est assez faible. Or c'est cette erreur que l'on veut minimiser. Ces performance s'explique par le pré-entraintement sur la base d'ImageNet.

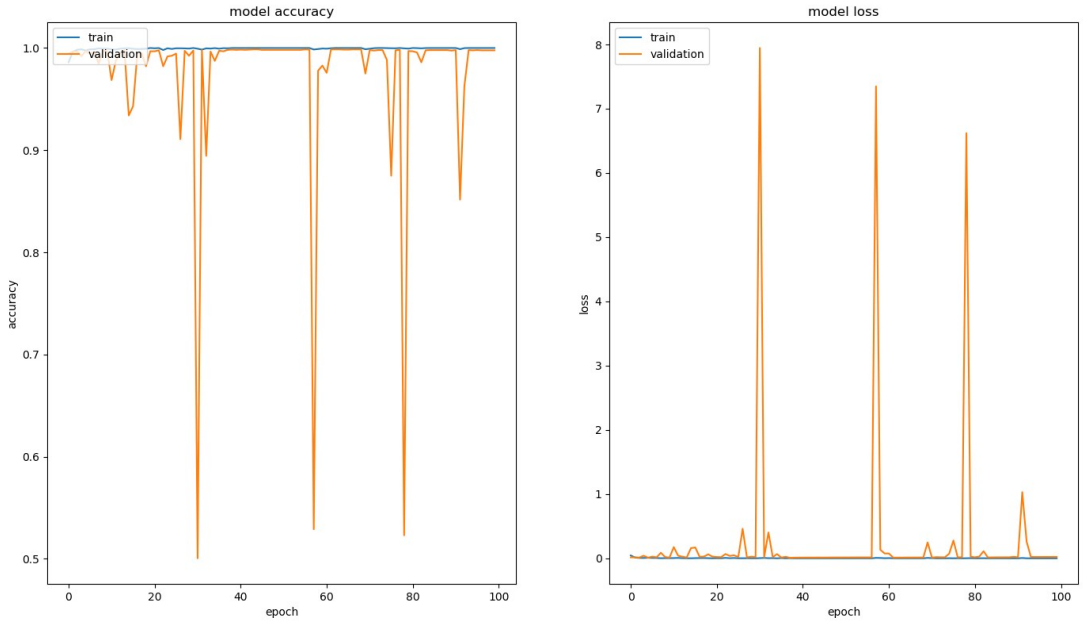

Figure 22: Erreur et Performance, pour 100 époques sur la répartition différente

Pour la première répartition, nous avons pris une image sur deux pour constituer la trame de nos ensembles et nous avons pris les 64 premiers % pour l'ensemble d'apprentissage, les 16 % suivant pour l'ensemble de validation, et les 20 derniers pour l'ensemble de test. Nous constatons que les performances ne sont pas significativement différentes. On peut supposer que notre réseau étant pré-appris sur image net nous permet d'extraire facilement les zones d'intérêt de l'image pour la classification souhaitée.

Nous remarquons dans la Figure 22 que la tendance est très stable jusqu'à l'époque 35, puis l'ensemble de validation connaît des pics réguliers qui semblent résulter du *Dropout*. Entre les époques 40 et 60, nous observons une stabilité. On pourra tout de même conclure que l'apprentissage semble se terminer.

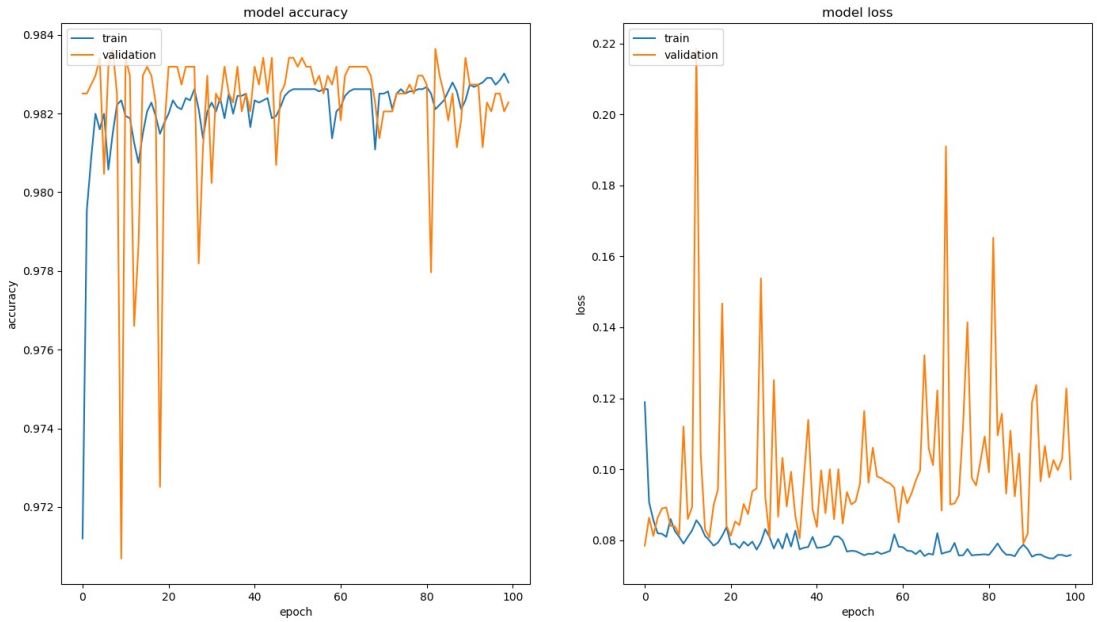

Figure 23: Erreur et Performance, pour 100 époques sur la troisième répartition

Avec la répartition pondérée, sans augmentation de données, l'apprentissage est beaucoup plus stable, sans pic extrême. L'augmentation de données pourrait accentué ces pics. Notons néanmoins beaucoup de fluctuation d'une époque à l'autre. Le *Dropout* en est responsable, car il ne permet pas l'apprentissage des poids à chaque batch. Cela a pour but de mieux répartir la décision sur l'ensemble des neurones de l'avant-dernière couche. Mais la conséquence est que lors de l'apprentissage lorsqu'on supprime le poids d'un neurone fortement contributif à la décision, les performances chutent. On note une tendance à l'augmentation de l'erreur de validation à partir de 40 époques, mais les fluctuations ne permettent pas de conclure clairement.
Nous considérerons donc que 50 époques sont acceptables pour l'apprentissage complet.

Les apprentissages sur ces répartitions semblent être terminés nous pouvons conclure sur les performances de notre réseau sur cet échantillon.
Nous avons des résultats meilleurs pour la première répartitions. Nous mettrons cette erreur sur le coup d'un ensemble de données trop petit. Cependant ses résultats sont à prendre a avec précaution, car ils sont issus du patient malade avec des gros parasites.

Nous allons donc ensuite garder la première répartition, car elle est meilleure que la deuxième. Les apprentissages semblaient être stables autour de 50 époques nous arrêterons nos apprentissages sur ce nombre d'époques.

