Université Paris Descartes
Mathématiques et Modélisation parcours Traitement d'images

Rapport de stage

Analyse de vidéos de coloscopie et apprentissage

Stagiaire:
chenyu ZHA

Encadrant:
John CHAUSSARD

March 6, 2017

Remerciement

Je tiens à remercier monsieur John CHAUSSARD de m'avoir offert la chance d'effectuer mon stage de fin d'étude au sein du LAGA.Je tiens tout particulièrement à lui remercier de m'avoir apporté les explications nécessite à la compréhension du phénomène modélisé, et de m'avoir guidé dans l'aspect mathématique du stage et l'élaboration du code.

Je ne saurais terminer sans adresser un grand Merci à tous nos enseignants, à la direction de notre formation.

Table des matières

1 Introduction **3**
1.1 Contexte du stage 3
1.2 Objectif du stage 3
1.3 Environnement techniques 4

2 Étude de l'état de l'art **5**
2.1 Morphologie mathématique 5
2.1.1 Bases morphologiques 5
2.1.2 Outils morphologiques avancés 7
2.2 Réseaux de neurones 13
2.2.1 Introduction 13
2.2.2 Réseaux de neurones 15
2.2.3 Réseaux de neurones convolutif 21
2.2.4 Comparaisons entre ces deux modèles 24

3 Application **26**
3.1 Détection de texte 26
3.1.1 Algorithme statistiques 26
3.1.2 Algorithme morphologiques 29
3.1.3 Algorithme apprentissage 30
3.2 Estimation la position de coloscope 37
3.3 Détection de spéculaires 41
3.3.1 Algorithme morphologiques 41
3.3.2 Algorithme de calcul du flot optique 45

4 Conclusion **47**

1 Introduction

1.1 Contexte du stage

J'ai effectué mon stage au sein d'un Laboratoire Analyse, Géométrie et Applications(LAGA) en l'équipe **Mathématiques pour le traitement de l'information et de l'image**. Le LAGA est un laboratoire associé au CNRS. Il est rattaché à l'institut Galilée, composante de l'Université Paris 13, et à l'Université Paris 8. Le stage s'appuie sur analyser des vidéos de coloscopies de la maladie RCH(rectocolite hémorragique), elle fait partie des Maladies Inflammatoires Chroniques de l'intestin (MICI). Elle se caractérise par des lésions continues le plus souvent superficielles qui débutent dans le rectum et peuvent s'étendre sur l'ensemble du colon sans jamais atteindre d'autres segments du tube digestif. Les *scores endoscopiques* sont donc développés permettant de définir l'activité de la maladie, suivre l'évolution sous traitement d'un patient. Cependant, à cause de subjectivité, ces scores varient selon l'avis personnelle de médecins. Afin d'avoir une thérapeutique plus correcte, l'application de traitement des images de coloscopies avec les méthodes mathématiques et statistiques, qui est l'objectif de caractériser les symptômes(lésions, ulcères etc..) afin de prédire l'évaluation de la maladie, devient nécessaire et robuste.

1.2 Objectif du stage

Selon le contexte de stage, la première étape dont mon objectif du stage consiste à pré-traiter ces images de coloscopies pour avoir les résultats préliminaires avant de passer l'étape suivante. Concrètement, ces images contiennent des textes et spéculaires qui ont les intensités lumineuses assez élevés. Ces caractéristiques peuvent rendre les mauvaises influences pour les traitements de symptômes à la suivante, par exemple les textes et spéculaires sont considérés comme des anomalies. Pour débarrasser ces facteurs, ainsi on s'intéresse à appliquer les méthodes morphologiques et l'apprentissage automatiques afin de les détecter. Par ailleurs, lorsque d'une coloscopie, les observations des ulcères ou lésions se déroulent plutôt dans l'endroit de *muqueuse* que celui de *fond* de colon. En plus une déforme des images sont introduit dans ces deux situations. Basé sur cet initiative on voudrait trouver un moyen pour distinguer ces deux cas, autrement dit, l'estimation la position de coloscope, à l'aide de l'outil du traitement d'images. En résumé, l'objectif du stage est suivant:

- Détection de texte

- Estimation la position de coloscope

- Détection de spéculaires

1.3 Environnement techniques

Le matériel informatique utilisé est un portable ordinateur MacBook pro avec 8 Go mémoire, la vitesse processus atteint 2.7 GHz, ceci satisfait la plupart de tâches informatique durant le stage. Aussi, en connectant avec la machine *MAGI* qui est une *superMachine* du calcul via ssh, le temps du calcul de certaines tâches devient robuste. Au point de programmation, le langage *python* est utilisé tout au long du stage incluant les libraires *openCV, Numpy, Tensorflow....* Parmi lesquelles la librairie Tensorflow, qui est une open source de Machine Learning issu de Google, il s'agit en réalité d'un ensemble de fonctions mathématiques et de méthodes de calcul qui permettent de développer des fonctions d'apprentissage par la machine à adapter à chaque cas de figure.

2 Étude de l'état de l'art

2.1 Morphologie mathématique

La morphologie mathématique est une théorie de traitement non linéaire de l'information apparue en France dans les années 60 inventée par Georges Matheron et Jean Serra dans les laboratoire de MINES ParisTech. Dans un premier temps, il répondait principalement à des problème dans le domaine de l'exploitation minière. Aujourd'hui, il est largement utilisée en analyse d'images, dont le but est l'étude des objets en fonction de leurs forme, de leur taille,de leurs texture ainsi de suite. Par les transformations qu'elle propose, elle se situe à différent niveaux de traitement d'images comme filtrage, segmentation, quantification et modélisation.

2.1.1 Bases morphologiques

Les opérations morphologiques sont des filtres non-linéaire qui peuvent s'appliquer tant aux images niveaux de gris qu'à celles couleurs. Nous allons décrire quatre opérations morphologiques de bases, à savoir la *érosion, dilatation, fermeture, ouverture*. Ces quatre opérations sont largement utilisées pour filtrer des images dans vidéos au préalable.

Élément structurant

L'idée de base de la morphologie mathématique est de comparer l'ensemble à analyser avec un ensemble de géométrie connue appelé élément structurant. Un élément structurant est une image binaire où les pixels appartenant à l'objet noté 1, et les pixels du complémentaires seront noté 0. Un élément structurant peut avoir des formes différentes selon le choix(carré,cercle,diamond..). En 2d, il existe deux élément structurant importants: Γ_4 et Γ_8, qui associe respectivement à un point ses 4 voisins et 8 voisins.

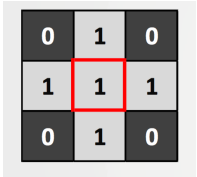


Figure 1: $\Gamma_4 = (-1,0),(1,0),(0,0),(0,1),(0,-1)$

Définition 2.2.1 l'application d'un élément structurant à un point d'espace *Soit E est un élément structurant, $E \subset Z^n$, est soit $x \in Z^n$, l'application de E sur x est $E_x = \{v + x \mid v \in E\}$*
Cette application est équivalent à une translation de E par x. Étant donnée un élément structurant et une relation, l'image de départ est transformé en translatant l'élément structurant en tout point en examinant si la relation entre l'objet et l'élément structurant translaté est vérifié. Ainsi, quarte opérations de base de la morphologie mathématiques (érosion, dilatation, ouverture, fermeture) seront introduites.
Définition 2.2.2 Érosion binaire *Soit I une image n-dimensionnelle de A à B, I:$A \subset Z^n \to B$, et $E \subset Z^n$ un élément structurant.On définit l'érosion de I par E l'application notée $I \ominus E$ telle que,*

$$\forall x \in A, (I \ominus E)(x) = Min_{y \in E_x} I(y)$$

L'érosion d'une image consiste à calculer dans l'environnement de l'élément structurant 'l'inf' de valeurs(la valeur minimum). L'effet de l'érosion est d'abord de rétrécir les objets en hauteur et largeur, en particulier si l'élément structurant est plus large ou plus haut que la figure, l'érosion de celle-ci sera vide.Cette opération fait disparaitre les petits objets et la figure seront déformés en fonction de l'élément structurant.

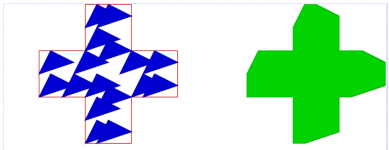


Figure 2: exemple d'une érosion[1]

Définition 2.2.3 Dilatation binaire*Soit I une image n-dimensionnelle de A à B, I:$A \subset Z^n \to B$, et $E \subset Z^n$ un élément structurant.On définit la dilatation de I par E l'application notée $I \oplus E$ telle que,*

$$\forall x \in A, (I \oplus E)(x) = Max_{y \in \breve{E}_x}(y)$$

où $\breve{E}$ est de rotation de 180 degrés de E.
La dilatation d'une image consiste à calculer dans l'environnement de l'élément structurant 'le sup' de valeurs(la valeur maximum).L'effet de la dilatation

[1]https://dpt-info.u-strasbg.fr/ cronse/TIDOC/MM/deof.html

est d'abord d'élargir la figure en hauteur et largeur. Cette opération fait disparaitre les petits trous et fait grossir les objets.

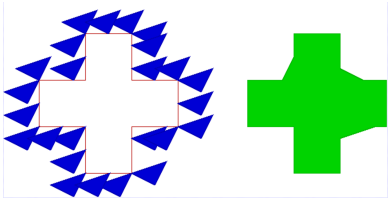


Figure 3: exemple d'une dilatation[2]

Définition 2.2.4 Fermeture *Soit I:A* $\subset Z^n \rightarrow B$*,et* $E \subset Z^n$ *un élément structurant.On définit la fermeture de I par E l'application notée* $I \bullet E$ *telle que,*

$$I \bullet E = (I \oplus E) \ominus E$$

La fermeture d'une image consiste à faire une dilatation suivie une érosion.Cette opération sera éliminée les zones sombres moins larges que l'élément structurant.Dans un premier temps, la dilatation rend l'image plus brillante tandis que l'érosion redonne le niveau de brillance moyen.
Définition 2.2.5 Ouverture *Soit I:A* $\subset Z^n \rightarrow B$*,et* $E \subset Z^n$ *un élément structurant.On définit l'ouverture de I par E l'application notée* $I \circ E$ *telle que,*

$$I \circ E = (I \ominus E) \oplus E$$

L'ouverture d'une image consiste à faire une érosion suivie une dilatation. Cette opération sera éliminée des pics brillants moins larges que l'élément structurant. Dans un premier temps, l'érosion rend l'image plus foncée tandis que la dilatation redonne le niveau de brillance moyen.

2.1.2 Outils morphologiques avancés

De nombreux outils morphologiques peuvent être construits par l'assemblage d'opérations de bases et appliqués sur des images.Dans cette section,quelques algorithmes issus de base morphologiques seront introduits.Aussi, ces méthodes sont utilisées durant le stage permettant d'analyser, filtrer où segmenter des images selon le cas.

[2]https://dpt-info.u-strasbg.fr/ cronse/TIDOC/MM/deof.html

Définition 2.3.1 Gradient morphologique *Soit I :A* $\subset Z^n \rightarrow B$, *et E* $\subset Z^n$ *un élément structurant. On définit le gradient morphologique de I par E est notée:*

$$\bigtriangledown_E(I) = (I \oplus E) - (I \ominus E)$$

Appelons I l'image originale dont on veux extraire les contours. Soit E un élément structurant, alors la différence $(I \oplus E) - I$ produit les contours extérieurs d'images tandis que $I - (I \ominus E)$ produit les contours intérieures. Par combinaison ces deux opérateurs tous destinés à extraire les contours de l'image.Le gradient morphologique se définit comme la somme de contours extérieurs et intérieurs, qui est équivalent de dire la différence entre *dilatation* et *érosion* de I par E.

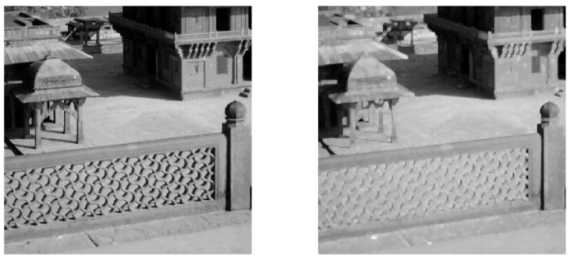


Figure 4: gauche: Image originale. droite:Image dilatée[3]

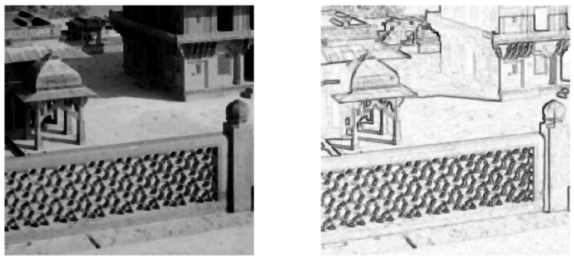


Figure 5: gauche: Image érodée. droite:Image gradient(inverse)[4]

Définition 2.3.2 Ouverture par reconstruction *Soit I :A* $\subset Z^n \rightarrow B$, *et E,F* $\subset Z^n$ *un élément structurant. On définit l'ouverture par reconstruc-*

[3]http://www.telecom.ulg.ac.be
[4]http://www.telecom.ulg.ac.be

tion(sous F) de I par E est notée:

$$I \circ_F E = I \triangle_F (I \circ E)$$

L'opération ouverture par reconstruction consiste à réaliser une ouverture, puis reconstruction de l'image à partir de son ouvert. Tandis que l'ouverture classique modifie les contours, cette transformation permet de reconstruire efficacement et avec précision de contours des objets. Dans un premier temps, l'ouverture de I par E supprime les composantes connexes de tailles inférieures à E, puis la reconstruction par l'ensemble marquer M $= I \circ E$ reconstruit les autres composantes connexes. En général, le choix d'élément structurant F qui sera utilisé pour la reconstruction est Γ_4, Γ_8...

