IPSI 2020

Rapport Final

Bureau d'Etude :

« Classification des images »

Réalisé par :

-BANI Hachim

-EL MEKKI Noufissa

-EDDAHMANI Ikram

-RADID Youness

-SABBAR Youssef

Encadré par : Mme FOSSATI Caroline

Table de matière :

I. État de l'art :

1. Classification : définition

La classification consiste à regrouper des individus, d'objets ou un ensemble de données dans des catégories ou des classes homogènes prédéfinis. En d'autres termes, un système de classification d'images est la reconnaissance de formes permettant à affecter automatiquement une classe à une image.

2. Intérêt de la classification d'images :

L'objectif principal de la classification est de créer un système apte de catégoriser un ensemble d'images données. Ainsi, capable de réaliser des tâches qui peuvent coûter cher à l'humain tel : la concentration, la fatigue et la durée d'exécution de données d'images.

Applications de la classification automatique des images :

- Domaine médical : Reconnaissance de cellules, de tumeurs…etc.
- Domaine d'agriculture : Reconnaissance de sol et des grains, classification d'herbes
- Domaine de Documents : Reconnaissance d'écriture pour les chèques
- Domaine urbain : Reconnaissance de panneaux de signalisation, de piétons, véhicules, localisation
- Domaine biométrie : Reconnaissance de visages, d'empreintes.

3. Types de classification :

En se basant sur la recherche que nous avons faite, il s'est avéré qu'il existe deux approches principales de classification :
Classification supervisée(ou assistée) et classification non-supervisée. Dans l'approche supervisée, chaque image est associée à une étiquette qui décrit sa classe d'appartenance. Par contre, dans l'approche non-supervisée les données sont classées automatiquement suivant des algorithmes de classification.

4. Méthodes de classification non-supervisées (auto-contrôlées) :

Les majeurs approches de classification non-supervisées sont groupées sous formes de cinq catégories, à savoir :

- Classification basée sur la connectivité
- approches probabilistes
- méthodes basées sur les centres des classes
- approches basées sur la densité
- et classification basée sur les frontières des classes.

• Méthodes basées sur la connectivité :

Cette approche de classification, dite hiérarchique, est basée sur des algorithmes de partitions imbriquées. A chaque niveau de l'arbre correspond une partition composée de classes de taille de plus en plus réduite en s'approchant des feuilles. Ce type de classification est subdivisé en deux méthodes :

Approches ascendantes :

Cette méthode est basée sur la fusion des classes les plus similaires en partant d'un ensemble de singletons. En d'autres termes, elle part d'une matrice de similarité de taille N*N soit matrice de connectivité qui permet par la suite de tracer le graphe de mesures de similarités. Ainsi, selon sa partition on peut distinguer une approche par liaison unique (on considère que la distance entre deux classes est la distance minimale entre toutes les paires d'individus inter-classes), et une approche par liaison complète (on considère la distance maximale).

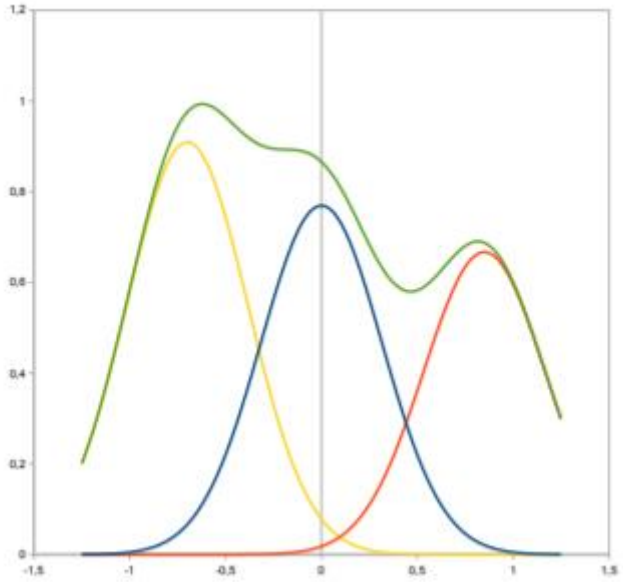
Approches descendantes :

Elle découpe récursivement les nœuds de l'arbre en des classes de population de plus en plus réduite en partant d'une classe initiale contenant tout les échantillons de la base. Cette procédure de découpage s'arrête lorsqu'on atteint la condition d'arrêt.

• Méthodes probabilistes :

Elles s'appuient sur un point de vue conceptuel. En effet, chaque classe suit un modèle inconnu dont les caractéristiques sont définies par une approche probabiliste. Autrement dit, les données, d'après ces méthodes, sont issues d'une combinaison de plusieurs classes dont les fonctions de distribution sont à caractériser.

Il existe plusieurs façons de déterminer les paramètres d'une distribution de probabilités. La plupart des travaux qui résolvent ce problème pensent que les composantes du mélange de distribution sont gaussiennes. Par exemple, dans la figure ci-dessous, la courbe verte est issue d'une combinaison de trois gaussiennes dans un espace de dimension 2.



