République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Ecole supérieur de l'informatique

Ex. INI (institut nationale de formation en informatique)

Rapport TP BDM

Classification supervisée via CNN

Réalisé par :

- Chelihi Sofiane
- Boumendjel Mohamed Islam

Groupe : SIL1

Encadré par : Mme Hamdad

Année universitaire : 2022 / 2023

Table des matières

Table des figures

1. Deep Learning

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

Le deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Chaque couche est composée de plusieurs neurones. La première couche du réseau est appelée « couche d'entrée » où le nombre de neurones dans cette couche est égal au nombre de features du dataset. Les couches intermédiaires sont intitulées « couches cachées » et le nombre de neurones dans ces couches est définie selon l'architecture. La dernière couche est appelée « couche de sortie » où le nombre de sorties est égal au nombre de classes en cas de classification ou bien un seul neurone en cas de regression. Chaque neurone reçoit ses entrées x_j de la couche précédente puis il fait une combinaison linéaire des entrées x_j et leurs poids w_j. Ensuit, il applique une fonction d'activation sur cette combinaison afin d'introduire la non-linéarité et ainsi de suite.

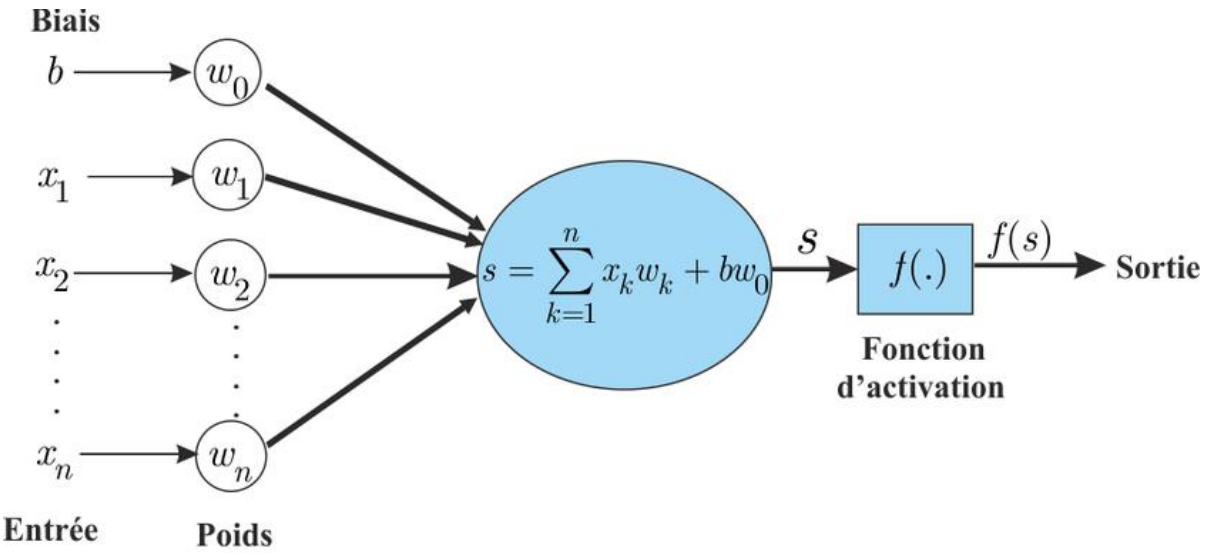

Figure 1 structure d'un neurone artificiel

Les fonctions d'activations les plus utilisées :

- $f(x) = x$| fonction linéaire
- $f(x) = \frac{1}{1+e^{-x}}$ | la fonction sigmoid pour la classification binaire
- $f(x) = \max(0, x)$ | la fonction ReLU pour annuler les valeurs négatives
- $f(x_j) = \frac{e^{x_j}}{\sum x_j}$ | la fonction softmax pour la classification multiclasses

Le deep learning est utilisé pour résoudre les problèmes complexes tels que les problèmes non linéaires, reconnaissance des images, reconnaissance de la parole…etc.

L'entrainement d'un modèle de deep learning se fait en prenant un dataset (X, Y) tel que les X sont les entrées et les Y sont les sorties. A chaque itération, on prend une entrée X_i et on la fait propager sur tous les couches du réseau de neurones jusqu'à la couche de sortie. Puis, en calcule l'erreur entre la valeur réelle de Y et la valeur prédite Y' avec une fonction de calcul d'erreur « loss function ». Cette erreur sera propagée en inverse dans le réseau afin d'ajuster les poids des entrées de chaque neurone. Ce processus se répète jusqu'à atteindre une condition d'arrêt (nombre d'itérations, stabilisation de l'erreur, valeur minimale de l'erreur…).

2. Objectif de Deep Learning

L'objectif de Deep learning est de permettre aux machines d'apprendre à faire des tâches complexes à partir d'une grande quantité de données sans être explicitement programmées.