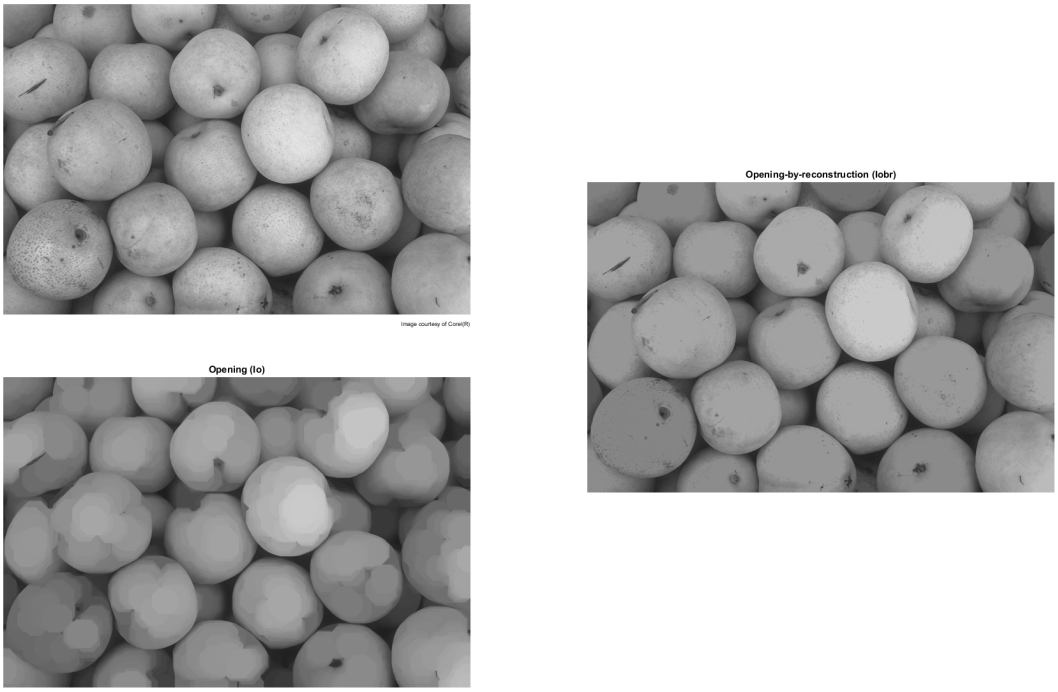


Figure 6: en haut à gauche: Image originale. en bas à gauche: résultat après une ouverture. droite: résultat après une ouverture par reconstruction.[6]

[6]https://fr.mathworks.com/help/images/examples/marker-controlled-watershed-segmentation.html

TopHat
La transformation TopHat consiste à la détection du *pic ou ravin* dans l'image en fonction d'un élément structurant choisi. Il existe deux type de Tophat : *White Top Hat et Black TopHat.* Le premier est appliqué en calculant la différence entre l'image originale et son ouverture tandis que le dernier calcule celle entre l'image originale et sa fermeture.L'effet de TopHat sur une image est d'extraire les pixels ayant les intensités lumineuses élevées(*White TopHat*) ou ceux faibles(*Black TopHat*).

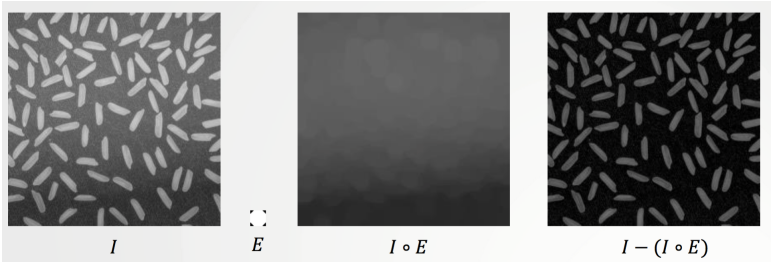


Figure 7: extraire des grains du riz en utilisant *White TopHat*[7]

La méthode *White TopHat* voir est utilisée pendant les travaux sur la détection de texte. L'intérêt d'implémenter cette méthode est d'extraire les parties textes dans l'images afin de créer une masque. Cette masque s'utilise ensuite à travers l'ensemble de l'images pour générer les bases de données qui serviront à réseaux de neurones.
H-maxima
La transformation H-maxima est définie en faisant une *reconstruction inférieure* avec le marqueur $M = I\text{-}h$, où h est la hauteur de l'image en considérant que celle est comme un relief topographique. L'effet de cette opération permet de supprimer les *pic* de l'image dont la hauteur est inférieur à certaines h choisi. En particulier si h égal 1, le résidu de la transformation H-maxima rend l'image *maximum régionaux.* Un maximum régional est une zone plate qui n'est pas adjacente à une zone plate de plus haute altitude.

Définition 2.3.2 Reconstruction inférieure *Soit I une image n-dimensionnelle de A à B, M un marqueur contenu dans I. I, M:A*$\subset Z^n \rightarrow B$*, E*$\subset Z^n$*. La reconstruction inférieure de M par E restreinte à I est notée*

$$I \triangle_E M = (M \oplus_I E)^\infty$$

[7]ppt de M.CHAUSSARD

où $M \oplus_I E = M \oplus E \wedge I$ décrit la *dilatation conditionnelle* de M par E restreinte à I. Il s'agit d'itérer la *dilatation conditionnelle* jusqu'à idempotente.Cette opération permet la reconstruction de toutes les composantes connexes de I marquées par M.

Ligne de partage des eaux

La ligne de partage des eaux constitue l'approche morphologique de la segmentation d'une image en niveaux de gris. Le but est d'obtenir une partition d'images en régions regroupant des pixels jugés équivalents en fonction d'un certain critère. Considérer une image comme une surface topographique, où l'altitude correspond au niveaux de gris, le principe de construction de la LPE par inondation est le suivant: Imaginons tous les minima régionaux sont percés,on commence à injecter de l'eau.A chaque fois que la hauteur de l'eau atteint l'altitude d'un minimum régional, un nouveau bassin versant est crée.Si deux bassins se rencontrent, on empêche leur fusion en construisant un barrage.Ces barrages constituent ce qu'on appelle la *ligne de partage des eaux*. Le résultat est une partition de l'image en bassins versants et en ligne de partage des eaux(barrages).

Dans le cas d'images naturelles, l'application de l'algorithme produit une sur-segmentation puisque chaque minima induit un bassin versant. Pour résoudre ce problème, on utilisera des *marqueurs*, qui ont pour rôle d'imposer la présence de certains bassins versants. Concrètement, ces *marqueurs* contiennent des pixels appartenant à l'objet(*foreground*) et son complémentaire (*background*). Le choix d'un marqueur peut être un maxima(minima) régional, qui est obtenu par *H-maxima*. Un grand avantage d'utiliser un maxima(minima) régional comme marquer est son indépendance de valeurs seuils.

Lorsqu'on applique *lpe contrôlée par marqueurs*, supposons ces marqueurs sont placés dans les *minima régionaux*. Chaque marqueur est associé à une couleur. Ensuite, l'eau va inonder les bassins versants ceux qui n'ont pas de marqueurs. Si deux couleurs de l'eau commence à fusionner, on les empêche en plaçant un barrage. Les bassins versants crées avec différentes couleurs correspondent celles de marqueurs associés.

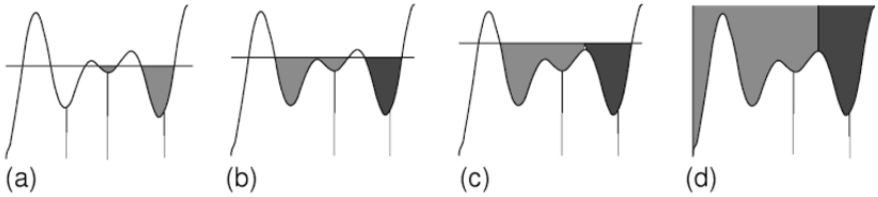


Figure 8: simulation 1D *lpe contrôlée par marqueurs*. (a)état initial. (b)l'eau est inondé les bassins versants ceux qui n'ont pas de marqueurs. (c)un barrage est crée lorsque deux couleurs de l'eau commence à fusionner. (d)résultat final,un seul barrage(*ligne de partage des eaux*).[9]

[9]Microscope Image Processing

2.2 Réseaux de neurones

2.2.1 Introduction

Contexte biologique Un neurones est une cellule nerveuse constituant la base du système nerveux spécialisée dans le traitement des signaux électriques, il se compose d'un *corps cellulaire*, un *axone*, des *dendrites*, des *synapses*. Un corps cellulaire est le centre de contrôle d'un neurone, qui fait la somme des informations qui lui parviennent. Il traite ensuite l'information et renvoie le résultat à l'entrée des autres neurones au travers de son *axone*, ceci joue un rôle important dans le comportement logique de l'ensemble. Quant aux dendrites, sont les récepteurs principaux du neurones, elles véhiculent le signal nerveaux À la fin, les synapses reçoivent les informations des autres neurones via l'axone et permettent donc aux neurones de communiquer entre eux.

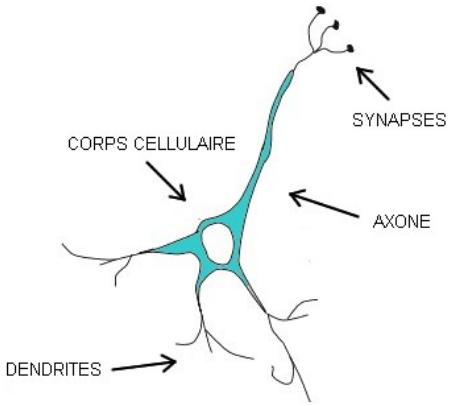
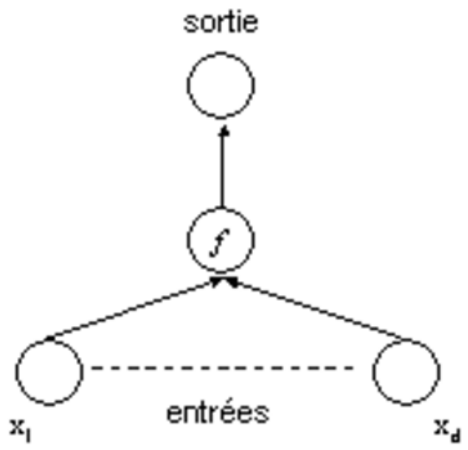


Figure 9: Schéma d'un neurone biologique[10]

Du neurone biologique au neurone artificiel Les réseaux de neurones biologiques qui constituent le cerveau humain réalisent simplement de nombreuse applications telles que la reconnaissance de formes, le traitement du signal, la mémorisation à l'aide un grand nombre de neurones qui unissent leurs efforts. Inspirée du fonctionnement des neurones biologiques, les scientistes ont crée un *réseaux de neurones artificiels* donc un moyen de modélisation le mécanisme d'apprentissage et de traitement de l'information qui se produit dans le cerveau humain.

[10] https://fr.wikiversity.org/



Modèle à un Seul Neurone

Concrètement, considérons ici la forme la plus simple des *réseaux de neurones artificiels* constitué d'un seul neurone avec certains nombre d'entrées et une seule sortie. Chaque entrée est associée une valeur *poids*, celle-ci décrit l'intensité de chaque entrée, donc plus un poids w est important, plus l'entrée associée est influente. Ensuite une somme pondérée des entrées est calculée pour constitution la *fonciton d'activation* f, le rôle de cette fonction est de produire la *sortie* du neurone. Par conséquent:

$$sortie = f(w_1x_1 + w_2x_2...w_nx_n)$$

La sortie d'un neurone est en effet une prévision du modèle à un seul neurone, on suppose qu'il existe une relation entre les entrées et la sortie, à l'aide de réseau de neurone, qui modélise cette relation en reliant les entrées aux cibles par une fonction mathématique adaptée. En gros, un réseau de neurone peut extraire des règles et des tendances à partir d'une basse de données en utilisant des modèle mathématique. L'une des propriétés intéressantes des réseaux de neurones est qu'ils savent prévoir avec précision des données qui ne faisaient pas partie des données d'apprentissage, un processus connu sous le nom de généralisation, il est donc bien adapté au certain domaines :

- Traitement du signal
- Classification
- Analyse de l'image et synthèse vocale

- Reconnaissance de formes
- ...

A la suite de rapport, deux modèles de réseaux de neurones seront introduits par leurs architectures, leurs fonctions mathématiques, leurs usages issus des problèmes classiques. L'objectif est d'avoir une revue des modèles et de les comparer selon des différents caractéristiques.

2.2.2 Réseaux de neurones

versions la plus simple Pour comprendre la logique de l'ensemble de réseaux de neurones, prenons la version la plus simple dont celle ne contient pas de *couche cachée*. Considérons un seul neurone avec plusieurs entrées x_1, x_2.. et ses poids w_1, w_2... correspondants. On calcule la somme pondérée des entrées puis applique à cette valeur une *fonction d'activation*. Le résultat obtenu est ensuite comparé avec un *seuil* choisi.