- **Méthodes basées sur les centres des classes :**

  Ces méthodes représentent chaque classe par un point unique. On dit que ceux sont des méthodes de types « r-estimation itérative des barycentres ». Une fois les barycentres déterminés, chaque classe est identifiée par un ensemble d'individus avec les plus proches au centre qui lui est affecté. L'objective de la fonction consiste donc à mesurer la similarité entre un point et le barycentre de la classe à laquelle il appartient.

  Ces méthodes sont généralement faciles à mettre en œuvre et assez polyvalentes car elles permettent de traiter différents types de données et sont moins sensibles aux situations isolées. Parmi ces méthodes, l'algorithme K-means est la méthode de classification la plus populaire.

- **Algorithme K-means :** La méthode des nuées dynamiques tente de diviser l'espace en K classes fixé a priori.

- **K-means Intelligent :** Le principal inconvénient de l'algorithme des nuées dynamiques est qu'il dépend du paramètre K et du choix du centre initial. Cette approche a comme objectif, sélectionner les échantillons d'initialisation ou pour déterminer le nombre optimal K de classes.

- **Méthodes basées sur la densité :**

  Ces méthodes présentent certains inconvénients à savoir :

  - Non détection des classes composées de zones de densités non homogènes
  - Incapacité de gérer des classes de densités variables
  - Les classes de formes irrégulières sont souvent difficiles à interpréter.

  Elle est illustrée par l'algorithme Mean-Shift (approche itérative). Il consiste à déplacer itérativement tous les points de la base jusqu'à la convergence vers le point de densité maximale.

- **Méthodes basées sur les frontières des classes :**

  Dans cette partie, on parle des réseaux de neurones artificiels. La méthode SOM (Self-Organizing Map), ou carte auto adaptative ou carte de Kohonen, est un réseau de neurones particulièrement populaire dans le domaine de la quantification vectorielle. Cette méthode peut sélectionner les descripteurs les plus distingués et les normaliser par les poids associés aux différents neurones. Cependant, la mauvaise initialisation de ce dernier affectera significativement les résultats de la classification. Par ailleurs, la convergence de l'algorithme SOM dépend de nombreux paramètres comme le taux d'apprentissage, le rayon de voisinage, etc.

5. Méthodes de classification supervisées (d'Affectation) :

Afin de détecter efficacement les images on utilise plusieurs techniques faisant un recours au deep Learning. On devra tout d'abord essayer de trouver un moyen ou une technique pour classifier les différentes images présentes dans n'importe quelle situation. Pour faire cela on devra se focaliser sur la phase d'apprentissage qui s'avère primordiale. En effet, il existe deux principales méthodes dites méthode d'apprentissage inductif ou déductif.

✓ Méthodes d'apprentissage inductif :

Les méthodes d'apprentissage inductif reposent sur le principe de passer d'un cas général à un cas plus particulier :

• Méthode des k les plus proches :

C'est un algorithme d'apprentissage supervisé, il est nécessaire d'avoir donc de données labélisées.
A partir de ces données qui dont d'ores et déjà classé dans un ensemble dit E on pourra déterminer le label d'une nouvelle donnée n'appartenant pas à E. soit par exemple un ensemble des données classé selon deux label (étoile triangles) chacune des deux données a é caractéristiques (taille poids) ou (longueur largeur) DONC on peut les repartir dans un repère orthonormé. Soit un point présent dans ce repère dont on connait les 2 caractéristique mais on veut lui attribuer une classe qui est triangle ou Etoile.

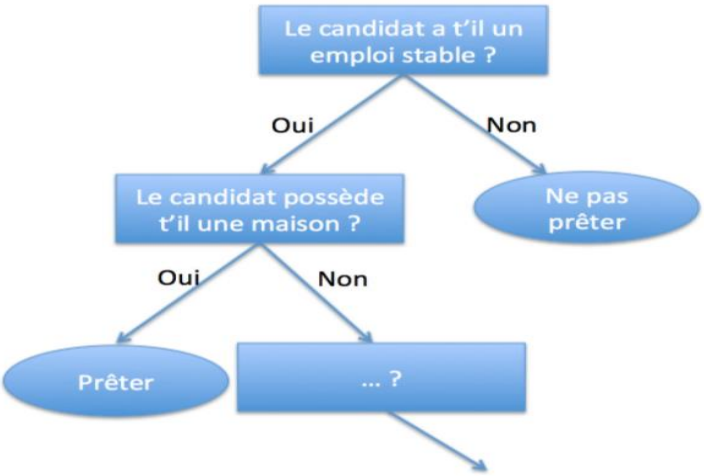
La méthode consiste à attribuer à ce nouveau point la même classe que les plus proches des points appartenant u nuage initial. Maintenant il reste à savoir combien des plus k plus proches voisins on va considérer.
On prend un nombre knn = x on affectera le label selon les plus grands nombres proche de celui-ci puis après knn= x+1.