2. Comparaison des canaux de couleur

Nous comparons maintenant les différents canaux de couleur. Les experts nous disent que le canal vert est le plus discriminant, car les échantillons subissent un traitement de coloration qui réagit avec le parasite pour lui donner une couleur violette. Or si le parasite apparaît violet c'est qu'il absorbe le vert. La cellule étant relativement claire avec le canal vert nous aurons les parasites en noir car absorbants dans ce canal, et les cellules en clair car non coloré. Nous utiliserons aussi les images RVB, car le pré-entraînement nous oblige à utiliser trois canaux. Nous comparerons donc la copie de chaque canal avec la combinaison des trois.

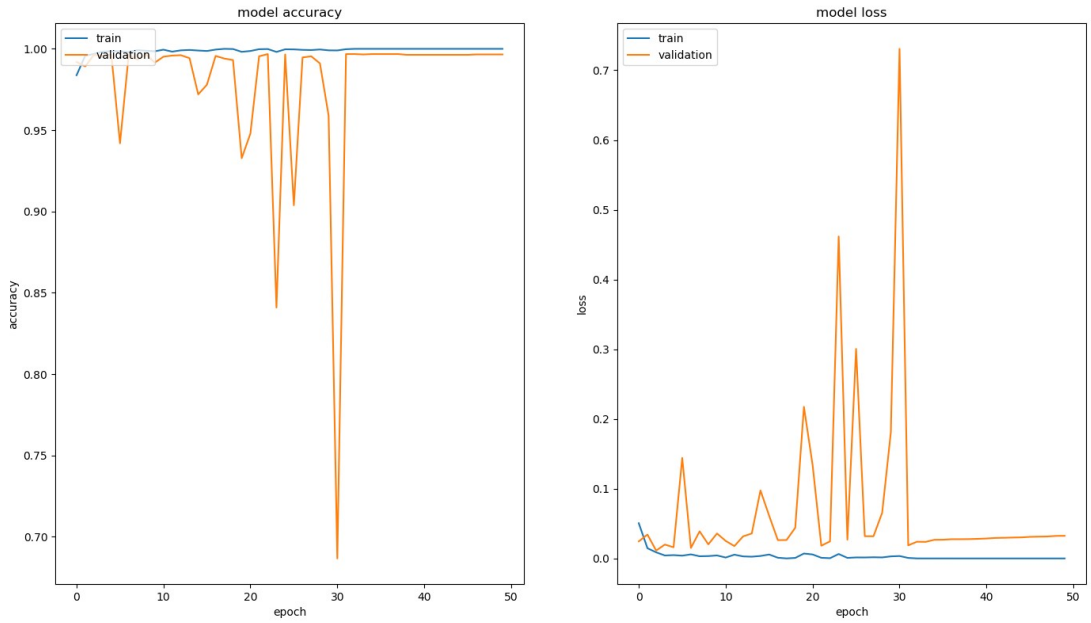

Figure 24: Erreur et Performance, pour 50 époques sur le canal bleu

Nous avons ici un léger sur apprentissage. On voit qu'à partir de l'époque 30 nous avons une augmentation de l'erreur sur l'ensemble de validation. Cependant les performances paraissent stables et la stabilité des performances de validation semble atteinte.

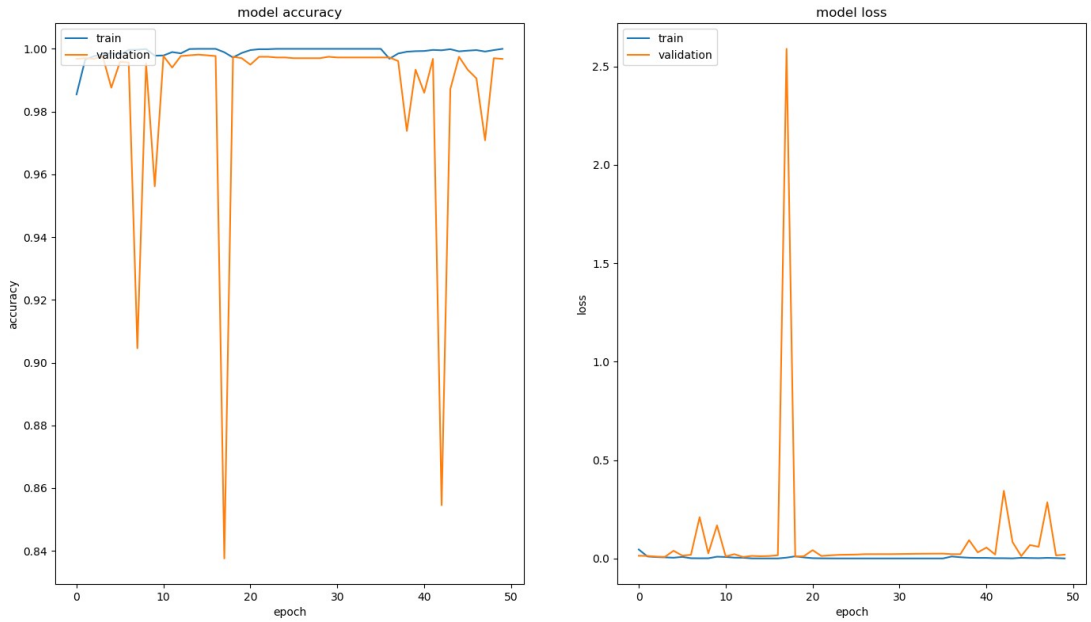

Figure 25: Erreur et Performance, pour 50 époques sur le canal vert

Nous avons ici moins de fluctuation, sauf à la fin de l'apprentissage. Nous espérons que cela ne dégradera pas nos performances.

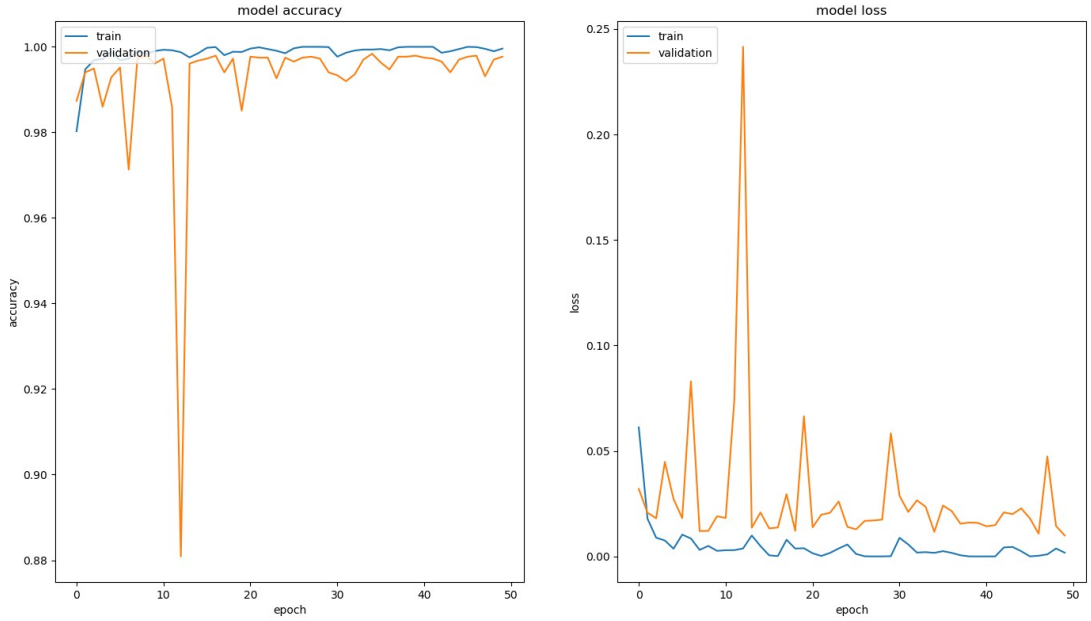

Figure 26: Erreur et Performance, pour 50 époques sur le canal rouge

Nous voyons Figure 26 que l'apprentissage est fluctuant et qu'il n'est probablement pas fini. Nous nous attendons donc à de moins bons résultats que pour les deux autres canaux.