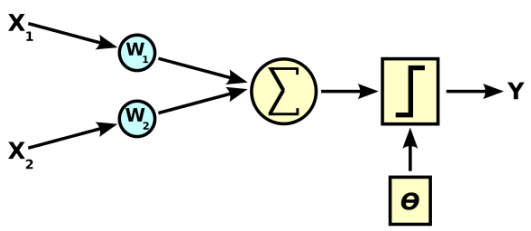


Figure 10: Neurone formel avec 2 entrées et une fonction d'activation à seuil[12]

Perceptron
La méthode perceptron est définie suivante:

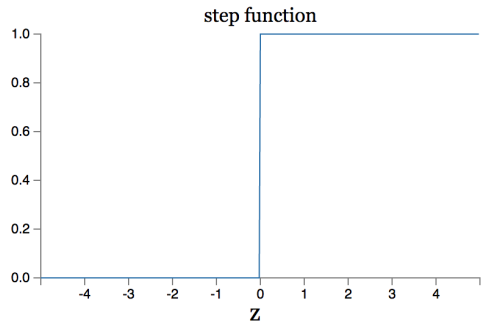
$$\text{sortie} \quad = \quad \begin{cases} 0 & \text{si } \sum_j w_j x_j \leq \text{ seuil} \\ 1 & \text{si } \sum_j w_j x_j > \text{ seuil} \end{cases} \tag{1}$$

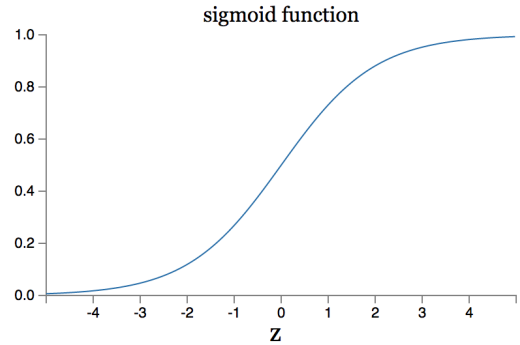
Pour simplifier la formule précédente, un *bias* b est introduit pour remplacer la valeur *seuil* tandis que $w \cdot x \equiv \sum_j w_j x_j$. Ainsi, la méthode preceptron est définie:

$$\text{sortie} \quad = \quad \begin{cases} 0 & \text{si } w \cdot x + \text{ b} \leq 0 \\ 1 & \text{si } w \cdot x + \text{ b} > 0 \end{cases} \tag{2}$$

[12]https://fr.wikipedia.org/wiki/Neuroneformel

Le modèle perceptron est un modèle très basique en choisissant la *fonction d'activation* comme *step function*. Dans la pratique, cette méthode ne produit pas les résultats précises et convenables avec des entrées plus complexes. Imagions que le réseau avait mal classifié une image comme '8' au lieu de '9'. L'intérêt est de modifier un petit peu le *bias* et *poids* au fur et à mesure en espérant que la sortie soit aussi modifiée un petit peu afin d'approcher de temps en temps au bon résultat. Durant ce processus, le réseau s'entraine. Or, le modèle perceptron risque à sauter complètement(de 0 à 1 par ex.) lorsque l'on modifie un peu le *bias* et *poids*. Ce changement imprévu peut influer les autres parties de réseau et les résultats deviennent difficiles à contrôler et évaluer. Ainsi, pour résoudre ce problème, un nouveau type de neurone sera introduit appelé *neurone sigmoïde* et de la même manière, la *function d'activation* sera remplacé de *step function* : *la fonction sigmoïde*.





Neurone Sigmoïde Le principe de neurone sigmoïde reste le même que ceci de perceptron: les entrées $x_1, x_2...$ et ses poids $w_1, w_2...$ et un terme *bias*, b. En revanche, la sortie ne sera plus 0 ou 1 mais une valeur qui est calculée par $\sigma(w \cdot x + b)$, où σ est la *fonction d'activation* appelée *fonction sigmoïde*, dont la formule explicite est suivante:

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}. \tag{3}$$

En remplaçant z par $\sum_j w_j x_j + b$, la sortie s'écrit:

$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)}. \tag{4}$$

Ainsi, la sortie est une valeur dans l'intervalle $[0, 1]$, ce qui permet d'interpréter la sortie du neurone comme une probabilité. Le lien entre la *fonction*

sigmoïde et *step fonction* est lorsque $z \equiv w \cdot x + b$ égal un nombre très grand ou très négative, $\sigma(z) \approx 1$ ou 0 respectivement. Ces deux extrémités correspondent exactement au comportement de *step fonction*. En outre, comme la fonction σ est une fonction continue et dérivable, cela signifie qu'une petite modification dans Δw_j et Δb produira aussi une petite modification $\Delta sortie$ d'après les calculs:

$$\Delta \text{sortie} \approx \sum_j \frac{\partial \, \text{sortie}}{\partial w_j} \Delta w_j + \frac{\partial \, \text{sortie}}{\partial b} \Delta b \tag{5}$$

Ainsi, grâce à la linéarité de cette fonction, on peut observer directement le changement de la sortie en fonction de Δw_j et Δb.
En pratique, il existe nombreux de *fonction d'activation* telles que *tangente hyperbolique*, *unité de Rectification Linéaire(ReLu)*, *Softmax* ainsi de suite. Le choix de la fonction d'activation influent en effet sur celle du réseau et il est donc important de bien choisir celle-ci pour obtenir un modèle utile.
Modèle avec couches cachées Dans cette section, un *véritable* réseau de neurone sera introduit tout commenceront par son architecture et sa conception. Ensuite, les explications de méthodes mathématiques relèvent comment le réseau apprend pour prédire les résultats.

L'architecture du réseau de neurones

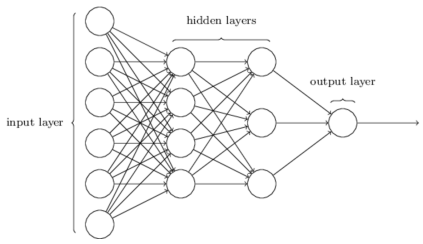


Figure 11: Un exemple de l'architecture du réseau de neurones[13]

Un réseau de neurones est organisé en plusieurs couches: une *couche d'entrée*, une *couche de sortie*, au milieu il s'agit de *couches cachées*, cela signifie 'ni couche d'entrée ni couche de sortie'. Un réseau peut avoir plusieurs couche cachées et le nombre de neurones sur cette couche et implicite et doit être ajusté. En général, on peut commencer par une taille moyenne des couches d'entrée et de sortie mais ce n'est pas toujours le meilleur choix. Il sera souvent préférable pour obtenir de bon résultats, d'essayer le plus de tailles

possibles. L'exemple au-dessus correspond un réseau de type *feedforward*, puisque l'information circule de la couche d'entrée vers la couche de sortie uniquement. Il existe aussi le réseau de neurones récurrent, où il existe au moins un cycle dans la structure. Dans la suite du rapport, on conçoit et implémente plutôt le modèle *feedforward*.

La conception d'un réseau
Pour la création d'un réseau de neurone, en gros, il nécessite de suivre ces quartes grandes étapes:

- **Choix et préparation des bases de données**
  Le processus d'élaboration d'un réseau de neurones commence toujours par le choix et la préparation des bases de données. Ces bases de données ensuite sont divisées par trois parties : *données d'apprentissage*, *données de validation*, *données de test*. Les données d'apprentissage utilisent pour s'entrainer notre réseau tandis que celles de validations permettent d'optimiser les *hyper-paramètres* et réduction le *overfitting*. Quant aux données de test, sont appliquées pour évaluer la performance du réseau. Par conséquent, c'est une étape cruciale pendant la conception d'un réseau.

- **Élaboration de la structure du réseau**
  La structure du réseau consiste à déterminer de la manière explicite le nombre de couches cachées et le nombre de neurones dans la couche cachée. Le choix est assez heuristique et il vaut mieux tester toutes les possibilités.

- **Apprentissage**
  L'apprentissage d'un réseau de neurones consiste à déterminer les *poids* et *bias* suivant le problème à résoudre. La méthode la plus utilisée est la *rétro-propagation du gradient* : une méthode qui permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première, les *poids* et le *bias* du réseau de neurones sont alors changés. C'est un cycle qui est répété jusqu'à ce que la courbe d'erreurs du réseau ne soit croissante (il faut bien prendre garde ne ne pas sur-entrainer un réseau de neurones qui deviendra alors moins performant). Les *poids* et *bias* ajustés d'après la phase d'apprentissage sont utilisé pour le réseau.

[13] http://neuralnetworksanddeeplearning.com/chap1.html

- **Validation et Tests**
  Alors que les tests concernent la vérification des performances d'un réseau de neurones et sa capacité de généralisation, la validation est parfois utilisée lors de l'apprentissage pour éviter le problème *ouverfitting*. Dans le cas général, une partie de bases de données est simplement écarté de données d'apprentissage et conservé pour les tests. On peut par exemple utiliser 60 % de l'échantillon pour l'apprentissage, 20 % pour la validation et 20 % pour les tests. Lors de la validation, on calcule un *score*(mesurer comment réseau pénalise l'écart entre le signal prévu et réel) qui est normalisé entre 0 et 1. Quand d'après avoir dépassé un certain nombre d'époque prédéfinie, ce score n'est pas toujours atteint son dernier meilleur score, on considère que le réseau est atteint *overfitting*. Dans ce cas là, si l'on continue à entrainer le réseau, il perdra sa régularisation. Du coup on impose au réseau d'arrêter son entrainement et cette technique s'appelle *Early stopping*.

Comment un réseau s'entraine

Un réseau de neurones s'entraine lors de son apprentissage. Prenons un exemple classique concernant la classification des chiffres écrit à la main. Autrement dit, la classification de base *MNIST*(Mixed National Institute of Standards and Technology). La base MNIST regroupe 60000 images d'apprentissage et 10000 images de test, ce sont des images en noir et blanc avec la taille 28 x 28.

Figure 12: MNIST handwritten digits

Chaque entrée x_i correspond un vecteur de taille 28 x 28 =784 dont 784 neurones associée la couche d'entrée tandis que la couche sortie est définie par un vecteur de taille 10 dont 10 neurones. Si l'entrée image x est un chiffre 4, la sortie ciblée est ainsi $y(x) = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0)^T$. On introduit aussi la sortie de couche correspondant l'entrée image la valeur a. Afin de quantifier les *erreurs* entre la sortie ciblée et la sortie de couche, une fonction de coût est définie de la manière suivante:

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2. \tag{6}$$

L'objectif est ainsi de trouver les valeur de w et b qui minimise C. Pour ce faire, on utilise la méthode *décente gradient*, il s'agit de modifier w et b itérativement en calculant la dérivé partielle de chaque composante dans w et b associé un pas de gradient η(dans le réseau de neurone on l'appelle aussi *learning rate*).

$$w_k \quad \rightarrow \quad w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{7}$$

$$b_l \quad \rightarrow \quad b'_l = b_l - \eta \frac{\partial C}{\partial b_l}. \tag{8}$$

En répété ce processus, on espère d'atteindre le minimum de C. En pratique, pour calculer le gradient ∇C, on besoins de calculer ∇C_x pour chaque input x. Ce processus est lent en cas d'avoir une base de données massive. Ainsi, on crée les *mini-batch* qui sont choisi aléatoirement parmi les entrées. Notons le nombre est m, on a la relation suivante:

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C, \tag{9}$$

Ainsi, les poids w et b sont redéfinis de la manière suivante:

$$w_k \quad \rightarrow \quad w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k} \tag{10}$$

$$b_l \quad \rightarrow \quad b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}, \tag{11}$$

Grâce à ce changement, l'algorithme devient plus robuste. Pour la classification de MNIST, avant la taille d'images d'entrées est 6000, en choisissant $m = 10$, cela signifie une accélération de 600 fois plus en théorie.
Ensuite, pour calculer le gradient de la fonction de coût, dont la dérivée partielle $\partial C/\partial w$ et $\partial C/\partial b$, la méthode *rétro-propagation* sera appliquée, qui décrit à la manière suivante:

$\diamondsuit$ Pour une entrée d'image x, calculer la valeur a^1 d'après la *fonction d'activation* σ: $a^1 \equiv \sigma(w^1 x + b^1)$

$\diamondsuit$ Pour l nombre de couches =2,3...L, calcule *feedforward* la valeur sortie de chaque couche :$a^l \equiv \sigma(w^l a^{l-1} + b^l)$

$\diamondsuit$ L'erreur de couche sortie L est calculée par: $\delta^L = \nabla_a C \odot \sigma'(w^L a^{L-1} + b^L)$, où $\odot$ désigne le produit matriciel de Hadamard. Le terme $\nabla_a C$ est un vecteur dont chaque composante est la dérivée partielle $\partial C / \partial {a_j}^L$ du neurone j dans la couche L.

$\diamondsuit$ On calcule l'erreur δ^l, $\forall l \in L-1, L-2...2$ depuis l'erreur précédente. Autrement-dit,on rétro-propage l'erreur: $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

$\diamondsuit$ Le gradient de *fonction de coût* est donné par: $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ et $\frac{\partial C}{\partial b^l_j} = \delta^l_j$, où w^l_{jk} signifie le poids du neurone k dans la couche $l-1$ à neurone j dans la couche l.

En combinant la méthode *rétro-propagation* et *décente gradient stochastique*(ceci avec *mini-batch*), pour chaque entrée d'image x, les poids w et bias b s'écrit:

$$w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T \tag{12}$$

$$b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}, \forall l \in L, L-1....2 \tag{13}$$

La condition d'arrêt de l'algorithme dépend de certaines conditions qui peuvent être atteindre le nombre maximum d'époques ou quand l'erreur ne diminue plus assez. Pour rendre un réseau qui s'entraine plus robuste, le choix de *fonction d'activation* et *fonction de coût* ou bien même le type de réseau concernent beaucoup. Dans la section suivante, un autre modèle de réseau sera introduit qui est largement utilisé aujourd'hui pour la classification l'images, la reconnaissance de formes etc.

2.2.3 Réseaux de neurones convolutif

Un réseaux de neurones convolutif est un type de réseau *feedforward*, qui se compose des algorithmes du *Deep Learning*, qui fait parties d'une famille de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données. Il s'agit d'extraire les caractéristiques d'images à travers multiples couches cachées.

L'architecture d'un réseau de neurones convolutifs

- **Couche de convolution**
  Prenons le même exemple d'une image x issue de base MNIST. Cette fois-ci, l'entrée est une matrice de neurones taille 28 x 28 (contredit celle d'un vecteur 784 neurones). Ensuite, on l'associe à la première couche cachée notée la *couche de convolution*. Explicitement, une image est passée à travers un *noyau de convolution*, créant de nouvelles images appelé les cartes de convolutions.