• Affectation par l'approche d'arbre de décision :

C'est un algorithme d'apprentissage permettant d'effectuer des taches de classification et de régression. C'est un algorithme qui s'adapte à des données complexes.

En effet, Les arbres de décision (AD) sont une catégorie d'arbres utilisée dans l'exploration de données et en informatique décisionnelle. La structure des données est représentée sous formes de tests afin de prédire le résultat d'attribution à une classe.

Un exemple basique :



Chaque élément x de la base de données est représenté par un vecteur multidimensionnel (x1,x2,x3,x4....). Chaque nœud interne de l'arbre correspond à un test fait sur une des variables xi:

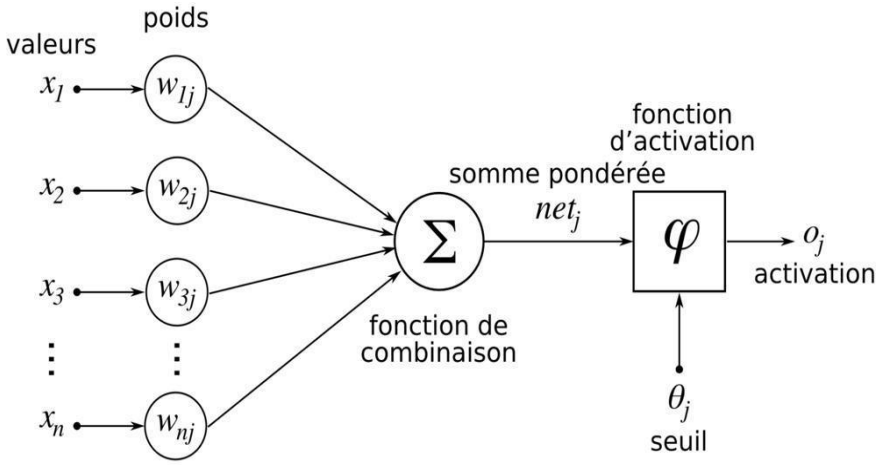
- **Variable catégorielle :** génère une branche (un descendant) par valeur de l'attribut ;
- **Variable numérique :** test par intervalles (tranches) de valeurs.
  Les *feuilles* de l'arbre spécifient les classes.
  Une fois l'arbre construit, classer un nouvel candidat se fait par une descente dans l'arbre, de la racine vers une des feuilles (qui encode la décision ou la classe). A chaque niveau de la descente on passe un nœud intermédiaire où une variable xi est testée pour décider du chemin (ou sous-arbre) à choisir pour continuer la descente.

  Au départ, les points des la base d'apprentissage sont tous placés dans le nœud racine. Une des variables de description des points est la classe du point (la « vérité terrain ») ; cette variable est dite « variable cible ». La variable cible peut être catégorielle (problème de classement) ou valeur réelle (problème de régression). Chaque nœud est coupé (opération *split*) donnant naissance à plusieurs nœuds descendants. Un élément de la base d'apprentissage situé dans un nœud se retrouvera dans un seul de ses descendants.
- L'arbre est construit par partition récursive de chaque nœud en fonction de la valeur de l'attribut testé à chaque itération (top-down induction). Le critère optimisé est l'homogénéité des descendants par rapport à la variable cible. La variable qui est testée dans un nœud sera celle qui maximise cette homogénéité.
- Le processus s'arrête quand les éléments d'un nœud ont la même valeur pour la variable cible (homogénéité).

- **L'approche des réseaux de neurones :**
  Cette approche consiste à définir des unités simples appelées neurones, chacune étant capable de réaliser quelques calculs élémentaires sur des données numériques.

  Commentaire : Commençons d'abord par donner quelques définitions relatives à la théorie des réseaux de neurones.

  Réseau de neurones : Un neurone est une unité de traitement de l'information
  Un réseau de neurones se compose de neurones connectés de façon à ce que la sortie neurone puisse être l'entrée d'un ou plusieurs autres neurones.
  On peut modéliser un réseau de neurones comme suite :



- Les poids W1j… Wnj représentent les coefficients synaptiques associés aux entrées. Chaque poids transmet une information /un stimulus provenant du neurone source Xi
- Ce stimulus (sa valeur) correspondant à l'information envoyé par le neurone source i est modulé par le poids liant les neurones i et j == » Wij* Xi
- **Σ Cellule de somation ou appelé aussi fonction de combinaison :** Chaque neurones en amont reçoit un certain nombre de valeurs via ses connexions synaptiques, il produit donc une valeur en utilisant la fonction de combinaison Σ (Wij* Xi)
- **La fonction d'activation :** la notion de la fonction d'activation ou appelé aussi fonction de transfert vient de l'idée si l'ensemble des stimuli en entrée d'un neurone atteignent son seuil d'excitabilité, alors ce neurone fournit une sortie.
- **Le Seuil Oj présente généralement 3 intervalles :**
  - En dessous du seuil, le neurone est non-actif, sa sortie dans ce cas vaut O ou -1
  - Aux alentours du seuil, une phase de transition
  - Au-dessus de seuil, le neurone est actif, sa sortie vaut 1.