Le deep learning utilise un réseau de neurones afin de capturer les informations des données brutes comme les images, les vidéos, les sons et les textes. Puis, les analyser et les exploiter pour réaliser une certaine tâche comme un système de recommandation, classification des images, détection des objets dans des images, reconnaissance de la parole, analyse des sentiments des commentaires…

L'apprentissage profond est utilisé par un grand nombre d'organisations, notamment des entreprises technologiques, des prestataires de soins de santé et des institutions financières. Par exemple, Google utilise des algorithmes d'apprentissage profond pour améliorer ses résultats de recherche et ses capacités de reconnaissance vocale. Dans le secteur de la santé, l'apprentissage profond est utilisé pour analyser les images médicales et identifier des modèles qui peuvent être utilisés pour diagnostiquer des maladies.

3. Types de deep learning

- **Réseaux de neurones feedforward (à propagation avant)**

Ce réseau de neurones à propagation avant est le premier type de réseau neuronal artificiel conçu. C'est aussi le plus simple. Dans ce réseau, l'information ne se déplace que dans une seule direction, vers l'avant, à partir des nœuds d'entrée, en passant par les couches cachées (le cas échéant) et vers les nœuds de sortie. Il n'y a pas de cycles ou de boucles dans le réseau.[1]

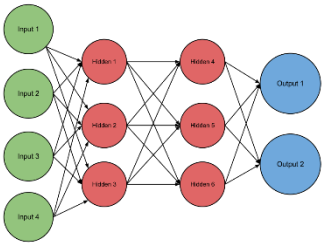

Figure 2 architecture d'un réseau de neurones feedforward

- **Réseaux de neurones récurrents**

Un réseau de neurones récurrents (RNN) est un réseau de neurones artificiels feedforward présentant des connexions récurrentes. Un réseau de neurones récurrents est constitué de couches de neurones où il existe au moins un cycle dans la structure.[2] A chaque itération, on injecte les données d'entrées ainsi que les résultats de la couche de sortie (données réinjectées) dans la couche d'entrée.

Les réseaux de neurones récurrents sont adaptés pour des données d'entrée de taille variable. Ils conviennent en particulier pour l'analyse du texte comme analyse des sentiments, de séries temporelles. Ils sont utilisés en reconnaissance automatique de la parole ou de l'écriture manuscrite - plus généralement en reconnaissance de formes - ou encore en traduction automatique.

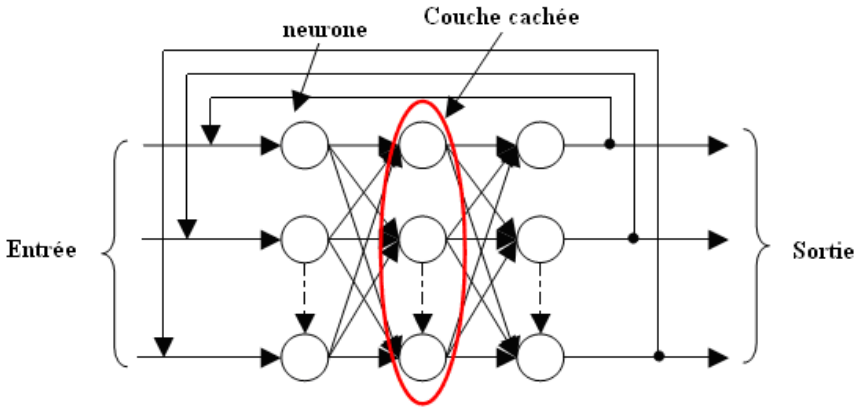

Figure 3 architecture des réseaux de neurones récurrents

[1] Réseau de neurones à propagation avant — Wikipédia (wikipedia.org)
[2] Réseau de neurones récurrents — Wikipédia (wikipedia.org)

- **Réseaux de neurones convolutifs**

Les réseaux de neurones convolutifs (CNN) sont des réseaux de neurones feedforward qui contiennent des couches de convolutions. Les couches de convolution appliquent le produit de convolution afin de filtrer et extraire les caractéristiques des données à partir des entrées brutes comme les images, vidéos ou la voix. Ils sont généralement adapté pour le traitement des images et les vidéos.

Une image est représenté par une matrice à 3 dimension (n * m * c) tel que :

- n : la longueur de l'image en pixel
- m : la largeur de l'image en pixel
- c : nombre de couches (channels). Il est en général égal à 3 (RGB : red, green, blue)

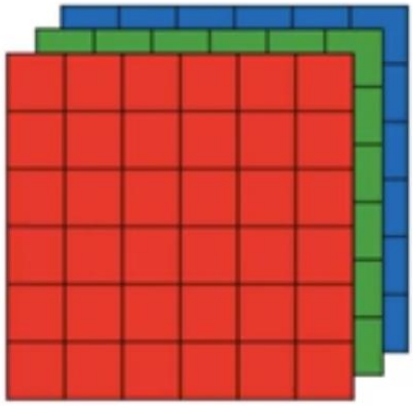

Figure 4 représentation matricielle d'une image

Une couche de convolution applique m filtres de taille (n * n). Le m et le n sont des hyper paramètres de la couche de convolution. Voici un exemple d'un filtre (3 * 3) qui détecte les bordures verticales.