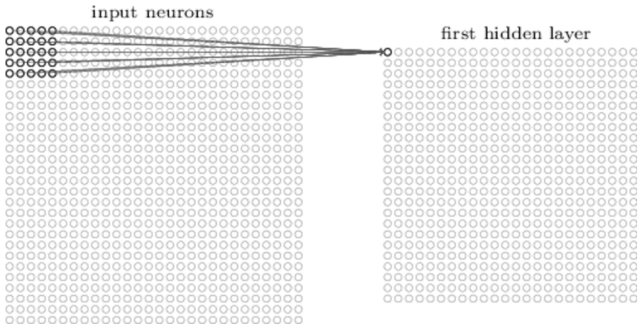


Figure 13: Un exemple du noyau de convolution avec taille 5x5, le neurone du couche cachée est obtenu par: $\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} x_{j+l,k+m}\right)$. On utilise les même poids w et bias b dans le *noyau de convolution* tout au long le chemin de parcours. La taille de la première couche cachée égale 24x24.

- **Couche de pooling**
  Les sorties de *couche de convolution* est ensuite considérées comme les entrées de la couche suivante: *La couche de pooling.* Dans cette couche, on choisi l'élément maximum de neurones dans l'image par une région 2 x 2, qui s'appelle aussi *max-pooling.* Ainsi, on obtient une image de taille 12 x 12 à la sortie. L'intérêt de l'introduire cette couche a pour le but de réduire la taille d'image intermédiaire et le nombre de paramètres dans le réseau.

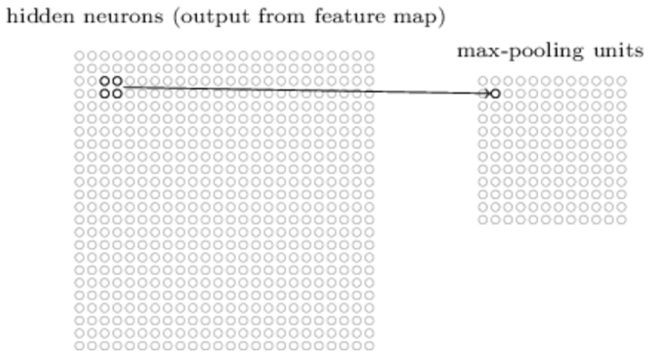


Figure 14: Schéma de *couche de pooling*: L'image sortie de la couche convolution avec taille 24 x 24 ainsi que celle de la sortie de la couche pooling est 12 x 12.

- **Couche entièrement connectées**
  Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via de la *couche entièrement connectée*. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente. Les sorties peuvent être calculées par $\sigma(wa+b)$, où a désigne les sorties redimensionnées à un vecteur d'après la couche de pooling.

- **Couche de sortie**
  Pour la couche de sortie, la *fonction softmax* est utilisée comme la *fonction d'activation* avec l'avantage que les sorties de neurones soient considérés comme une distribution de probabilité. Par conséquent, pour le problème de MNIST, les sorties seront dix valeurs entre [0, 1] qui somment jusqu'à un.

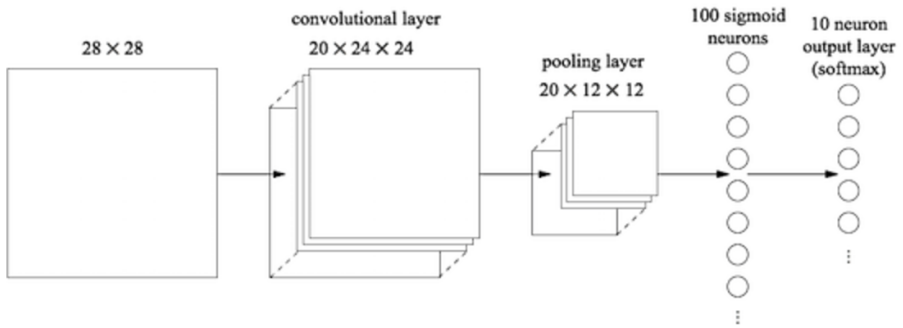


Figure 15: l'architecture d'un réseau de neurones convolutifs. Au lieu de prendre un seul *noyau de convolution* dans le cas précédent, on en a défini 20.

Concernant la partie d'apprentissage, le principe de ceci dans réseau de neurones convolutifs reste même qu'un autre modèle. L'intérêt d'ajouter la *couche de convolution* est pour que le réseau apprenne la structure spatiale tandis que la couche entièrement connectée permet réseau de s'entrainer les caractéristiques plus abstraites dans l'images.

2.2.4 Comparaisons entre ces deux modèles

On remarque un réseau de neurones convolutifs est basé sur un réseau de neurone(non convolutif) en raison de la *couche entièrement connectée*. Bien que les efficaces pour le traitement d'images, un réseau de neurone(non convolutif) a beaucoup de mal à gérer des images de grande taille, ce qui est dû à la croissance exponentielle du nombre de connexions avec la taille de l'image. Les avantages d'un réseau de neurones convolutif ont les suivants:

1. La couche de neurone n'est plus simplement une surface, mais devient un volume avec une profondeur(considère l'image entrée est une image de couleur). Cela préserve la structure spatiale d'image entrée.

2.En appliquant un noyau de convolution à l'image, un réseau de neurones convolutifs assure ainsi que ceci produise la réponse la plus forte à un motif d'entrée spatialement localisé, une telle représentation occupe moins d'espace en mémoire. De plus, le nombre de paramètres à estimer étant réduit.

3.Dans le réseau de neurones convolutifs, les poids w et bias b dans un *noyau de convolution* sont identiques pour tous les autres neurones dans l'images, ce qui induit une propriété d'invariance par translation.

En gros, ces propriétés permettent aux réseaux de neurones convolutifs d'obtenir une meilleure généralisation. Pour le problème de MNIST, souvent le réseau de neurones convolutif a un *score* plus élevé et la diminution de l'empreinte mémoire permet l'apprentissage de réseaux plus grands donc souvent plus puissants.

3 Application

Les applications des méthodes introduisant dans les sections précédents durant le stage consistent à pré-traiter et analyser les images issues des vidéos de coloscopie pour la maladie RCH. Parmi lesquelles les méthodes morphologiques sont utilisées à travers l'ensemble de travails permettant d'avoir des résultats préliminaires. Aussi, l'application de réseau de neurone joue un grand rôle pour la classification des images. En outre, grâce aux algorithmes (suivi d'objets, 2D gussien fitting), qui ainsi facilitent à atteindre des objectifs de certaines tâches.

3.1 Détection de texte

Dans certaines films endoscopiques, les textes apparaissent du coté gauche des vidéos ayant l'objectif d'enregistrer les informations nécessaires du diagnostique. Les textes sont assez brillant et donc ont des valeurs de luminosités élevées. La première tâche est d'implémenter des algorithmes morphologiques(ou statistiques) permettant d'extraire la partie textes. Ensuite, pour la généralisation, on a crée une base de données qui contiennent les images avec ou sans textes, qui sont utilisées pour la création d'un réseau de neurones afin de prédire le *score* en espérant que le réseau de neurones ait bien classifié les images.

3.1.1 Algorithme statistiques

Idées principales

Les textes apparaissant dans une vidéos ont des pixels avec positons plus au moins *stable*. Autrement dit, la variance de textes sont très faibles. Ainsi, on calcule l'image de variance en parcourant certaines frames, ensuite convertissons l'image à LAB, puis en analysant l'histogramme de canal L(luminosité), on remarque que l'histogramme est constituée trois gaussiennes : une de l'arrière plan(noir) dont la valeur de luminosité est 0, une de texte est une de reste dans la vidéo d'où c'est un *modèle de mélange gaussien*. Pour trouver un bon *seuil* afin d'extraire la partie texte, on estime ainsi les trois paramètre de chaque gaussiens : la *variance*, la *amplitude*, la *moyenne*, où le dernier correspond la valeur de seuil. Ces paramètres sont optimisés selon un critère de *maximum de vraisemblance* via l'algorithme espérance-maximisation (EM) pour approcher le plus possible la distribution recherchée.

Résultats numériques

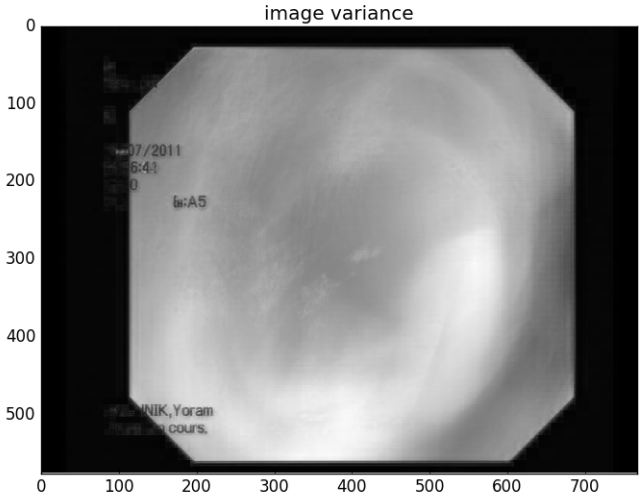


Figure 16: image de variance en passant 200 frames de la vidéo. Pour une bonne visualisation, on affiche l'image en choisissant *cmap = 'gray'*. En faisant comparaisons avec d'autres parties(hors l'arrière plan), la partie texte est sombre qui prouve qu'elle a une variance très faible.

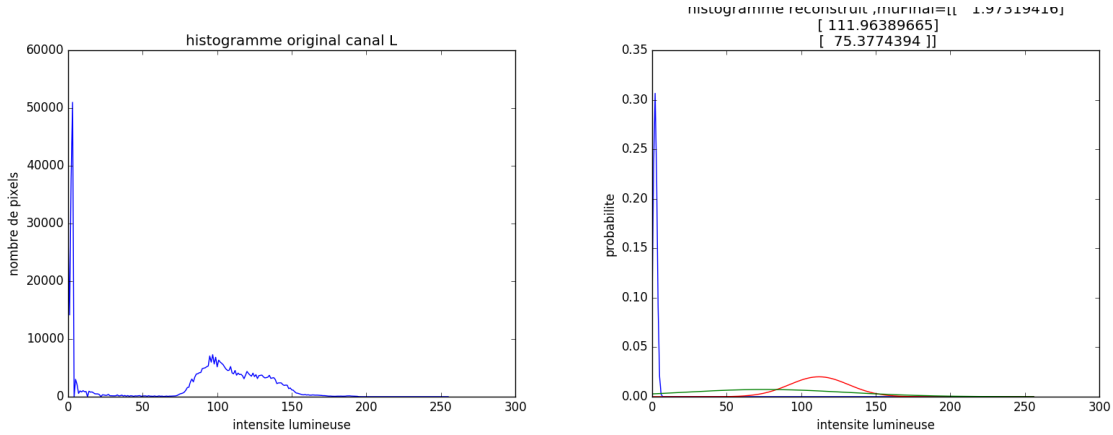


Figure 17: L'image a gauche représente histogramme en canal L et celle de droite est le résultat d'après l'algorithme EM.

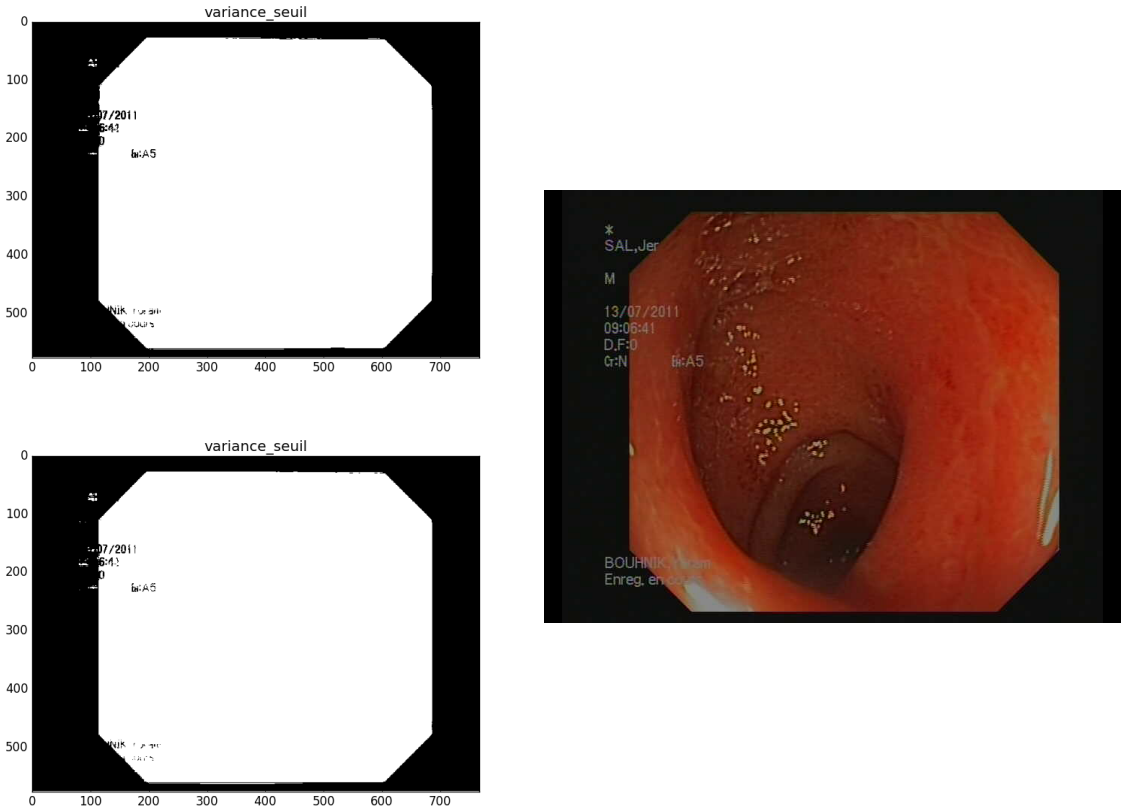


Figure 18: en haut a gauche: image seuillée en prenant seuil=100; en bas a gauche: image seuillée en prenant seuil = 90; droite: image au départ

Remarque et Conclusion

D'après la Figure.17, en visualisation, la reconstruction d'histogramme par EM est assez correcte malgré qu'elle a failli de repérer certaines max-locaux. On remarque aussi pour les deux gaussiennes(hors l'arrière plan) reconstruits, il existe une partie commune, ainsi le choix d'un seuil convenable devient assez compliqué.Par ailleurs, en regardant la Figure.18, l'image seuillée est malheureusement confondue dans la partie où les textes sont initialement superposés avec l'arrière plan. En général, utiliser l'algorithme statistiques pour la détection de texte, l'avantage est qu'il donne les résultats avec paramètres plus concrètes(μ,σ,π) afin d'analyser et évaluer, en revanche, à cause des erreurs de l'estimation et la sensibilité de choix de seuil, qui influence malheureusement l'image finale obtenue.