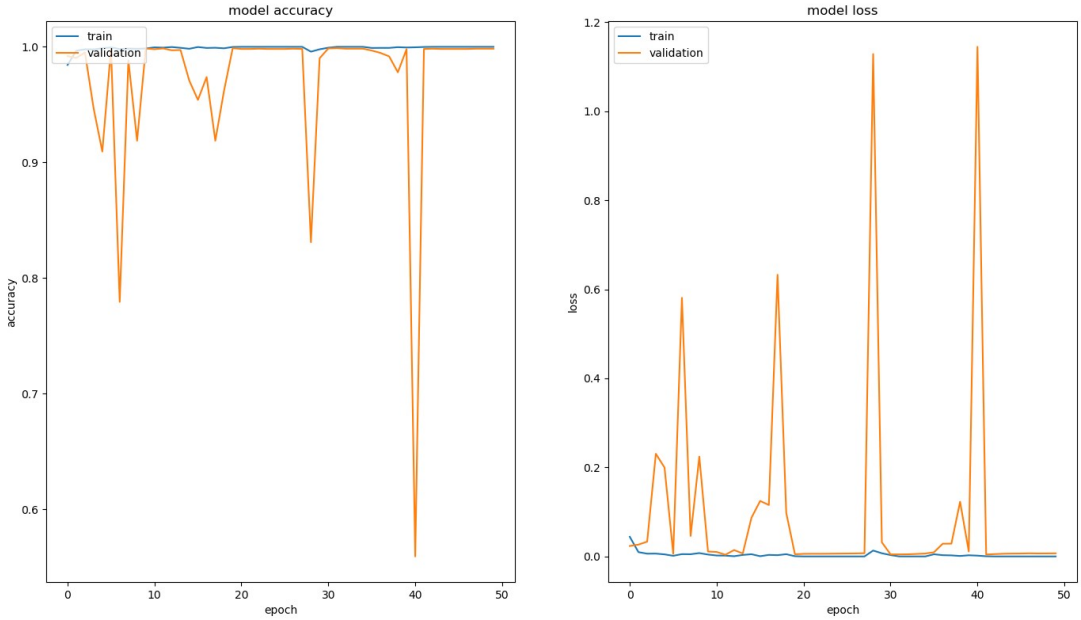

Figure 27: Erreur et performances, pour 50 époques sur les canaux RVB

L'apprentissage sur l'image en couleur semble terminé et les pics sont de même intensité que les deux premiers canaux.

		Matrice de confusion							
Nombre d'époque		50							
ensemble		rouge		vert		blue		couleur	
		prédit		prédit		prédit		prédit	
% / classe		sains	malade	sains	malade	sains	malade	sains	malade
réel	sains	99,44 %	0,56 %	99,66 %	0,34 %	99,44 %	0,56 %	99,89 %	0,11 %
	malade	0,29 %	99,71 %	0,07 %	99,93 %	0,29 %	99,71 %	0,29 %	99,71 %
conditions		dataset =CAT01, diode centrale, image 3 canaux							

Tableau 3: performances pour 50 époques par couleur

Nous voyons dans ce tableau les performances des différents canaux quand ils sont utilisés sur les canaux R, V et B séparément puis sur l'image en couleur. Nous ne notons pas de variation significative des résultats entre le canal rouge et bleu. Le canal vert lui, donne un taux de faux positifs plus élevé que l'image couleur, mais minimise les faux négatifs, objectif prioritaire dans notre projet. Nous conclurons donc que le canal vert est le meilleur canal pour classer les cellules en minimisant les faux négatifs. Ce tableau a été obtenu avec des apprentissages avec uniquement les données du patient ayant de gros parasites. Nous nous attendons donc à une baisse des résultats sur l'apprentissage global des malades.

Notons tout de même un point intéressant c'est que la minimisation des faux positifs peut s'expliquer par le fait qu'en image couleur le réseau fait plus facilement la différence entre une plaquette qui n'est pas teinté donc sombre, et un parasite qui lui est teinté en violet donc avec une différence significative entre les canaux.

Par la suite nous avons changé notre première répartition afin de la rendre plus aléatoire. Nous fixons un point aléatoire dans notre base d'image infecté et nous prenons les 64 % suivant pour l'ensemble d'apprentissage, les 16 % suivant pour l'ensemble de validation, et les 20 derniers pour l'ensemble de test ensuite nous prenons le même nombre d'image saines tiré ici au hasard car toutes les images sont différentes. La différence est minime mais la répartition change et il est donc moins probable qu'une configuration particulière sorte du lot et fausse l'estimation de la généralisation.

De plus nous avons changé la méthode d'apprentissage avant nous travaillons avec Keras et la méthode *fit* dans laquelle nous n'avons pas séparé manuellement les ensembles d'apprentissages et de validation. Par la suite nous utiliserons *fit_generator* avec laquelle nous séparerons manuellement les trois ensembles.

3. Apprentissage profond

Dans cette partie nous allons refaire une partie de cette étude mais sur un ensemble plus grand, nous allons ajouter les imagettes issus du patient avec des petits parasites, pour voir si les tendances se confirment. Nous effectuerons aussi plus de tests sur l'ensemble de pondération, car il y aura plus d'individu, on pourra donc mieux apprendre. Nous inclurons aussi les cellules des patients sains.

Nous tirerons pour la première répartition l'ensemble des cellules infectées et nous tirerons les cellules saines en prenant l'ensemble des cellules du patient avec de gros parasites, car nous en avons moins d'un quart (par rapport au nombre de cellules infectées après augmentation des images d'apprentissage) et le même nombre de cellules saines est tiré aléatoirement dans chaque autre patient.

Ici nous devons faire face à un nouveau problème. Jusqu'à présent les données rentraient dans la RAM et nous apprenions par batch sur la carte graphique. Mais ici les 16 Go de RAM ne permettent pas de

charger les ~130 000 images (50 images en moyenne, 1500 cellules par images, data augmentation, 50 cellules infectées en moyennes * 30 pour 10 images et *20 pour les 40 autres soit 3 Go bruts plus les données 9 Go pour le tenseur d'apprentissage). Nous devons donc faire un chargement des données par batch. Nous utiliserons laméthode *fit_generator* car plus simple d'utilisation.