Principe de cette méthode :
Le principe des méthodes de classification qui utilisent les réseaux de neurones consiste à modifier/ ajuster les paramètres des poids et des seuils par des algorithmes itératifs afin d'obtenir des réponses correctes.

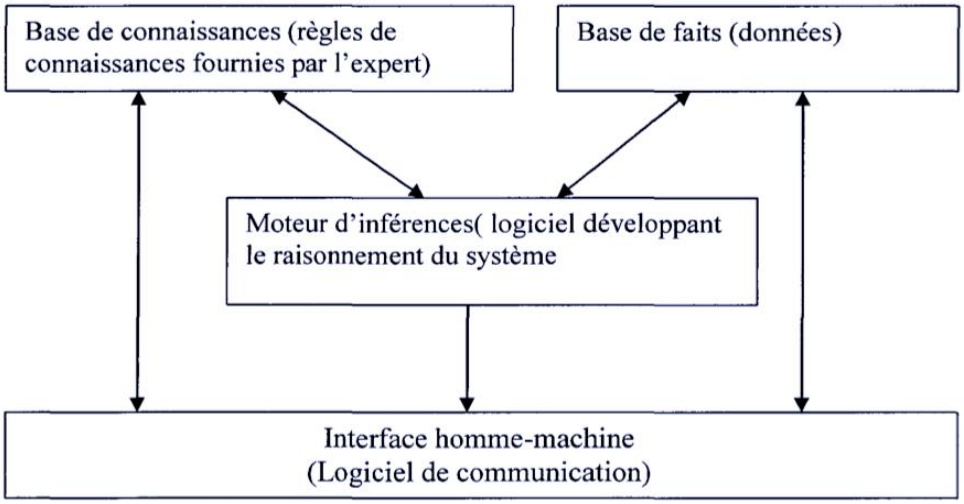
Parmi les méthodes de réseaux de neurones utilisées dans la classification des images on trouve :

- Méthode du perceptron à une seule sortie Consiste à préciser si un objet appartient à une classe donnée ou non.

✓ Méthodes d'apprentissage déductif :

Les méthodes d'apprentissage déductif utilisent un raisonnement analytique basé sur des inférences déductives a fin de transformer un ensemble de connaissance sous une forme désirée par l'utilisateur.
Parmi les méthodes d'apprentissages déductifs utilisés dans le cadre de classification des images on trouve :

Affectation par système expert : consiste à reproduire le comportement de l'expert lors de la résolution d'un problème en se basant sur une représentation des connaissances de ce dernier.
Il est nécessaire donc d'avoir une base de connaissance qui contient des règles fournit par l'expert afin de réaliser ou déclencher une action donnée.
Ainsi une base des faits qui représente des connaissances particulières en relation avec l'individu à traiter.
Finalement pour affecter les individus aux différentes classes le système cherche l'ensemble des règles applicables puis il effectue un choix. Et refaire le cycle.



II. Explication de l'article : « Réseaux de neurones convolutifs multi-échelle pour la classification cellulaire »

1. Introduction et contexte :

L'article traite des travaux qui s'inscrivent dans le cadre de l'analyse cytologique de cellules cancéreuses pour le diagnostic précoce du cancer de la plèvre. Le but de notre étude est de fournir aux pathologistes une classification automatique des cellules. Cette classification aide ensuite la cytopathologie à distinguer les cellules mésothéliales anormales et dystrophiques qui se révèlent être très importantes lors d'un diagnostic précoce des cancers liés à l'amiante.

Le but d'un outil de traitement automatique dans ce cas est de classer les centaines de milliers de cellules présente sur une lame virtuelle en différentes classes prédéfinies :

✓ Cellules malignes ;

✓ Cellules dystrophiques ;

✓ Cellules mésothéliales normales ;

✓ Macrophage ;

✓ Cellules inflammatoires ;

Le traitement complétement classique de lames virtuelles est classiquement décomposé en 3 étapes :

✓ Segmentation et extraction des objets cellulaires contenues dans la lame virtuelle ;

✓ Extraction de caractéristiques pour chaque cellule détectée ;

✓ Classification des cellules selon les caractéristiques extraites précédemment ;

L'article propose une approche basée sur la vision pour la classification cellulaire. En effet, les 2 dernières étapes du traitement automatique sont fusionnées au sein de réseaux de neurones profonds qui réalisent à la foi l'extraction des caractéristiques et leurs classifications. De plus, une approche multi-échelle est proposée utilisant plusieurs réseaux profonds à différentes, et dont les sorties sont fusionnées.