3.1.2 Algorithme morphologiques

Idée principale

Les textes apparaissant dans une vidéo ont des l'intensité lumineuse élevées. Considérons l'image est comme une surface topographique, les textes représentent donc les *pic* d'un relief. À l'aide de l'algorithme *topHat*, qui est en effet le résidu de l'opération *ouverture*, on peut récupérer que les *pic* dans l'image d'où les textes. Ensuite faire un seuillage pour avoir une image avec un mieux contraste.

Résultat numériques

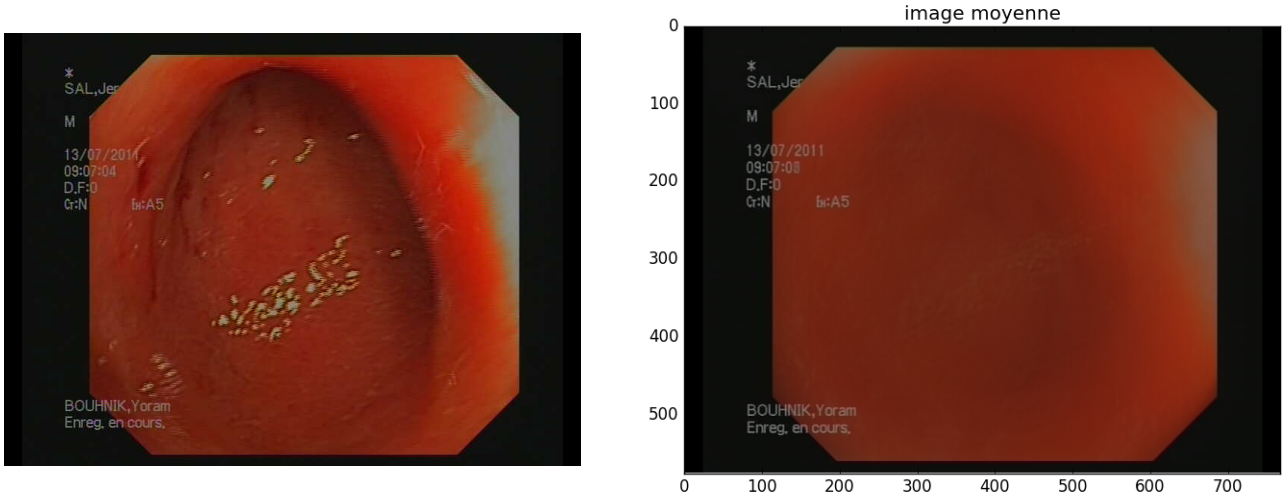


Figure 19: En choisissant l'image au départ égale frame 1400, on calcule image moyenne en passant 200 frames.

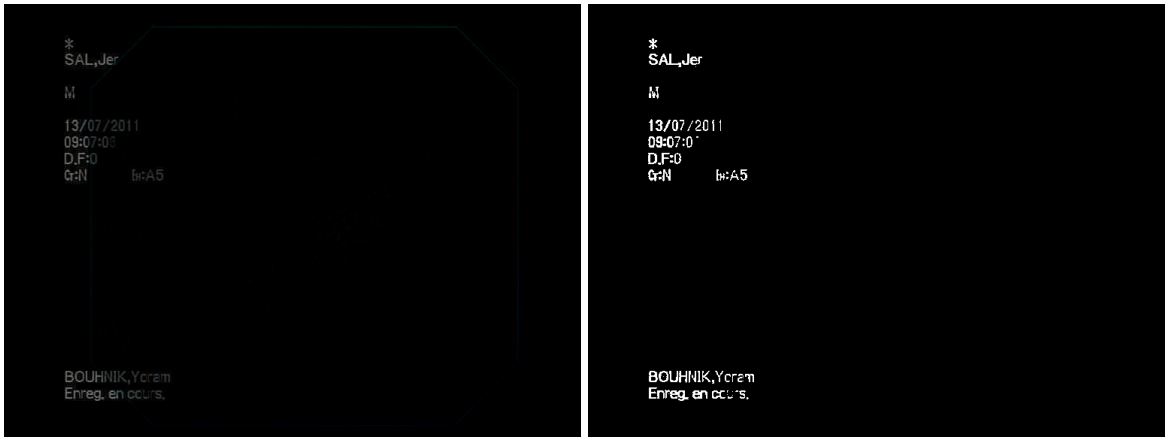


Figure 20: à gauche: image obtenue d'après appliquer la méthode *topHat*, *élément structurant* est un carré avec taille 5x5. à droite: l'image seuillée en prenant *seuil = 35*

Remarque et Conclusion

En regardant la Figure.19, l'image au départ contient aussi les *pic*(partie spéculaires) à part de textes. Afin que la méthode *topHat* ne détecte pas cette partie, on prend ainsi plusieurs frames et calcule l'image moyenne. Ensuite, le *topHat* est appliquée sur l'image moyenne, ceci a bien extrait uniquement la partie textes. A la fin, faire simplement un seuillage pour obtenir une image contrastée. En utilisant la méthode morphologique, on obtient une image quasi parfait. Aussi, la procédure est relativement rapide que celle de précédente. Ainsi, pour la détection de textes, l'algorithme morphologique est un moyen simple et robuste.

Dans la pratique, si on dispose des vidéos avec une quantité importante, l'utilisation de cet algorithme devient encombrant. Pour la généralisation, on utilise un réseau de neurones permettant de classifier les images avec et sans textes dans les vidéos.

3.1.3 Algorithme apprentissage

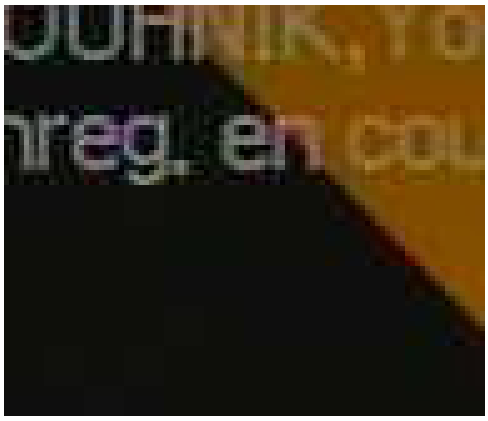
Idée principale

Afin de générer la détection de textes pour simplifier la tâche au cas où il existe une quantité important de vidéos, on s'intéresse à la conception un réseau de neurones à l'objectif de classifier les images. Pour ce faire, la première étape consiste à élaborer une base de données, ainsi on conçois les images avec textes et celles de sans textes et leurs ajouté des labels correspondants.(0 signifie avec textes, 1 signifie sans textes). La deuxième

étape consiste à l'entrainement du réseau du neurones avec des données d'apprentissages et celles de validation, à la sortie on obtiendra les *poids* et *bias* ajustés, qui sont utilisés pour l'évaluation de données de test. Tout au long au processus, on enregistre le *loss* et *score* par *époque*(une itération quand l'on parcourt tous les données d'apprentissage) pendant l'apprentissage et le test. Le *loss* est la somme des erreurs de chaque image entrée calculé par une *fonction de coût*, il s'agit de représenter le comportement du réseau(bien ou mauvais) après chaque itération. Le terme *score* n'est pas opposé de *loss* mais qui est déterminé par l'écart entre la sortie prévue et ciblé d'après chaque époque et pour l'ensemble de données de test. L'objectif est d'avoir non seulement un score assez élevé durant l'apprentissage mais aussi un score important durant le test.

Résultat numériques

Figure 21: visualisation de base de données: chaque frame est décomposée par 50 imagette, celles sont considérées avec textes en condition que le pourcentage de textes soit supérieur à 2%. Pour faciliter à l'utilisation, le nom de chaque imagette de nommée de la façon suivante: *nom de vidéo_ position d'imagette dans l'image_pourcentage de textes contient dans l'imagette*.png.



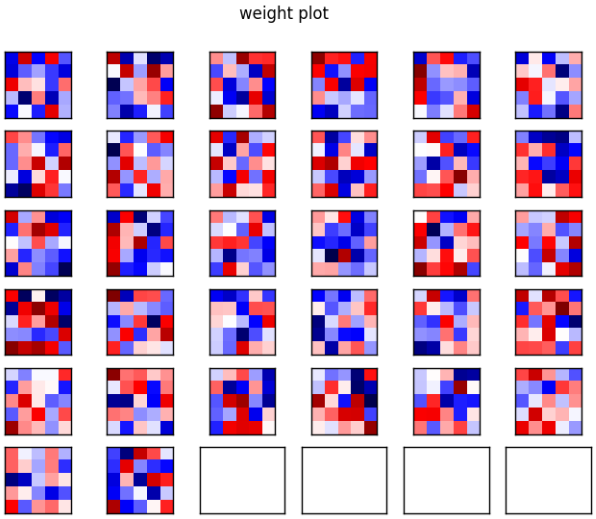


Figure 22: visualisation de noyau de convolutions lors que la classification d'une image.

À image droite, on dispose 32 noyau de convolutions, chaque noyau est représenté par 5x5 image par bloc, correspond donc 5 x 5 *poids*. Les blocs ont des couleurs claires représentent un poids plus petit, cela signifie les noyau de convolution correspondent moins avec les pixels entrées. Dans le cas contraire tel que les blocs ont des couleurs foncées, les noyau de convolutions correspondent plus avec les pixels entrées.

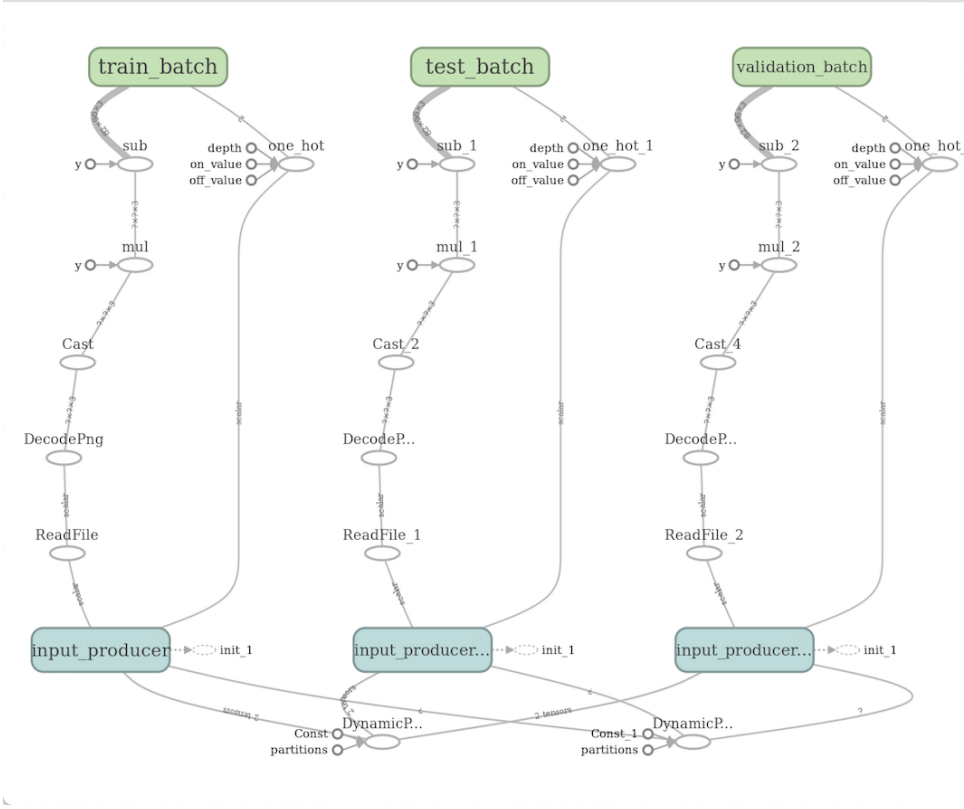


Figure 23: visualisation de la structure de la création de base de données par *tensorboard*

On crée la base de données de la manière suivante: lire des imagettes et leurs labels depuis un fichier .txt prédéfinis, et les convertir en *tensor* qui est une structure pour la représentation de données dans *tensorflow*. Ensuite, partitionner les données en 3 parties et décoder les imagettes depuis *tensor*. À la fin, définir les *BATCH_SIZE* pour chaque partition, qui signifie le nombre d'images entrées pour chaque itération. d'où les trois partitions: *train_batch*,*test_batch*,*validation_batch*.