Ici nous utilisons les deux patients infectées ce qui fait 5 fois plus de données nous avons fait une première vague d'apprentissage en se limitant à 100 époques, pour des raisons de temps. (10 à 12 heures). Il nous a paru judicieux de prendre des résultats sur 300 époques pour être sûr d'une part que les variations dues au *dropout* sont terminés et d'autre part pour être sûr que les apprentissages sont bien terminés. Nous n'avons pas eux le temps de faire les apprentissages sur les couleurs rouge et bleu mais aux vues des résultats précédents nous considérons que les meilleures performances sont sur le vert et la couleur.

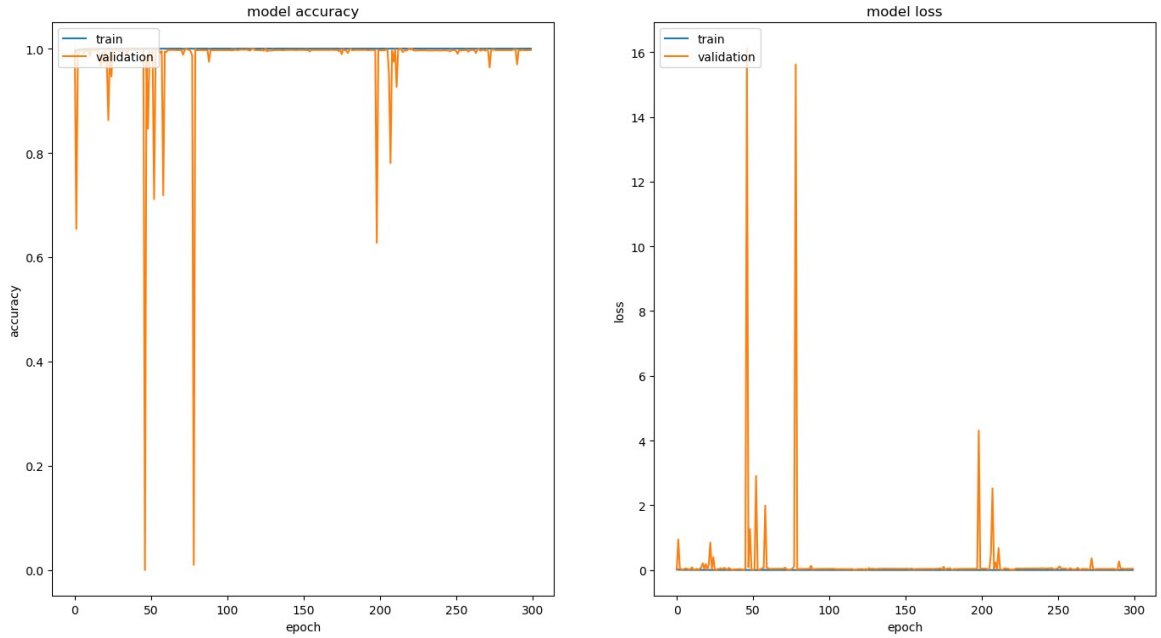

Figure 28: Erreur et Performance, pour 300 époques sur le canal vert

Sur la Figure 28 on voit que l'apprentissage terminé et stable. Nous avons ici un apprentissage plus stable mais une tendance au sur-apprentissage à la fin. Nous restons néanmoins au-dessus des 99 % de performances pour la classification.

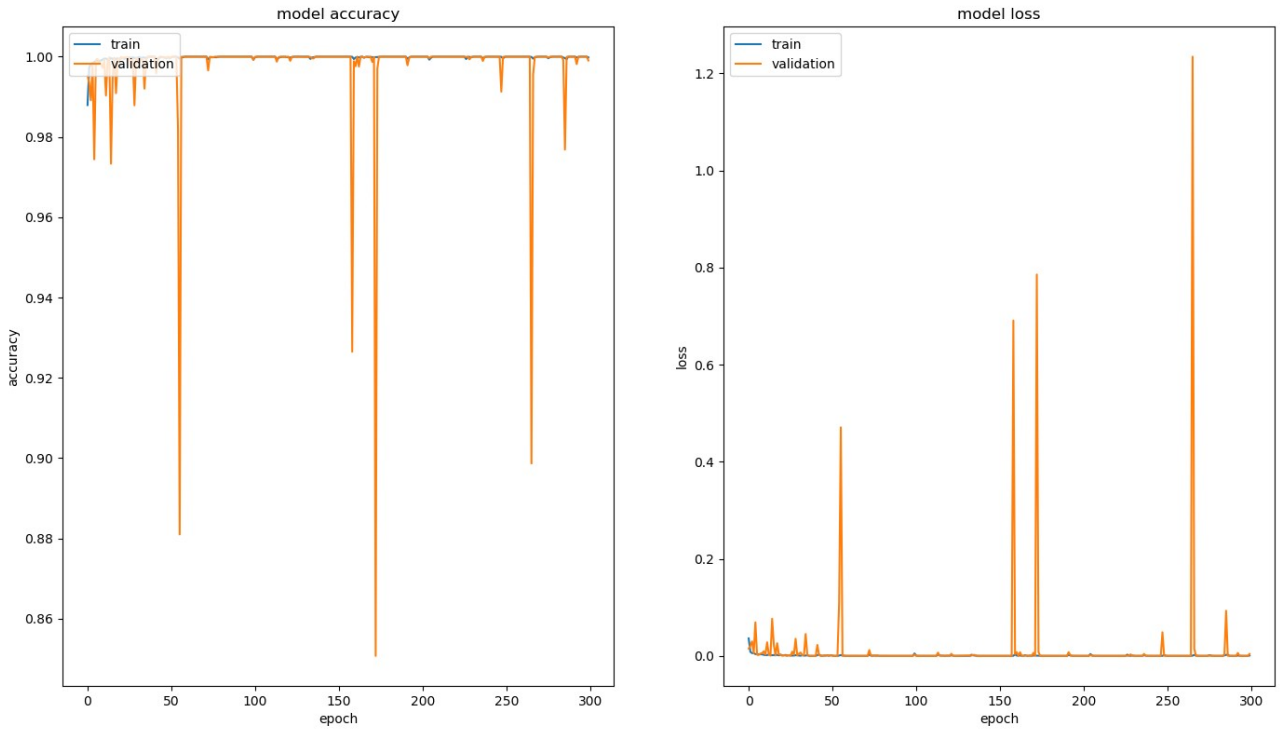

Figure 29: Erreur et Performances, pour 300 époques sur la couleur

Les courbes ressemblent beaucoup à celles du canal vert. L'apprentissage semble complet.

		Mat conf			
Nombre d'époque		300			
couleur		vert		couleur	
		prédit		prédit	
% / classe		sains	malade	sains	malade
réel	sains	99,85 %	0,15 %	99,73 %	0,27 %
	malade	0,51 %	99,49 %	5,07 %	94,93 %
conditions		dataset =tout , diode centrale, image 3 canaux, différent			

Tableau 4: Performances pour 300 époques pour le vert et la couleur

Nous avons ici une variation significatives des performances entre les canaux vert et RVB. On voit ici que le canal vert offre de meilleures performances. On peut penser que le canal rouge et le canal bleu ont gêné la classification pour la base RVB.

Nous comparons maintenant avec la répartition pondérée.