L'article est composé de 3 sections principales :

✓ **Section 1 :** Description de l'approche profonde multi-échelle MCNN ;

✓ **Section 2 :** Les expérimentations et les résultats ;

✓ **Section 3 :** Discussion et Conclusion ;

2. Section 1 : description de l'approche profonde multi-échelle MCNN ;

Convolutional Neural Network (CNN) est un réseau neuronal multicouche dédié aux tâches de reconnaissance de formes. Ils sont connus pour leur robustesse aux faibles changements d'entrée, le faible taux de prétraitement nécessaire à leur fonctionnement et Il n'est pas nécessaire de sélectionner un extracteur avec des caractéristiques spécifiques.

L'architecture proposée est basée sur plusieurs réseaux de neurones profonds, alternant entre couches convolutives et couches d'agrégation. Elle est constituée d'une série de couches de convolution et d'agrégation consécutives est consacrée à l'extraction automatique de caractéristiques au début, tandis que la seconde partie est constituée de couches de neurones entièrement connectées dédiées à la classification.

a. Couche de convolution :

La couche convolutive Ci (la i -ème couche du réseau) est paramétrée par le nombre N de cartes convolutives Mi,j, la taille du noyau de convolution Kx × Ky (généralement carré), et le schéma de connexion avec la couche précédente Li-1. Si la carte est complètement connectée aux cartes de la couche supérieure, le résultat :

Avec **: $\varphi(x) = 1.7159 \tanh(2/3\, x)$**

b. Couche de max-aggrégation :

Dans l'architecture de réseau neuronal convolutif classique, la couche convolutive est suivie par la couche de sous-échantillonnage. Cette dernière réduit la taille de la carte et introduit une invariance aux petits changements qui peuvent apparaître en entrée. La sortie de la couche max-agrégation est donnée par la valeur d'activation maximale de différentes régions de taille Kx × Ky qui ne se chevauchent pas. Similaire à la couche convolutive, un biais est ajouté et le résultat est passé à la fonction de transfert φ définie ci-dessus.

c. Couche de neurones classique :

Les paramètres de la couche convolutive et de la couche de regroupement sont sélectionnés de sorte que la taille de la carte d'activation de la dernière couche soit 1. Ensuite, une couche classique entièrement connectée composée de neurones est ajoutée au réseau pour effectuer la classification.

Dans notre travail, la dernière couche contient 6 neurones, et la fonction d'activation de type softmax est utilisée pour obtenir la probabilité d'appartenir à chaque classe.

d. CNN multi-échelle (MCNN) :

Dans ce travail, nous avons créé N réseaux de neurones convolutifs avec différentes tailles de rétine (c'est-à-dire que la taille de la rétine correspond à la taille de l'image d'entrée). La taille d'une entrée donnée sera ajustée N fois pour correspondre à la rétine de chaque matrice. Pour la reconnaissance de caractères manuscrits, le calcul de la moyenne de la sortie de chaque classificateur donne un taux de classification meilleur que celui la combinaison linéaire.

Cependant, il n'y a aucune raison de pondérer les classificateurs car ils fonctionnent à la même résolution. Dans notre cas, une hypothèse raisonnable est que les classificateurs à basse résolution peuvent être moins importants que ceux à haute résolution. Par conséquent, notre résultat final est calculé comme une combinaison linéaire des sorties de N CNN.

3. Section 2 : les expérimentations et les résultats :

3.1 Acquisition des données, segmentation et architecture du réseau :

Pour l'acquisition des données, le but était de créer une certaine base de données regroupant six entités nommées respectivement C1, C2, C3, C4, C5, C6. Après cette acquisition des données, il a fallu ensuite regrouper les différentes caractéristiques communes de ces données entre eux ce qui revient à faire une segmentation des données. En effet, la finalité de cette opération revenait à faire une séparation entre les pixels appartenant à la cellule que ceux du fond.

- Toutes les images sont prises sur une taille de 80x80
- Ce modèle choisit ne représente pas une exacte vérité

Les réseaux de neurones utilisés sont divisés en quatre types selon la résolution des images. On a trois tailles additionnelles à celle de 80x80 qui sont 56x56 40x40 28x28. Donc les réseaux de neurones sont nommés respectivement CNN80, CNN56, CNN40 et CNN20.

L'architecture utilisé est celle du réseau LeNet.