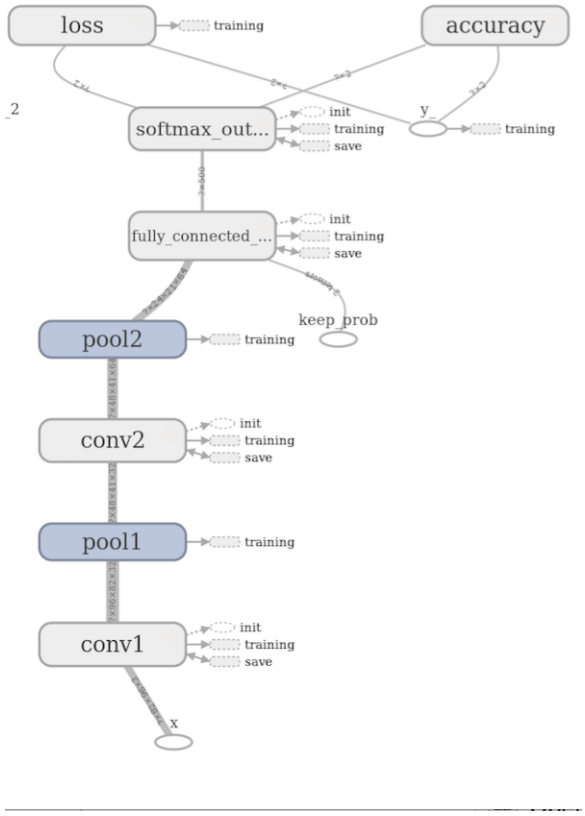


Figure 24: visualisation de la structure du réseau par *tensorboard*

Le structure représentée au-dessus est celle d'un réseau de neurones convolutif. les *conv1* et *conv2* signifient de la couche de convolution et les *pool1* et *pool2* signifient de la couche de pooling. Le nombre de noyau de convolutions associés à chaque *couche de convolution* égal 32 et 64 ainsi que le nombre de *poids* associés sont respectivement 5x5x3x32 et 5x5x32x64. Concernant la couche entièrement connectée(*fully_connected_layer*, le nombre de *poids* est définis à 500. La couche de sortie est équivalente à *softmax_output* dans le graphe et on obtiendra les résultats prévus qui sont ensuite utilisés pour calculer *score* et *loss*.

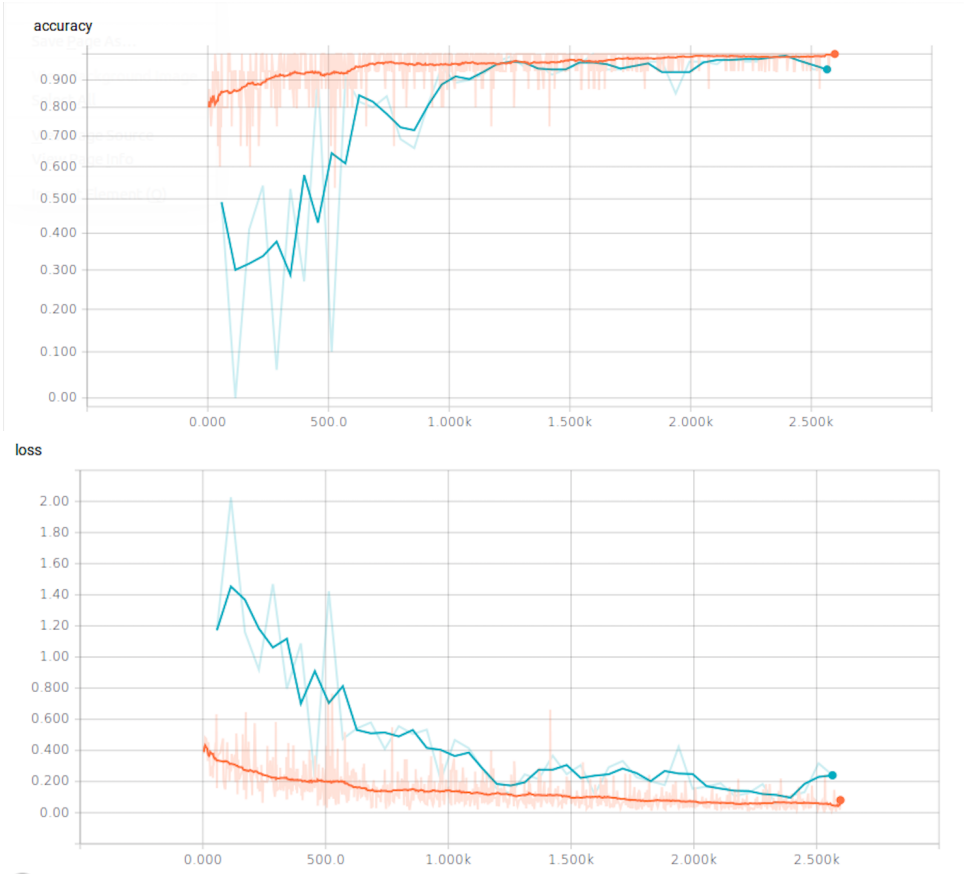


Figure 25: Résultat d'un réseau de neurones non convolutifs: les *score(accuracy)* et *loss* à chaque époque de données d'apprentissage(rouge) et de validations(bleue).Les paramètres sont définis suivant: *LearningRate=1e-4*, *BatchSize=15*, *ValidationData*=100, *TestData*=240, *TrainingData*=871.Le nombre d'époque est atteint **48** avant *overfitting*. Le *score* de données de tests égal 99.1736

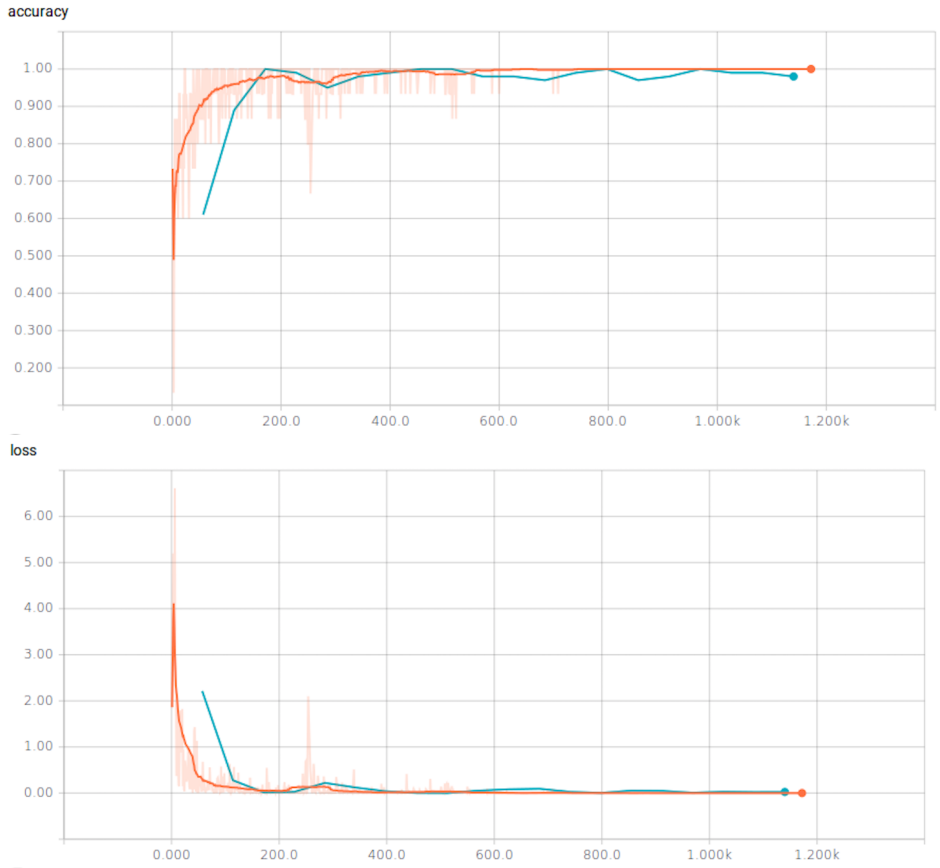


Figure 26: Résultat d'un réseau de neurones **convolutifs**: Les paramètres sont inchangés et le nombre d'époque est atteint **22** avant *overfitting*. Le *score* de données de tests égal 1.

Remarque et conclusion

D'après Figure 25 et 26, on observe que les courbes de *score* sont croissants ainsi ceux de *loss* décrois durant l'entrainement. En utilisant ces deux modèles, on a tous réussit à la classification d'une base de données d'images petites(1211 imagettes au total). Les *score* de données de validations sont inférieures à ceux de données d'apprentissage, cela est opposé dans le cas de *loss*. Ce résultat correspond tout à fait l'hypothèse puisque le *score* et *loss* de données de validation sont calculés uniquement à la fin de chaque époque, où les *poids* et *bias* sont obtenus par des données d'apprentissage. On remarque qu'en terme de performance, le réseau de neurones convolutifs est atteint un *score* de test plus élevé avec un nombre d'époque moins nom-

breux. Par ailleurs, les choix des hyper-paramètres peuvent influencer aussi le *score*. Avec un *learningRate* trop grand, le réseau perdra totalement son performance tel que le *score* est très faible.

3.2 Estimation la position de coloscope

Idée principale

Au cours d'une coloscopie, l'utilisation un *coloscope*(une petite caméra placée au bout d'un tube fin et souple contenant des fibres optiques) permet d'explorer l'intérieur de l'intestin. En général, les observations auront lieu dans la direction de *muqueuse* ou de *fond*. On suppose que les images de *fond*(les convertir à images avec *scaled colors*) peuvent être reconstruites par adapter a modèle *Gaussien 2D*, cela consiste à estimer les paramètres du modèle Gaussien 2D depuis image avec *scaled colors* à travers l'algorithme d'optimisation. Pour évaluer les précision des prédictions(images reconstruites), calculer les *écarts types des résidus* entre images avec *scaled colors* reconstruites et celles d'originales. Faisons le même procédure pour les images de *muqueuse*.

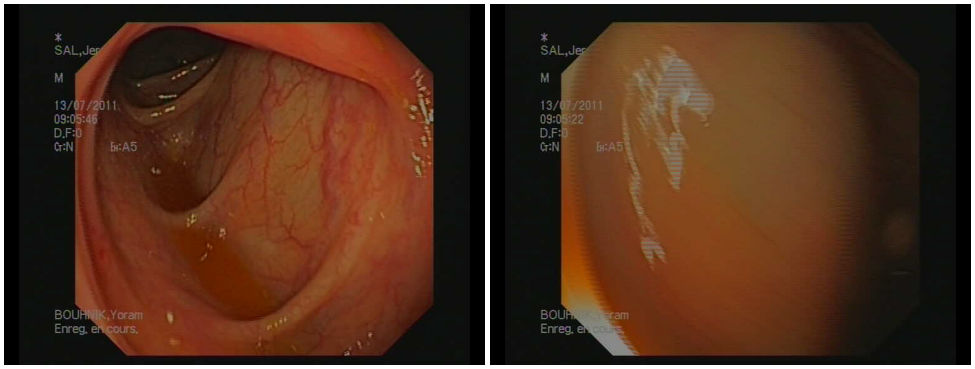


Figure 27: À gauche: image de *fond*; à droite: image de *muqueuse*

Résultats numériques

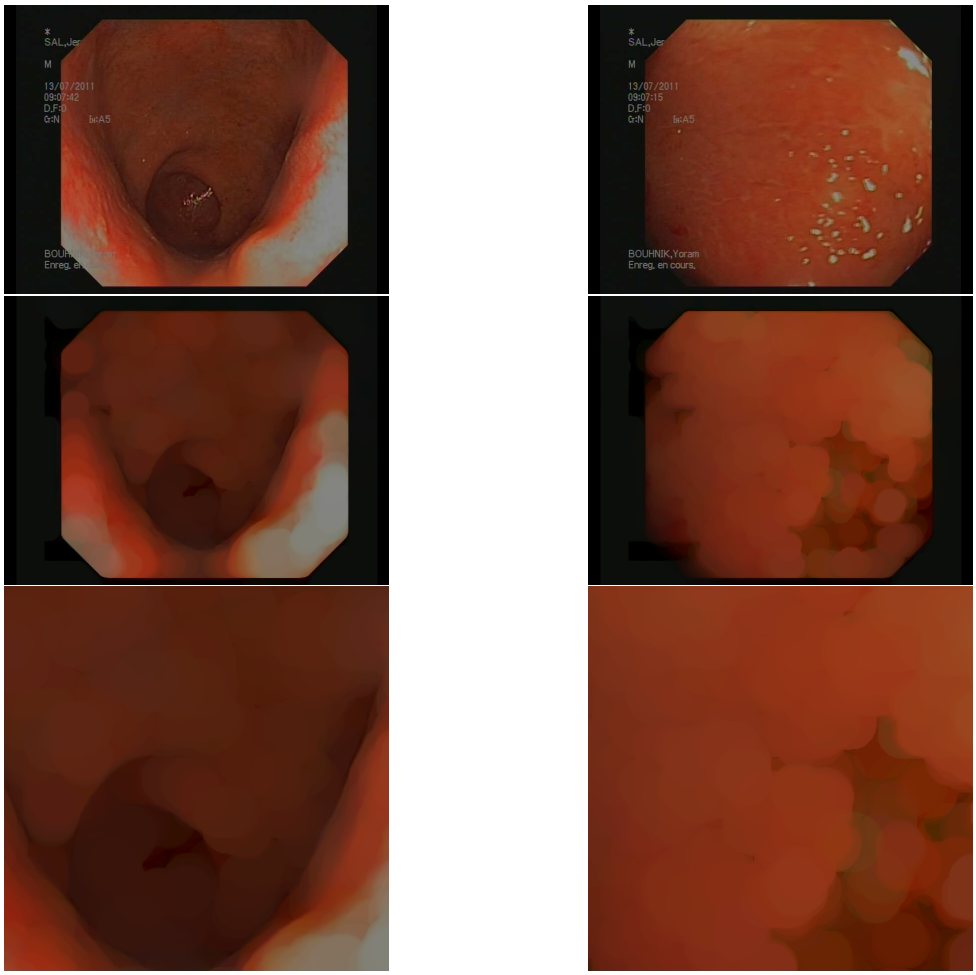


Figure 28: Pré-traitement des images.Pour chaque colonne, de haut à bas: image originale, l'ouverture d'image, découper l'image afin d'avoir uniquement la partie d'intérêt(sans l'arrière plan)

Afin d'appliquer au modèle *Gaussien 2D*, il nécessite de filtrer les images d'entrées et de sélectionner ensuite uniquement la partie dont on a besoins pour avoir un résultat moins bruité. Pour ce faire, on utilisera une *ouverture* morphologique avec un *élément structurant* assez grand, a la sortie, on obtient une image dont les détails(les *pics*) sont *effacés*. Ensuite, on sélectionne la partie où il contient le *fond* d'image. Le résultat obtenu sera converti à *scaled colors* pour *Gaussien 2D* .