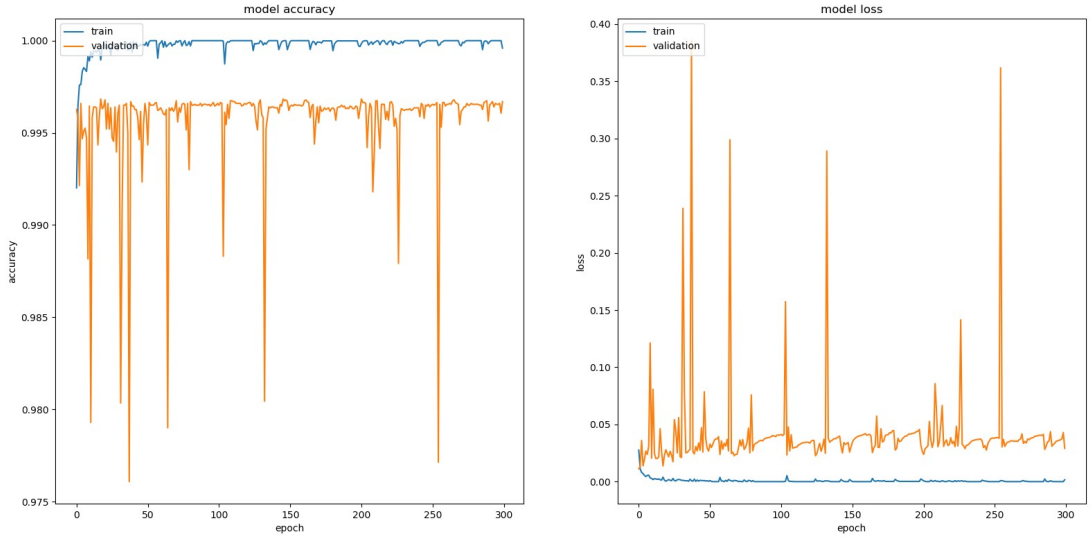

Figure 30: Erreur et Performances, pour 300 époques sur le canal vert

Nous voyons que sur la Figure 30 l'apprentissage est plus stable mais que des pics sont toujours présent.

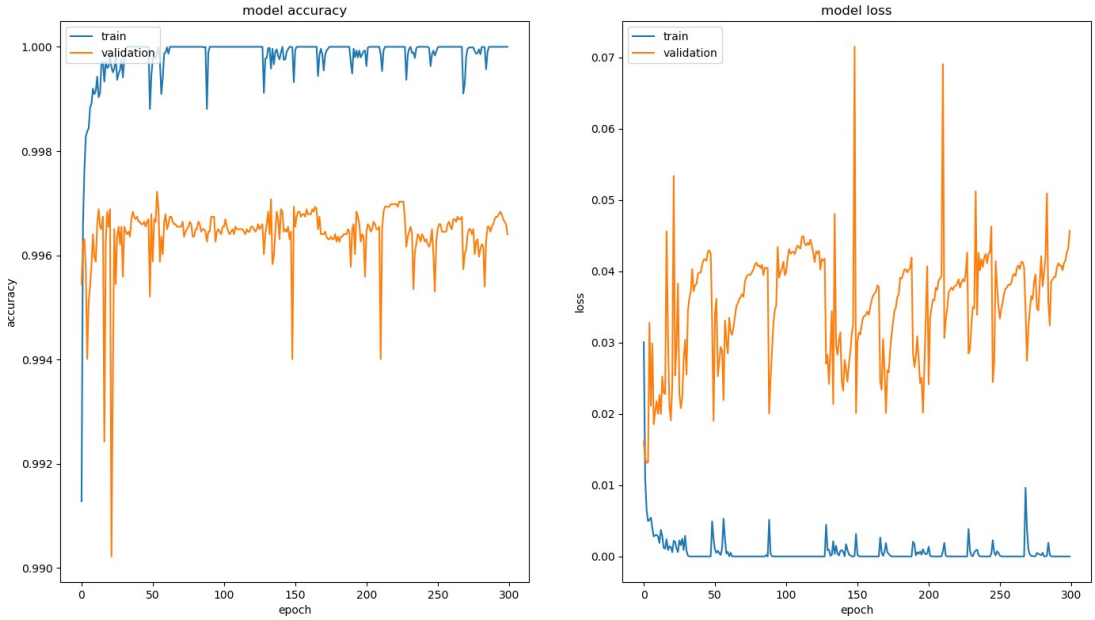

Figure 31: Erreur et Performance pour 300 époques sur la couleur

On voit dans la Figure 31 que l'apprentissage est beaucoup plus stable, les performances semblent se stabiliser à 99,6 %.

		Mat conf			
Nombre d'époque		300			
couleur		vert		couleur	
		prédit		prédit	
% / classe		sains	malade	sains	malade
réel	sains	99,82 %	0,18 %	99,89 %	0,11 %
	malade	8,16 %	91,84 %	11,62 %	88,38 %
conditions		dataset =tout , diode centrale, image 3 canaux, pondérer			

Tableau 5: Performances pour 300 époques pour le vert et la couleur

Dans le cas de la pondération on remarque que les faux positifs sont très dégradé par rapport à la répartition avec l'augmentation de données. Mais nous avons de meilleures performances pour la détection de faux positif. Nous expliquerons ses résultats par le fait que l'augmentation de données permet au réseau d'apprendre sur plus d'exemple d'image infectée et donc obtient de meilleures performances sur la classification d'image infectées et la répartition par pondération prend l'ensemble des cellules à disposition soit presque le double de cellules saines. Le réseau apprend donc mieux la diversité des cellules saines.

Il pourrait être intéressant de faire un apprentissage avec l'ensemble des données augmenter et faire une pondération pour inclure l'ensemble des cellules saines. Par manque de temps nous n'avons pas de résultat pour cette configuration dans ce rapport.

III. Résultat et amélioration

1. Bilan préliminaire

La constitution d'une base d'apprentissage correcte est une étape indispensable pour l'entraînement d'un réseau de neurone. La segmentation fut traitée par des méthodes morphologiques et transformation de Hough. L'idée d'une amélioration moins coûteuse en temps devra être étudiée.

Ensuite nous devons constituer les ensembles de test de validation et d'entraînement de manière aléatoire, avec augmentation des images. Sur les petits ensembles, l'augmentation de données s'avère plus efficace que la pondération, tendance qui se nuance sur les grands ensemble, l'erreur de faux positif est minimisé au détriment des faux négatifs. En effet l'augmentation de données permet d'obtenir des performances de 99 % des cellules bien classées, pour chaque classe. Cependant les contraintes réelles fixées par l'entreprise, sont de tester 200 000 cellules pour classer un patient donc pour que 90 % des patients sains soit classés comme sains il faudrait faire un faux positif sur 2 000 000. Cela pose 2 problèmes. Nous n'avons pas assez d'échantillons pour faire ce test et cela implique une précision extrême qui nécessitera un grand nombre de données. La condition du faux négatif paraît-elle moins contraignante. Car si 99 % des cellules infectées sont détectées, avec un taux de 1 % de faux négatif nous ratons 1 patient infecté pour 100 patients malades avec une seule cellule infectée. Ce qui signifie que ce patient ne serait pas grandement malade et donc qu'il pourrait être pris en charge plus tard sans que la maladie ne sont encore top avancée. Il faudrait donc concentrer nos efforts sur les faux positifs car dans la pratique ils sont plus contraignants pour le traitement des patients. Dans ce contexte, notre algorithme est non fonctionnel car il ne permet pas de considérer qu'un patient est sain. Nous devrons donc mettre l'accent sur cette problématique, soit en utilisant une augmentation des données des cellules saines, soit en ajoutant plus de cellules saines des autres patients, soit avec une pondération. Il est cependant intéressant de se poser la question des moyens engagés et mis à notre disposition.