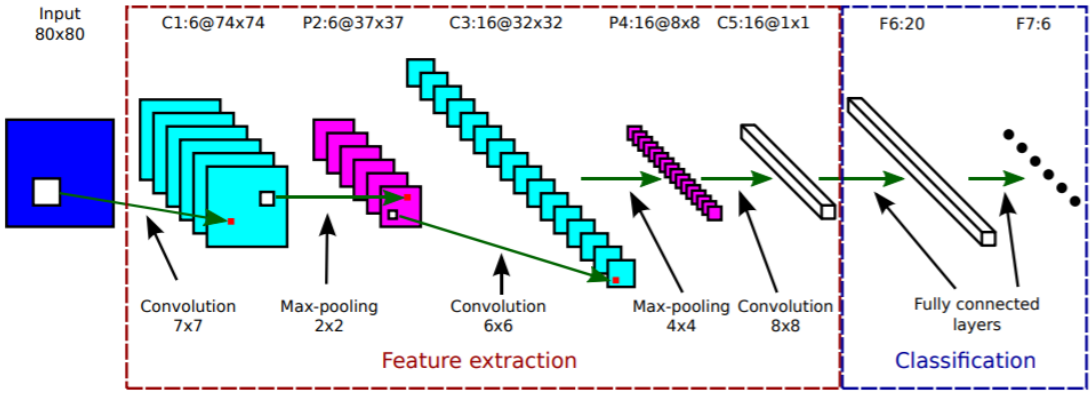


FIGURE 4 – Architecture du réseau CNN_{80}.

3.2 Approche de l'état d'art :

Cette partie repose en premier temps sur les approches classiques de la classification cellulaire et en 2eme temps sur la comparaison des résultats de ces approches avec celle présentée par l'article.

Parmi les approches classiques de la classification cellulaire on trouve les méthodes qui reposent sur :

- L'extraction des caractéristiques de forme, et de texture : surface, périmètre, compacité.
- L'extraction des caractéristiques photométriques : moyenne, écart-type des couleurs selon le spectre choisi.
- Densité optique intégré (DOI)

Pour comparer notre approche de classification basée sur un ensemble de réseaux de neurones convolutionnels (MCNN) à celle de l'état d'art, on va opter pour la méthode d'extraction des caractéristiques de forme et de texture qui donne des informations liées à la distribution de chromatine contenu dans le noyau de la cellule.

Principe :

1. On va extraire les caractéristiques couleur sur des images de partitions des cellules afin de visualiser la distribution des chromatines à l'intérieur de la cellule.

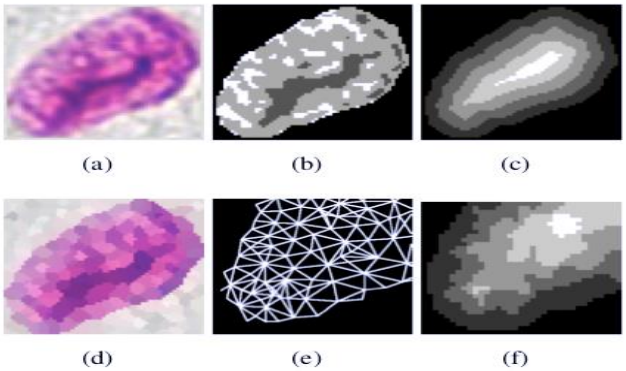


FIGURE 5 – (a) Cellule extraite, (b) image de textons, (c) ronds concentriques, (d) image partitionnée, (e) extrait du graphe d'adjacence de régions, et (f) extrait du partitionnement en régions basées graphe.

- Les cellules ont alors été divisées en rond concentriques, ou en régions géodésiques basés sur des graphes.

2. Après, on va classer ces images à l'aide d'un algorithme SVM (Support Vecteur Machine)

- SVM est un algorithme d'apprentissage supervisé qui permet de faire des prédictions. Il repose sur la présence des données d'entrés + la donnée cible afin de prédire la donnée cible.

3. On réalise une classification à l'aide de réseaux de neurones pour les mêmes caractéristiques, contenant 103 neurones en entrées, 80 neurones sur la couche cachée, et 6 en sortie.

3.3 Résultat de la classification :

L'application de la méthode de classification à l'aide des réseaux de neurones convolutionnels, la réalisation d'un apprentissage stochastique et l'utilisation des méthodes pour accélérer l'apprentissage plusieurs fois, ont permis de tracer le tableau 2 qui représente les taux d'erreur finaux de chaque approche :

TABLE 2 – Évaluation des différentes méthodes (Écart-type entre parenthèses).

Méthode	*Taux d'erreur*
CNN_{28}	9.04% (1.07)
CNN_{40}	8.02% (1.02)
CNN_{56}	8.01% (0.98)
NN	7.85% (0.98)
SVM	7.75% (0.95)
CNN_{80}	7.55% (0.89)
MV /w CNN_{28}	6.13% (0.84)
MKL	6.12% (0.83)
MCNN	6.02% (0.82)
$_{w}$MCNN	5.90% (0.80)
$_{w}$MCNN /w CNN_{28}	**5.74%** (0.79)

Plus la résolution de l'image d'entrée est grande (capacité de capturer plus d'informations), plus le taux d'erreur est faible.