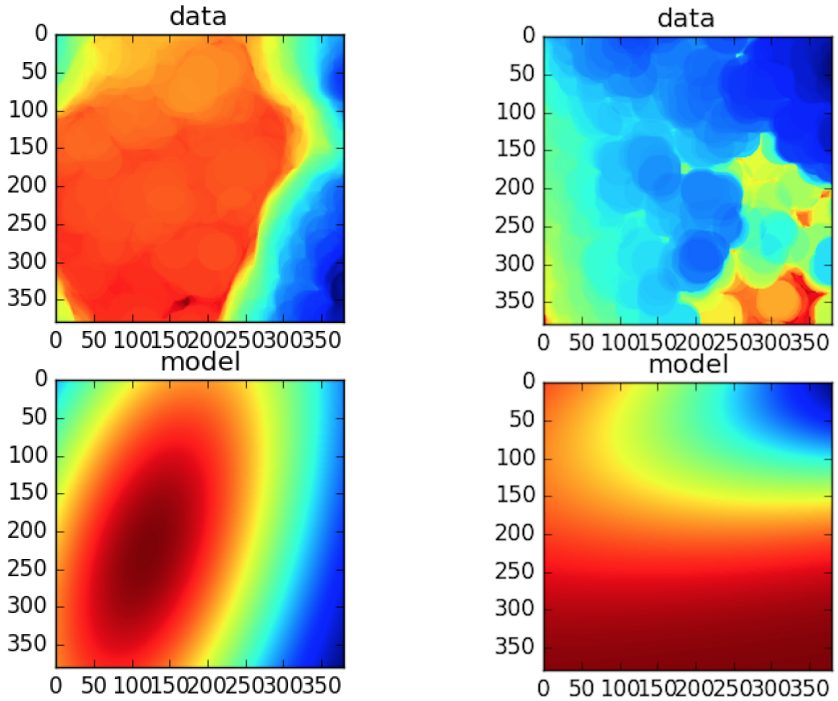


Figure 29: Colonne gauche: en haut: image de *fond* convertissant à *scaled color*(sens l'inverse); en bas: image reconstruite; Colonne droite: en haut: image de *muqueuse* convertissant à *scaled color*(sens l'inverse); en bas: image reconstruite

Rappelons le modèle Gaussien 2D en forme générale:

$$f(x,y) = Ae^{-a(x-x_0)^2-2b(x-x_0)(y-y_0)-c(y-y_0)^2}$$

où (x_0, y_0) est le centre et A désigne l'amplitude; σ_x σ_y sont l'écart type selon x et y. a, b, c sont des paramètres qui sont définis:

$$a = \left(\frac{\cos^2(\theta)}{2\sigma_x^2} + \frac{\sin^2(\theta)}{2\sigma_y^2}\right)$$

$$b = \left(-\frac{\sin(2\theta)}{4\sigma_x^2} + \frac{\sin(2\theta)}{4\sigma_y^2}\right)$$

$$c = \left(\frac{\sin^2(\theta)}{2\sigma_x^2} + \frac{\cos^2(\theta)}{2\sigma_y^2}\right)$$

Les paramètres du Gaussien $(A, x_0, y_0, \sigma_x, \sigma_y, \theta)$ seront estimée par l'algorithme *Levenberg–Marquardt*, il s'agit d'estimer les paramètres en minimisant d'une fonction. En utilisant la libraire *astropy.modeling*, on obtient aisément les résultats nécessaires. En comparant les images issues de Figure.29, on remarque que pour l'image de *fond*, la reconstruction semble bien au niveau de visuel que celle de l'image de *muqueuse*. Pour le vérifier, on calcule ainsi les *écarts types des résidus* a travers multiples images de *fond* et de *muqueuse* puis tracer les courbes.

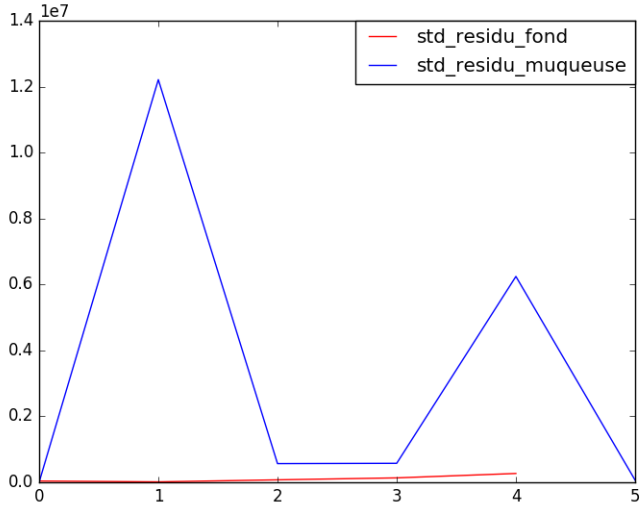


Figure 30: Les courbes écarts types des résidus selon le cas

Remarque et Conclusion

Le *écarts type des résidus* permet à évaluer la dispersion des résidus et les précision des prédictions. La plus petite cette valeur, le mieux le modèle adapte au *data*. D'après Figure.30, on remarque que pour les images de *fond*, les *écarts types des résidus* sont autour de 0 qui montre que le modèle *Gaussien 2D* a bien adapté aux images de *fond*, or pour celles de *muqueuse*, le modèle n'avait pas réussit à reconstruire l'images. Ainsi, pour l'estimation la position d'un coloscope afin de distinguer les deux cas, le *écart type des résidus* est un critère validé selon notre expérimentation.

3.3 Détection de spéculaires

Idée principale

Les spéculaires sont considérées comme des petites *'taches'* brillantes qui apparaissent et disparaissent brutalement lors d'une coloscopie. L'objectif est d'isoler ces *particules* en utilisant la segmentation morphologique: *ligne de partage des eaux*(lpe). Ainsi, durant cette opération, un des points délicats consiste à déterminer une image *marqueur*, on va tester au travers de différentes méthodes(*H-maxima*, *l'ouverture par reconstruction*, *la reconstruction inf + sup*) et évaluer leurs résultats. Par ailleurs, durant la coloscopie, les mouvement des objets entre deux frames consécutifs sont supposé stables s'il n'apparait pas de spéculaires. Lorsqu'il existe des spéculaires, on peut ainsi les détecter par mouvement en calculant du *flot optique* **dense** à l'aide de l'algorithme de Gunnar Farneback.

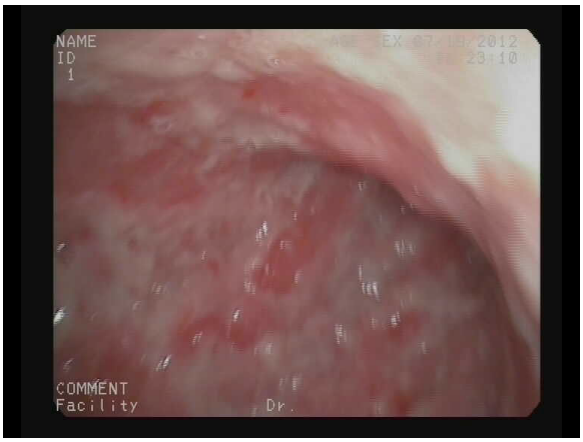


Figure 31: Un exemple des spéculaires apparaissant dans une frame

3.3.1 Algorithme morphologiques

l'ouverture par reconstruction+H-maxima +lpe

Dans un premier temps, on détermine *l'image marqueur* par la méthode H-maxima. Il s'agit de filtrer l'image par la méthode *l'ouverture par reconstruction* et calculer son image résidu, qui servira à la méthode H-maxima. Ainsi, on choisit H = 40 dans un premier temps afin de diminuer le nombre de maximum locaux, puis calculer le résidu en prenant H=1, d'où les maximum régionaux. Cela sera utilisé pour l'image marqueur et implémenté ensuite *lpe*.

l'ouverture par reconstruction+ Ostu +lpe

On dispose une autre méthode à construire l'image de marqueur, au lieu d'utiliser la méthode H-maxima, on prend le seuil automatique *Ostu* en entrant l'image résidu de l'ouverture par reconstruction. Un avantage d'utiliser cette technique est qu'il n'y a pas de paramètres à déterminer et ainsi plus robuste à priori.

reconstruction inf +sup + Ostu +lpe

La troisième méthode consiste à remplacer l'opération *l'ouverture par reconstruction* par *reconstruction inf +sup*. L'initiative est de toujours filtrer l'image entrée puis calculer son image résidu. En choisissant le dernier, on peut ainsi mettre nous même les marqueurs sur l'images durant l'opération *reconstruction inférieure* et *reconstruction supérieure*.

Résultats numériques

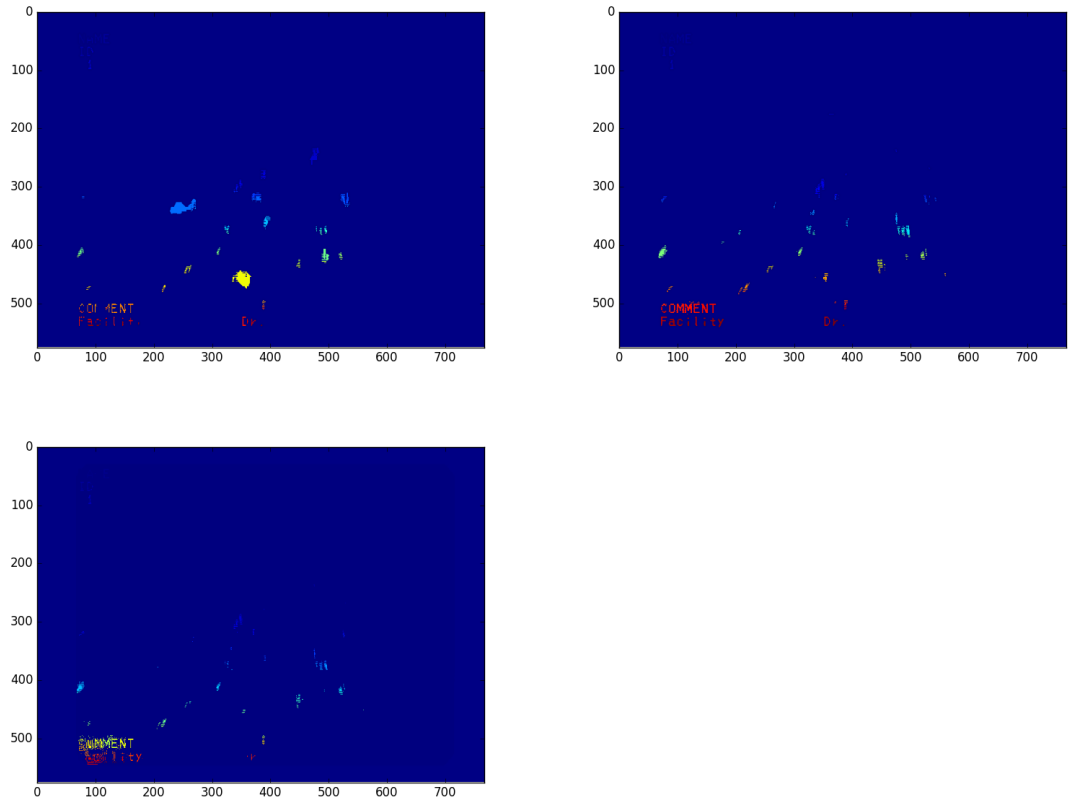


Figure 32: De gauche a droite: image marqueurs de : H-maxima, l'ouverture par reconstruction, reconstruction inf +sup

Cas d'une frame contenant les petites spéculaires

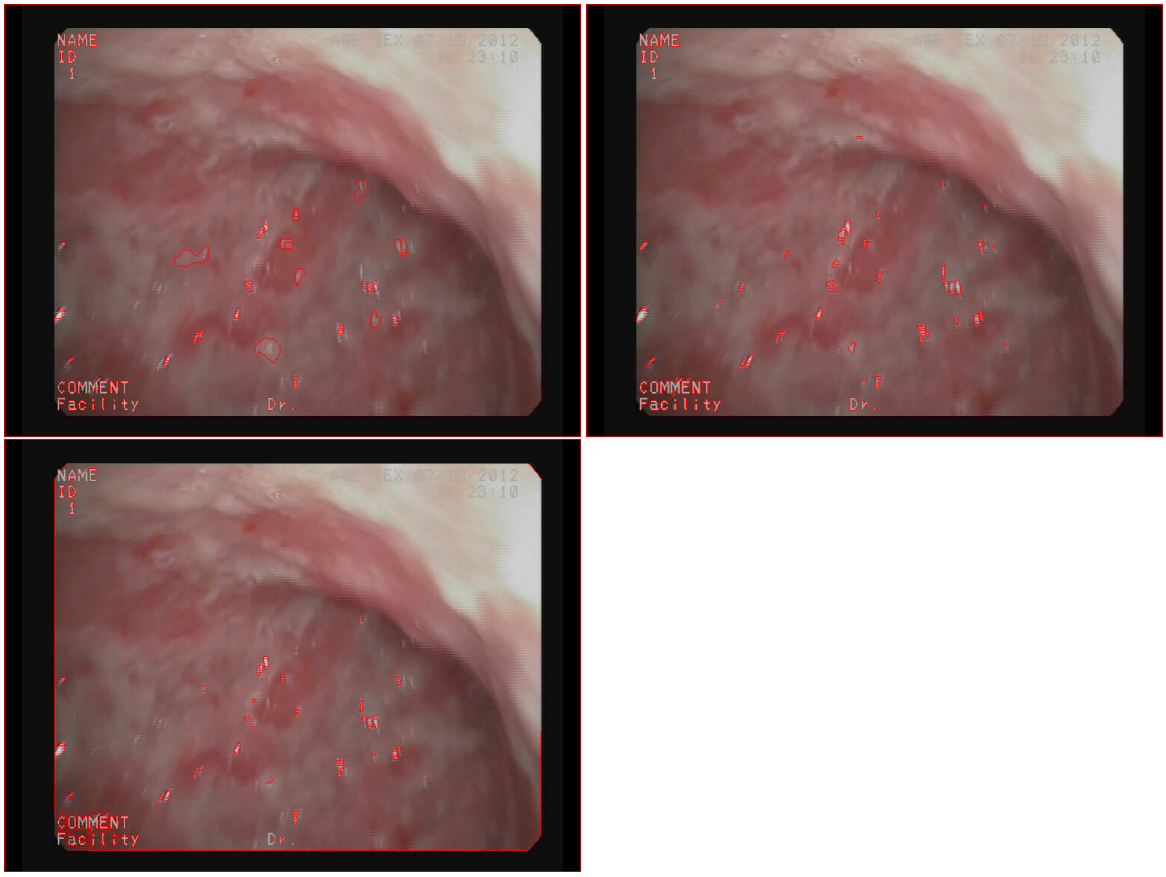


Figure 33: De gauche a droite: image d'après LPE: H-maxima, l'ouverture par reconstruction, reconstruction inf +sup