L'utilisation de différent point de vue comme les images des autres DELs ou les champs appelés BF-11 et DF-8 peuvent aussi être un complément d'information pour améliorer les performances. Il faut faire attention cependant, car l'ajout d'information entraîne deux contraintes. La première est qu'il faudra plus d'apprentissage pour que le réseau situe et arrive à extraire l'information. Cela se comprend très bien, plus on met d'information à sa disposition plus il doit faire le tri de ce qui est important et ce qui ne l'est pas. La seconde est qu'il faudra plus de données, dans le même raisonnement le réseau a plus d'information, il a donc besoin de plus de comparaison avec différents exemples pour comprendre la donnée.

Dans l'optique d'améliorer notre classifieur nous pouvont envisager de faire du dopage (*Boosting*). C'est-à-dire régulièrement de faire une pause dans l'apprentissage de tester l'ensemble d'apprentissage sur le réseau et de pondérer les erreurs pour ainsi forcer les performances sur l'ensemble d'apprentissage. Bien sur il faudra vérifier que ce genre de pratique ne dégrade les performances sur l'ensemble de validation.

2. Autres architectures plus performantes

Nous n'avons ici présenté que les résultats sur l'architecture ResNet 101, il pourrait être intéressant d'essayer d'autre architecture plus complexe. Mon encadrante utilise une architecture VggNet 16 et une AlexNet. Ces architectures plus complexes ont l'avantage d'être beaucoup plus petites, on parle de ResNet 101 et VGG 16. Cependant nous ne pourrons comparer nos résultats dans ce rapport, car la mise en place des jeux de données c'est fait sur le mois d'août et de septembre or le Dr.Pattanaik est absente pour 5 semaines. Nous pourrions aussi utiliser une architecture GoogleNet avec ses modules Inception. Ces architectures étant plus complexes et moins profonde la comparaison devrait être intéressante. Tout dépendra du résultat du temps de calcul et des performances que nous voudrons avoir. Les modèles complexes présentent l'avantage d'être moins profonds et donc de nécessiter moins de données et moins d'époques pour apprendre correctement. Dans la situation dans laquelle nous nous trouvons, il pourrait être judicieux d'utiliser des architectures qui s'adapteraient mieux aux données. Cependant les architectures complexes vont nous contraindre davantage sur le format et le choix des images. Il faudra donc évaluer les avantages et inconvénients de chacun. L'étude de la parallélisation de l'architecture comme l'AlexNet pourrait donner une approche différente et des résultats intéressant tout comme les architectures GoogleNet avec les blocs inception qui parallélise les couches de convolution par blocs. Les architectures DenseNet [11] pourrait aussi être étudiée. Une architecture plus complexe pourrait donc extraire plus efficacement l'information avec moins de calculs. Car dans notre application le temps est un facteur important. Si nous avons 200 000 images à traité en un minimum de temps un gain d'une dizaine de micro-seconde par image serait un gain précieux. Au bout de 30 patient cela représente déjà une minute.

Nous pouvons aussi essayer des combinaisons de canaux, nous avons précédemment faire l'apprentissage sur les images couleurs qui semblaient le plus naturel, car d'une part le réseau est pré entraîné sur des images en couleur, mais on pourrait utiliser une combinaison de DELs, comme le BF, différentes (ici nous n'avons utilisé que la DEL centrale).

3. Réseau de neurone pour la segmentation

Nous avons ici fait le choix en juin d'utiliser un segmenteur basé sur des filtres morphologiques et la transformée circulaire de Hough car à l'époque nous n'avions que peu d'image et la labellisation à la main était inenvisageable car beaucoup trop longue. Mais maintenant nous avons 50 images de patients infectés et 60 images de patients sains. Nous avons donc 110 images. Nous pourrions compléter à la main, la segmentation mise en place précédemment et segmenter manuellement les cellules ratées ou les supprimer pour ensuite réaliser l'apprentissage de la segmentation. Il faudrait tout de même quelques milliers d'images pour essayer ce genre d'apprentissage. On pourrait couper des imagettes de 256×256 dans les grandes images avec un décalage de 128 pixels, ce qui nous ferait 15 images par lignes 19 par colonnes soit 285 imagettes par image, soient 31 350 images sur notre ensemble.

Un R-CNN pourrait être une méthode intéressante pour combiner la segmentation et la classification. L'intérêt est que notre algorithme de segmentation peut servir dans la création d'un ensemble d'apprentissage. Nous n'aurions plus les mêmes contraintes sur notre algorithme. Il devra être performant par image et non robuste pour fonctionner sur différentes images. Cela permettrait de déplacer le problème de la robustesse aux différentes conditions d'acquisition dans le réseau neurone. L'intérêt de ce type de méthode est aussi de pouvoir compléter la base d'apprentissage à la main. Cela permet de réduire considérablement les images mal ou non segmenter que notre base contient. Mes collègues Khalil ABID et Mahdi MASMOUDI ont eux aussi envisagé cette possibilité (en projet long du master). Ils se sont cependant confronté au même problème que j'ai rencontré en juin, un ensemble d'image trop faible et des données non étiquetées pour réaliser l'apprentissage. Cependant, au vu de la quantité d'image que nous avons, nous pouvons contourner ce problème. En ce qui concerne l'étiquetage, l'algorithme développé en première partie peut servir de base et être complété par un découpage à la main ou un ajustement des paramètres propre à l'image. Comme ici nous ne sommes plus dans l'optique de créer un algorithme robuste pour traité une grande quantité de données de la manière nous pouvons modifier les paramètres en fonction de chaque image afin d'obtenir un ensemble d'apprentissage parfait qui lui, permettra de créer un réseau de neurone qui pourra traiter de manière égale un grand jeu de données. Cependant nous n'avons toujours pas expliqué le fonctionnement du masque R-CNN. Un R-CNN, c'est un réseau qui à travers des convolutions va détecter et labellisée des zones d'intérêt. Dans un premier temps il détecte les régions d'intérêt comme un segmenteur. Ensuite il conserve les meilleurs boite englobante issu de sa segmentation, pour réaliser une classification et ainsi labelliser la boite. On parle ici de boite englobante mais le masque R-CNN peut fournir un masque au pixel près de la zone segmentée à partir du moment où l'on entraîne ce réseau sur des zones segmentées de la même manière. Cependant comme ce type de réseau en combine deux il est relativement long à la mise en œuvre. Mais il existe d'autre solution qui sont des Fast R-CNN et Faster R-CNN qui peuvent être des pistes intéressantes.

4. Meilleur ensemble d’apprentissage

L’entreprise qui fixe les contraintes du projet nous impose d’avoir un traitement qui doit aller de la prise de sang à la communication du résultat au patient en 2 heures. Nos algorithmes doivent traiter l’échantillon en moins d’une heure.