Ainsi, selon les résultats obtenus, nous avons pu définir (ou tester) trois schémas de fusion :

- **Schéma MCNN :** cette approche calcule la moyenne des sorties de chaque réseau CNN.
- **Schéma wMCNN :** elle effectue la pondération des sorties de chaque réseau. Dans cette approche, le réseau de faible taux de classification reçoit le poids le plus faible et vice versa.
  Le poids de chaque CNN est calculé par la formule suivante :
  w(CNNx)=moyenne(CRx) /std(CRx) où : CRx est le taux de classification du CNNx

- **Schéma de vote majoritaire (MV).**

Il s'est avéré que la meilleure approche est celle qui ne prend pas en considération le réseau ayant la résolution la plus faible. Pour ce, le taux d'erreur le plus faible est obtenu à l'aide du deuxième schéma de fusion de données **wMCNN /w CNN28** (pondération du réseau sans prise en compte du réseau CNN28).

4. Section 3 : Conclusion

L'approche des réseaux de neurones convolutionnels traitée dans cet article a pour but d'éviter les difficultés liées aux extractions manuelles des caractéristiques telles que :

— La conception de caractéristiques robustes et faciles à calculer,

— l'évaluation de ces caractéristiques et leur pertinence pour la séparation des classes,

— et le cas le plus général de la sélection des caractéristiques pertinentes.

Ainsi, pour que chaque réseau puisse apprendre différentes caractéristiques d'une même entrée à l'aide de plusieurs réseaux de neurones demande :

- L'application de distorsions à des cellules peut changer leur taille ou leur distribution de chromatine, ce qui n'est pas fiable pour la classification.
- Un outil de segmentation robuste étant donné que les caractéristiques d'une cellule sont principalement calculées sur une version masquée.
  Par exemple, si une cellule est mal segmentée et que son masque contient des pixels du fond, sa surface sera plus grande, et une simple mésothéliale normale peut alors être confondue avec une mésothéliale dystrophique. Du coup, la segmentation doit être rapide étant donné que les lames virtuelles à haute résolution peuvent contenir des centaines de milliers d'objets cellulaires.

Pour ce, la solution proposée par l'article était l'utilisation d'une approche multi-échelle et l'implémentation d'un réseau adéquat agissant à des résolutions différentes pour une détection automatique des cellules.

Points positifs :

- Cette approche évite l'étape fatigante de l'extraction manuelle des caractéristiques.
- Elle donne les meilleurs taux de classification par rapport aux autres méthodes de classification classiques.

III. Création et implémentation du Réseau de neurone Convolutif :

Préambule :

Dans cette partie, nous allons programmer la méthode CNN_{56} pour classifier des cellules cancéreuses. Pour ce, et vu que nous n'avons pas pu trouver assez de données, nous avons créé une base de données contenant quelques cellules de l'article.

En effet, cette base de données est subdivisée en deux sous-dossiers (Test et Train) et chaque sous dossiers contient deux classes :

- Classe 1 : « Parasitized », elle contient les cellules mésothéliales anormales.
- Classe 2 : « Uninfected », elle contient les cellules mésothéliales normales.

Etape 1 : « Visualisation des données »

Comme première étape, nous avons essayé de télécharger notre base de données sur l'espace de travail. Et ceci, après importation des bibliothèques nécessaires à l'élaboration de ce programme.

```
# Téléchargement et extraction du fichier zip
import requests, zipfile, io
zip_file_url = 'https://filebin.net/whllde39ts0irnpb/Cellule_Img.zip?t=
zrb2wp5j'
r = requests.get(zip_file_url)
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
```

Ainsi, nous avons exécuté quelques commandes pour :

- S'assurer que le dossier importé est bel et bien un dossier binaire (contenant deux classes).
- Lire et voir une image de la base de données
- Vérifier le nombre d'images existantes dans la BD
- Savoir les dimensions moyennes de ces images pour pouvoir les redimensionner à la même taille car un réseau de neurone n'accepte pas des images de tailles différentes.

Dans notre cas, nous avons fixé la taille à (77, 77,3) vu que la taille moyenne de chaque classe est de (79, 75,3).

Etape 2 : « Création du modèle CNN_{56} »

Afin de pouvoir obtenir les résultats souhaités et vu que nous n'avons pas un grand nombre d'échantillons, nous avons choisi de travailler avec le modèle CNN_{56}.

	CNN_{80}	CNN_{56}	CNN_{40}	CNN_{28}
Taille rétine	80×80	56×56	40×40	28×28
C1	7×7 74×74	7×7 50×50	5×5 36×36	5×5 24×24
P2	2×2 37×37	2×2 25×25	2×2 18×18	2×2 12×12
C3	6×6 32×32	6×6 20×20	5×5 14×14	5×5 8×8
P4	4×4 8×8	4×4 5×5	2×2 7×7	2×2 4×4
C5	8×8 1×1	5×5 1×1	7×7 1×1	4×4 1×1

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
, Conv2D, MaxPooling2D

model = Sequential()

model.add(Conv2D(filters=50, kernel_size=(7,7),input_shape=image_shape,
 activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=20, kernel_size=(6,6),input_shape=image_shape,
 activation='relu',))
model.add(MaxPooling2D(pool_size=(4, 4)))

model.add(Conv2D(filters=20, kernel_size=(5,5),input_shape=image_shape,
 activation='relu',))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))