Cas d'une frame contenant les grandes spéculaires

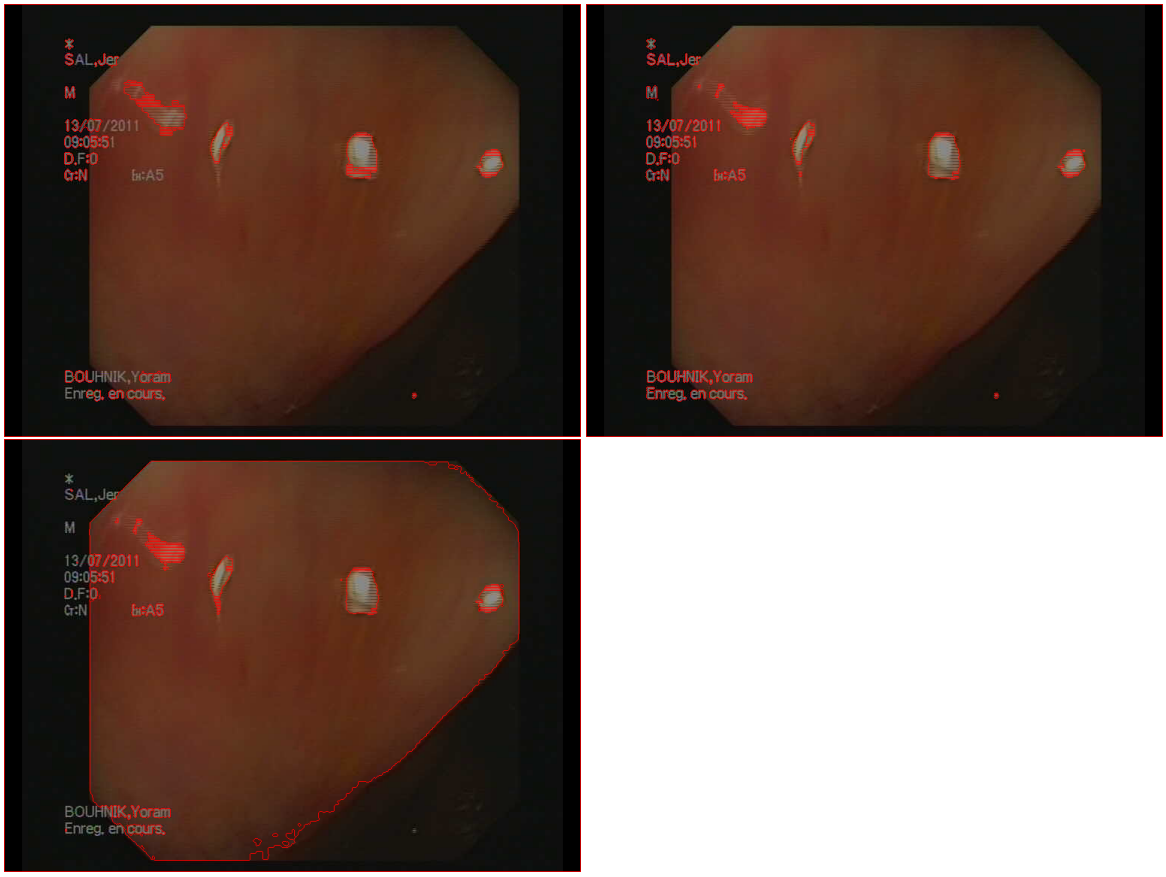


Figure 34: De gauche a droite: image d'après LPE: H-maxima, l'ouverture par reconstruction, reconstruction inf +sup

Remarque et Conclusion

La Figure.32 représente les images marqueurs obtenus dont chaque couleur désigne un marqueur qui est considérée comme *foreground*. En comparaison de l'images marqueurs, on remarque qu'en gros, ces trois méthodes repèrent les marqueurs dans tous les endroits similaires, ensuite il existe un légèrement de différence concernant la forme de marqueur. D'après appliquer l'algorithme *lpe* avec les différents marqueurs, en comparant les images obtenues, on observe que pour le cas de petites spéculaires apparaissant dans une frame, la segmentation de la méthode H-maxima est moins précise en terme de contours par rapport les autres méthodes. En plus, la méthode re-

construction inf + sup a repéré un peu moins de spéculaires que l'ouverture par reconstruction mais globalement il a réussit de détecter les spéculaires principales. Pour le cas de grandes spéculaires, en ajustant la hauteur de méthode H-maxima, les grandes *'taches'* sont mieux détectés. Pour conclure, parmi ces trois méthodes, *l'ouverture par reconstruction +Ostu+lpe* est efficace pour la détection de petites et grandes spéculaires tandis que la méthode *l'ouverture par reconstruction +H-maxima+lpe* est très efficace pour la détection de grandes spéculaires. À la fin, la méthode *reconstruction inf +sup +Otsu +lpe* marche bien en général mais il est moins automatique en terme d'exécution le programme.

3.3.2 Algorithme de calcul du flot optique

Idée Principale

Le flot optique est un champ de déplacement visuel qui permet d'expliquer des variations dans une image animée en terme de déplacement de points images. La plupart d'algorithmes calcul du flot optique font l'hypothèse que l'intensité lumineuse d'image ne varie pas au cours du temps. Ainsi, l'équation du flot optique s'écrit:

$$I_x \cdot u + I_y \cdot v + I_t = \bigtriangledown I \cdot \mathbf{v} + I_t = 0$$

où I_x, I_y ,I_t représentent respectivement: les dérivée partielles l'intensité I selon les variables d'espace x,y, et temps t, $\mathbf{v} = (u, v)^T$ désigne un vecteur de vitesse au temps t à déterminer. Pour ce faire, on utilise l'algorithme de Gunnar Farneback, qui permet de calculer le *flot optique* dense par approximer **chaque** voisinage de pixel dans les deux frames successives par un polynôme quadratique.

$$f(x) = x^T Ax + b^T x + c$$

où A est une matrice symétrique, b un vecteur et c un scalaire.Puis on estime le déplacement $\mathbf{v}$ en observant comment une translation de vecteur $\mathbf{v}$ transforme les coefficients du polynôme:

$$f_2(x) = f_1(x - \mathbf{v}) = (x - \mathbf{v} - A_1 hx - \mathbf{v}) + b_1^T(x - \mathbf{v}) + c_1 = x^T A_2 x + b_2^T x + c_2$$

Si la matrice A_1 est inversible, on obtient le déplacement $\mathbf{v}$ dont le flot optique:

$$\mathbf{v} = -\frac{1}{2} A_1^{-1}(b_2 - b_1)$$

Les coefficients peuvent ensuite estimés par la méthode moindres carrées pondérés.

Résultats numériques

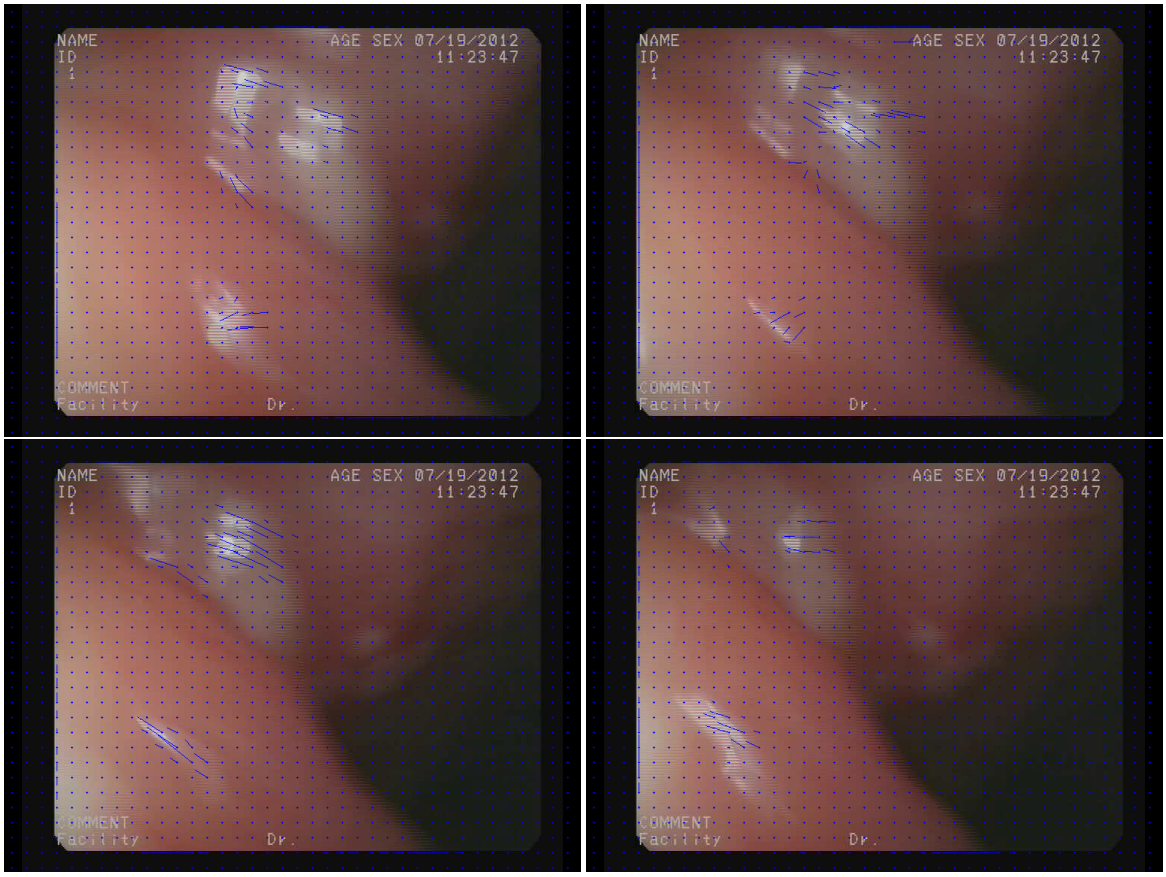


Figure 35: Résultats du flot optique détecté au zones de spéculaires(l'ordre de frame: de gauche à droite), les vecteurs représentent les directions et intensités de flot de chaque pixel

Remarques et conclusions

Le flot optique permet d'estimer le mouvement à partir de deux images successives en construisant un vecteur de mouvement par pixel ce qui fournit alors un champ de vecteurs vitesse. D'après Figure.35, on remarque bien que l'algorithme a réussi à détecté les mouvement uniquement dans la parties spéculaires pour quatre frames successives. Cela donne un autre moyen afin de détecter les spéculaires.

4 Conclusion

À travers ce stage, J'ai approfondis mes connaissances en traitement d'images avec les méthodes morphologiques ainsi que l'apprentissage automatique en traversant les vidéos de coloscopies. Notamment face aux différents problèmes à résoudre, l'objectif est d'avoir non seulement une solution mais en développer plusieurs afin de les comparer et évaluer. Dans le domaine de morphologie, on a pratiqué des méthodes que l'on a eu dans les cours, cela permet de perfectionner mes connaissances dans les différents champs d'applications. Dans le domaine d'apprentissage, d'après l'étude de l'état de l'art, j'ai ainsi une compréhension plus profonde concernant les méthodologies détaillées aux réseaux de neurones. En point d'informatique, j'ai appris la fameuse libraire de machine learning de Google *tensorflow*, cela me permet de concevoir un réseau de neurones qui facilite beaucoup au niveau technique. Aussi, durant ce stage j'ai eu beaucoup d'occasion d'appliquer le langage python à travers plusieurs bibliothèques (OpenCV, Numpy, Astropy, Matlabplotlib etc..), ce qui me permet de découvrir l'utilité de ce langage puissant.

References

[1] Michael A. Nielsen *Neural Networks and Deep Learning*. Determination Press, 2015

[2] Gunnar Farneback *Two-frame motion estimation based on polynomial expansion*. Notes in Computer Science, 2003

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* Journal of Machine Learning Research 15 (2014) 1929-1958

[4] D. Bernardini, D. Laharie *Cicatrisation muqueuse endoscopique* SpringerVerlag France 2013

[5] Qiang Wu, Fatima A. Merchant, Kenneth R. Castleman, *"Microscope Image Processing"* Academic Press, 2008

[6] Pierre Soille *"Morphological Image Analysis. Principles and Applications"* Springer, 2004