La contrainte de temps ici représente un gros problème, car mon algorithme de segmentation tourne en 4 minutes en moyenne par image, soit pour 1500 cellules, il nous faudrait donc analyser 140 images soit 9h20. Cela rajoute une raison supplémentaire d’entraîner un segmenteur de type R-CNN. Car il nous ferait gagner en précision et en temps. Cependant mon collègue du Centre de Morphologie Mathématique de Fontainebleau a développé un programme qui permet de détecter les images où les hématies sont correctement réparties et celles où elles ne le sont pas, ainsi que la segmentation des images en quelques secondes (15’’, cela ferait 35 minutes pour segmenter les 140 images). Cependant beaucoup de cellules sont encore en amas et il y a génération d’image vide. Cette méthode me paraît difficilement applicable. L’entreprise voudrait réduire le temps de traitement. Elle propose de prendre des images avec un zoom moindre et donc une résolution plus faible (x10 pour le zoom, et 0,45 micron pour la résolution). Nous devrons donc adapter nos algorithmes au changement d’échelle. Et re-générer l’ensemble de la base d’apprentissage. Cependant si nous obtenons les mêmes nombres d’images avec 4 fois plus de cellules, nous aurions un ensemble plus conséquent pour notre apprentissage. De plus les cellules seraient plus petites et même si les parasites aussi il sera plus aisé à l’algorithme de détecter même une petite zone sur des images plus petites. Mais cette piste n’est qu’à l’étude rien de concret n’a encore été fait en ce sens.

De plus, il faut un échantillon de 200 000 cellules pour que les tests statistiques de représentativité de l’échantillon soit vérifié. Nous prenons donc conscience à quel point notre ensemble de données est petit, car nous avons un total de 130 000 cellules différentes pour les quatre patients. Il y a la possibilité d’utiliser aussi des bases de données publiques pour avoir plus d’élément pour l’apprentissage [12]. Cette méthode a été envisagée et utilisé par mon collègue du centre de morphologie mathématique, mais les résultats qu’il a obtenus étaient assez faible moins de 90 % de bonne classification. Il faudra essayer avec notre base de segmentation pour voir si les résultats peuvent être meilleur. Mais les conditions d’acquisition semblaient suffisamment différentes pour que l’intérêt soit limité. Nous devrions néanmoins essayer d’évaluer nos performances sur cette base.

Il faut aussi garder à l’esprit les proportions. L’entreprise estime qu’il faut 200 000 cellules pour avoir une bonne représentation d’un patient et faire un diagnostic fiable. Nous n’avons pas 200 000 cellules de patient infectée, nous sommes donc contraints d’entraîner un réseau de neurone sur un ensemble qui ne respecte pas la diversité d’un seul patient. Il serait fort étrange que notre réseau donne des performances satisfaisantes pour une application médicale si l’entraînement n’est pas représentatif des situations à traiter.

L’idée d’utiliser aussi la ptychographie de manière indirecte pourrait être une solution. En tant normale la ptychographie consisterait à prendre les 35 images générer par les 35 DELs et en faire par une reconstruction une image de meilleur qualité. L’idée serait ici de mettre tout ou partie des 35 images en entrée comme 35 canaux d’une même image. La machine pourrait donc tirer parti des DELs les plus contributives et extraire l’information sans avoir à faire la reconstruction. Notons cependant que cela

demande de faire 35 acquisitions différentes qui aux vues des contraintes de temps imposées n'est pas forcément réaliste.

Conclusion

1. Problèmes surmontés

Nous avons tout au long de ce stage rencontré des difficultés qu'il a fallu contourner ou résoudre. Ces contraintes sont propres à la vie professionnelle. Que ce soit sur les contraintes matérielles RAM ou processeur graphique, processeur. Ici nous avions la chance de pouvoir travailler sur une machine équipée d'une 2080Ti et de 16 Go de RAM, Intel Core I7-3770K. Ou les contraintes logicielles comme dans l'implémentation des algorithmes fait en python alors que pour une application industrielle l'entreprise en a besoin en JAVA. Malgré ses contraintes nous avons trouvé des solutions en utilisant les données en batch avec keras, et comparant les codes sources pour obtenir les mêmes résultats avec le bassin versant en JAVA. Nous avons aussi vu qu'il faut s'adapter au projet car ayant reçu une formation générale en data science je n'ai pas suivi les modules de traitement d'image. J'ai donc dû apprendre et comprendre rapidement ses concepts afin de développer un programme s'appuyant sur ces différents procéder.

2. Contrainte temps respectée ?

Pour l'instant mon segmenteur ne permet pas de respecter la contrainte de temps. Nous pouvons espérer qu'un réseau de neurone donnera des résultats similaires si ce n'est mieux en un temps plus faible. Ou utilisé l'algorithme de notre collègue de l'institut de fontainebleau, réduire la taille des images sont autant de possibilité qui s'offre à nous. Je pense que la solution la plus prometteuse pour notre application reste un réseau de neurone pour la segmentation, car comme expliqué précédemment il permet de déplacer le temps de calcul en amont de l'utilisation, c'est-à-dire lors du développement. Un réseau de type R-CNN me paraît donc la meilleure piste pour répondre à la contrainte de temps fixé par l'entreprise tout en conservant des résultats acceptables pour une application commerciale.

3. Contrainte faux négatif respectée ?

La contrainte sur le nombre de faux négatif est assez sévère, car elle ne veut pas de faux négatif. Cette position peut se comprendre, car un patient malade que l'on renvoie chez lui incube la maladie et les soins sont plus compliqué, coûteux et le patient augmente ses risques de décès. Cependant un algorithme qui classe tout le monde comme malade obligerait le médecin à vérifier sans cesse et n'aurait aucun intérêt. Nous devons donc obtenir une erreur de faux positifs quand même raisonnable. Les résultats préliminaires sur le patient atteint de gros parasites semblait encourageant. Seulement 0,07 % des parasites était manqué. Cependant avec le patient atteint de petit parasite il est plus compliqué d'obtenir de si beau résultat nous obtenons en première approche 0,4 % d'erreur ce qui fait un parasite de manquer sur 250. Cela reste quand même acceptable car pour 250 patients venant consulté avec seulement 1 parasites dans l'échantillon prélevé nous en déclarions 1 seul comme sain.

Avec un seul parasite dans l'échantillon de 200 000 cellules on peut supposer qu'il sera à un stade précoce de la maladie. Bien cette extrapolation est à prendre avec du recul, car il est évident que les jeunes parasites plus petits sont plus souvent loupé que les vieux. Mais cet exemple est là pour placer l'erreur dans un contexte et considérer si elle est acceptable. Cependant la question du faux positif mérite qu'on s'y intéresse. Car comme énoncé précédemment il faudrait des taux extrêmement faibles pour arriver a une erreur acceptable.