# Les couches Dropout aident à réduire l'overfitting en désactivant les
 neurones de façon aléatoire pendant l'entraînement.
# Ici nous demandons de désactiver aléatoirement 50% des neurones.
model.add(Dropout(0.5))

# Dernière couche, nous utilisons la sigmoïde vu que c'est binaire
```

```
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy']
```

Voici l'architecture du modèle créé :

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 71, 71, 50)        7400
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 35, 35, 50)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 30, 30, 20)        36020
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 7, 7, 20)          0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 3, 3, 20)          10020
_________________________________________________________________
flatten_7 (Flatten)          (None, 180)               0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               23168
_________________________________________________________________
activation_13 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 129
_________________________________________________________________
activation_14 (Activation)   (None, 1)                 0
=================================================================
Total params: 76,737
Trainable params: 76,737
Non-trainable params: 0
```

Ainsi, nous avons créé deux générateurs, un générateur d'images train et un générateur d'images test (train_image_gen et test_image_gen) pour réaliser l'entrainement du modèle.

```
train_image_gen = image_gen.flow_from_directory(train_path,target_size=
image_shape[:2],color_mode='rgb',batch_size=batch_size,class_mode='bina
ry')

test_image_gen = image_gen.flow_from_directory(test_path,target_size=im
age_shape[:2],color_mode='rgb',batch_size=batch_size,class_mode='binary
',shuffle=False)
```

Après la création des générateurs, nous avons entrainé le modèle pour qu'on puisse continuer.

```
results = model.fit(train_image_gen,
epochs=20,validation_data=test_image_gen,callbacks=[early_stop])
```

Après l'entrainement des images test :

```
Epoch 1/20
1/1 [==============================] - 1s 807ms/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 2/20
1/1 [==============================] - 0s 96ms/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 3/20
1/1 [==============================] - 0s 83ms/step - loss: 0.6931 -
accuracy: 0.5000 - val_loss: 0.6931 - val_accuracy: 0.5000
```

Etape 3 : « Evaluation du modèle »

Dans cette partie, nous avons essayé d'évaluer notre modèle de telle façon à pouvoir visualiser l'historique du modèle après exécution. Ainsi nous avons créé une variable losses qui nous permet de visualiser l'historique de l'entraînement et savoir la valeur de perte et d'exactitude dues à l'exécution.

```
model.evaluate(test_image_gen)
```

Puis, nous avons créé une variable pred_probabilities qui nous permet de savoir les prédictions de notre modèle sur l'image en entrée, et ceci, est donné avec des probabilités.

```
from tensorflow.keras.preprocessing import image
pred_probabilities = model.predict(test_image_gen)
test_image_gen.classes
predictions = (pred_probabilities > 0.5)
```

Ensuite, nous avons généré un rapport de classification qui permet de comparer les résultats corrects avec les prédictions.

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(test_image_gen.classes,predictions))
```

	precision	recall	f1-score	support
0	0.60	1.00	0.75	3
1	1.00	0.33	0.50	3
accuracy			0.67	6
macro avg	0.80	0.67	0.62	6
weighted avg	0.80	0.67	0.62	6

Etape 4 : « Implémentation et Prédiction sur une image »

Finalement, notre modèle est prêt et voici le résultat obtenu :

```
para_cell="Cellule_Img/train//parasitized/Img3.png"
my_image = image.load_img(para_cell,target_size=image_shape)
my_image
```

```
my_image = image.img_to_array(my_image)
type(my_image)
my_image = np.expand_dims(my_image, axis=0)
my_image.shape
#(1, 77, 77, 3)

(model.predict(my_image) > 0.5).astype('int32')
```

Résultat: `array([[0]], dtype=int32)`

```
train_image_gen.class_indices
{'parasitized': 0, 'uninfected': 1}
```

Conclusion :

Le modèle nous donne exactement que l'image en entrée est bien une image dans la classe Parasitized.

IV. Conclusion et discussion :

L´approche basée sur les réseaux de neurones, élaborée dans cet article est une méthode efficace qui nous permet d'éviter plusieurs problèmes tels que la conception manuelle des caractéristiques de chaque classe et l'évaluation de ces caractéristiques. En effet, le principe est que les différents réseaux de neurones traitent la même image d'entrée mais avec différentes caractéristiques (couleurs, forme…).

Dans le but de tester cette méthode, nous avons opté à traiter un seul réseau de neurones CNN_{56} parmi les quatre mentionnés dans l'article (CNN_{80}, CNN_{56}, CNN_{40}, CNN_{28}). En utilisant les fonctions de génération définies par python.

Cette approche nous a permet d´atteindre un taux de classification bien satisfaisant sur la base de données de cellules que nous avons collectés.