4. Solution utilisable dans le monde professionnel ?

Au vu des résultats et des différentes contraintes, notre méthode n'est actuellement pas applicable pour un usage industriel. Nous avons néanmoins des pistes pour répondre à la problématique de temps et la contrainte de faux négatifs semble atteignable. Le problème actuellement rencontré comme énoncé dans une partie précédente. Est le nombre de faux positif car s'il apparaît faible dans un premier temps et pourrait donc satisfaire une recherche purement académique nous avons ici 0, 11 % d'erreur (dans le meilleur des cas) ce qui est largement au-dessus des $5*10^{-5}$ % d'erreur que nous espérons pour classer 90 % des patients sains comme sains. Nous pouvons espérer qu'avec plus de données nous pourrions faire des apprentissages plus longs et obtenir de meilleures performances en tests, l'augmentation de données des cellules saines paraît compliqué au vu du déséquilibre déjà important entre les cellules saines et infectées. Cependant une augmentation de données couplé à une pondération pourrait être une piste, comme le dopage des erreurs évoquer dans une partie précédente.

Référence

[1] https://www.who.int/malaria/media/world-malaria-report-2017/fr/ (retour texte)

[2] Autocount :
Autocount est un programme en libre accès issus de l'article si dessous. Son but est de détecter et classer les cellules de manière automatique.
[Ma et al. Malaria Journal 2010, 9:348]
https://malariajournal.biomedcentral.com/articles/10.1186/1475-2875-9-34 (retour texte)

[3] Bassin versant (watershed) :
C'est une méthode de morphologie qui à partir d'un marqueur va remplir les niveaux de gris d'une image en fonction de la connexité (si le point touche la zone marquée) et de la proximité du niveau (intensité proche).
https://scikit-image.org/docs/dev/api/skimage.morphology.html?highlight=watershed#skimage.morphology.watershed (retour texte)

[4] Transformée circulaire de Hough.
La transformée de Hough à l'origine sert à repérer des alignements de point dans une image. Par la suite cette transformation a été adaptée pour les cercles. On parle alors de transformée circulaire de Hough. Le principe reste le même détecter des cercles dans une image. Pour ce faire on calcule un grand nombre de cercle et on fait voter les points de contours pour le cercle sur lequel ils sont. Les cercles ayant les plus de vote sont conservé.
https://scikit-image.org/docs/dev/api/skimage.transform.html?highlight=hough_circle#skimage.transform.hough_circl (retour texte)

[5] Canny.
La méthode de Canny développée en 1986 a pour but de détecter les contours et de les affiner. Dans la fonction utilisée en python la méthode de Canny est couplé à un filtrage par hystérésis. Cette fonction permet de fixer un seuil haut et bas sur la détection de contour et ensuite de prolonger les points du seuil haut, s'il y a continuité avec un contour au-dessus du seuil bas.
https://scikit-image.org/docs/dev/auto_examples/edges/plot_canny.html (retour texte)

[6] Otsu.
Le seuillage par Otsu consiste à analyser l'histogramme d'une image et à trouver le seuil séparant les deux populations de l'histogramme.
Liens (retour texte)

[7] ImageNet.
ImageNet est une base de données composé d'image labellisé, réalisé en 2009 par l'Université de Princeton. En 2010 elle devient la base de données utilisé pour la compétition ILSVRC sur laquelle sont testés les différents algorithmes participant.
Réf : http://www.image-net.org

[8] Mask R-CNN :
https://arxiv.org/abs/1703.06870
https://www.pyimagesearch.com/2019/06/10/keras-mask-r-cnn/
https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272
https://steemit.com/fr/@rerere/detecter-des-objets-avec-des-reseaux-de-neurones-1
https://arxiv.org/pdf/1311.2524.pdf

[9] U-net. [17]
Un réseau de neurone de type U-net est un auto-encodeur. C'est-à-dire que nous avons une image en entrée et une image en sortie. Il est utilisé à la base pour labelliser les différentes zone d'une image. Dans ce contexte il est utilisé pour réaliser un traitement sur l'image de sortie.
https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (retour texte)

[10] ResNet 50
https://towardsdatascience.com/detecting-malaria-using-deep-learning-fd4fdcee1f5a (retour texte)

[11] DenseNet
https://towardsdatascience.com/densenet-2810936aeebb (retour texte)

[12] MP-IDB: The Malaria Parasite Image Database for Image Processing and Analysis
Loddo, Andrea & Di Ruberto, Cecilia & Kocher, Michel & Prod'Hom, Guy. (2019). MP-IDB: The Malaria Parasite Image Database for Image Processing and Analysis. 10.1007/978-3-030-13835-6_7.
https://www.researchgate.net/publication/331570908_MP-IDB_The_Malaria_Parasite_Image_Database_for_Image_Processing_and_Analysis (retour texte)

[13] Autres
https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624
https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

Bibliographie

[14] Wider or Deeper : Revisinting the ResNet Model for Visual Recognition.
Doi:10.1016/j.patcog.2019.01.006 *Pattern Recognition 90 (1019) 119-133*
Zifeng Wu, Chunhua Shen, Anton van den Henge (retour texte)

[15] Deep cnn frameworks comparaison for malaria diagnosis.
http://imvip.ie/ IMVIP 2019 Irish machine vision and image processing.
Priyadarshini Adyasha Pattanaik, Zelong Wang and Patrick Horain.
(retour texte)

[16] How useful is PCR in the diagnosis of malaria?
Doi:10.1016/S1471-4922(02)02348-6, Opinion| Volume 18, ISSUE 9, P395-398, September 01, 2002
Thomas Hänscheid, Martin P.Grobush, doi: 10.1128/JCM.44.3.1087-1089.2006
(retour texte)

[17] U-Net: Convolutional Networks for BiomedicalImage Segmentation

arxiv:1505.04597 Computer Vision and Pattern Recognition
Olaf Ronneberger, Philipp Fischer, Thomas Brox
(retour texte)

[18] TS-LSTM and temporal-inception : Exploiting spatiotemporal dynamics for activity recognition.
doi:10.1016/j.image.2018.09.003 *Signal Processing : Image Communication 71 (2019) 76-87*
Chih-Yoa Ma, Min-Hung Chen, Zsolt Kira, Ghassan AlRegib.

[19] Deep Residual Learning for Image Recognition
arxiv: 1512.03385 Computer Vision and Pattern Recognition
Kaiming He, Xiangyu Zangn Shaoqing Ren, Jian Sun

[20] Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images.
https://peerj.com/articles/4568.pdf
Sivaramakrishnan Rajaraman, Sameer K. Antani, Mahdieh Poostchi1,Kamolrat Silamut, Md. A. Hossain, Richard J. Maude, Stefan Jaeger et George R. Thoma.

[21] Malaria Cell Image Recognition
Chuping Wang, Buyun Gao, Fanfei Xu, Wenquan Chen

[22] Loddo, A.; Di Ruberto, C.; Kocher, M. Recent Advances of Malaria Parasites Detection Systems Based on Mathematical Morphology. *Sensors* **2018**, *18*, 513.

[23] Sornapudi, S.; Meng, F.; Yi, S. Region-Based Automated Localization of Colonoscopy and Wireless Capsule Endoscopy Polyps. *Appl. Sci.* **2019**, *9*, 2404.
doi:10.3390/app9122404

[24] Identity Mappings in Deep Residual Networks
arxiv:1603.05027 Computer Vision and Pattern Recognition

[25] Fully-automated patient-level malaria assessment on field-prepared thin blood film microscopy images, including Supplementary information.
Arxiv:1908.01901
Charles B. Delahumt and al.