1	0	-1
1	0	-1
1	0	-1

Figure 5 filtre de détection de bordures verticales

Soit a_{ij} un élément de la matrice A, et b_{ij} un élément du filtre. L'élément c_{ij} résultat est égal à

$$c_{ij} = \sum a_{ij} * b_{ij}$$

Puis on déplace le filtre par un pas s appelé « stride » et on réapplique la même formule jusqu'à parcourir toute la matrice d'entrée A. Le s est aussi un hyper-paramètres. On générale, on applique la fonction d'activation « ReLU » dans les couches de convolutions.

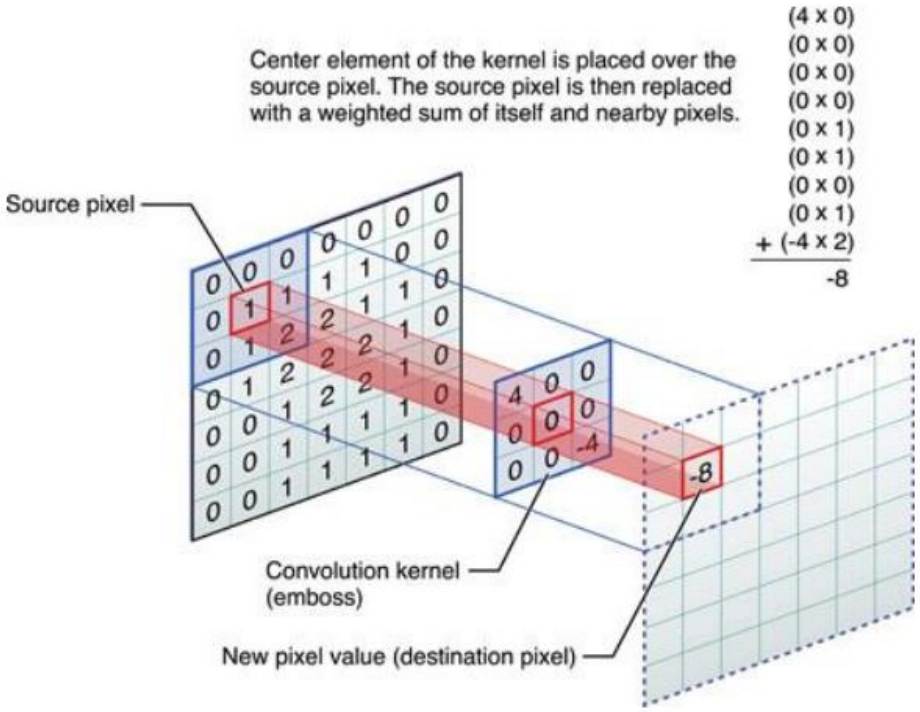

Figure 6 exemple d'un produit de convolution

Dans les CNN, il existe un autre types de couches appelées couches de « Pooling ». Ces couches vont diviser la matrice en sous-matrices et remplacer chaque sous-matrice par le résultat d'une fonction. Cette fonction peut être : max, min, moyenne. Son rôle est de compresser la matrice et garder que les caractéristiques importantes.

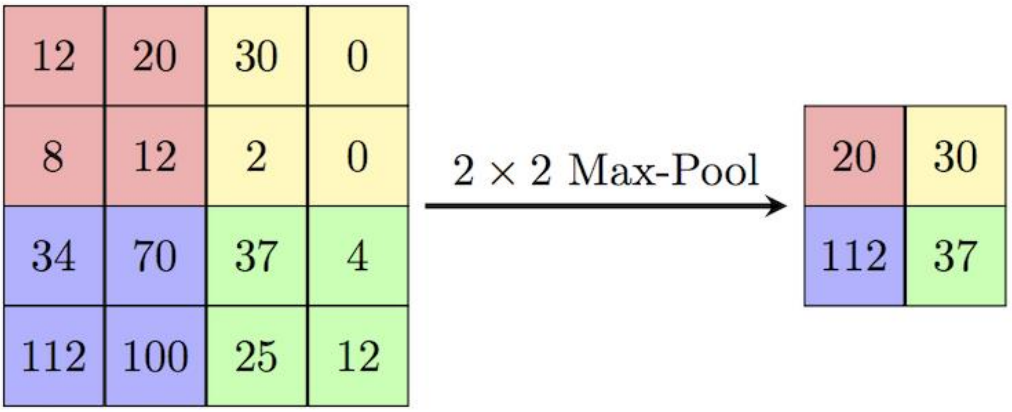

Figure 7 exemple d'une couche MaxPooling

Un troisième type de couche dans les CNN sont les couches entièrement connectées (fully connected) qui sont de même type de couches utilisées dans les réseaux de neurones feedforward.

- **LSTM**

Long Short Term Memory sont des réseaux récurrents à mémoire court et long terme ou plus explicitement réseau de neurones récurrents à mémoire court-terme et long terme, est l'architecture de réseau de neurones récurrents la plus utilisée en pratique qui permet de répondre au problème de disparition de gradient[3]. Ils sont utilisés avec les séries temporelles.

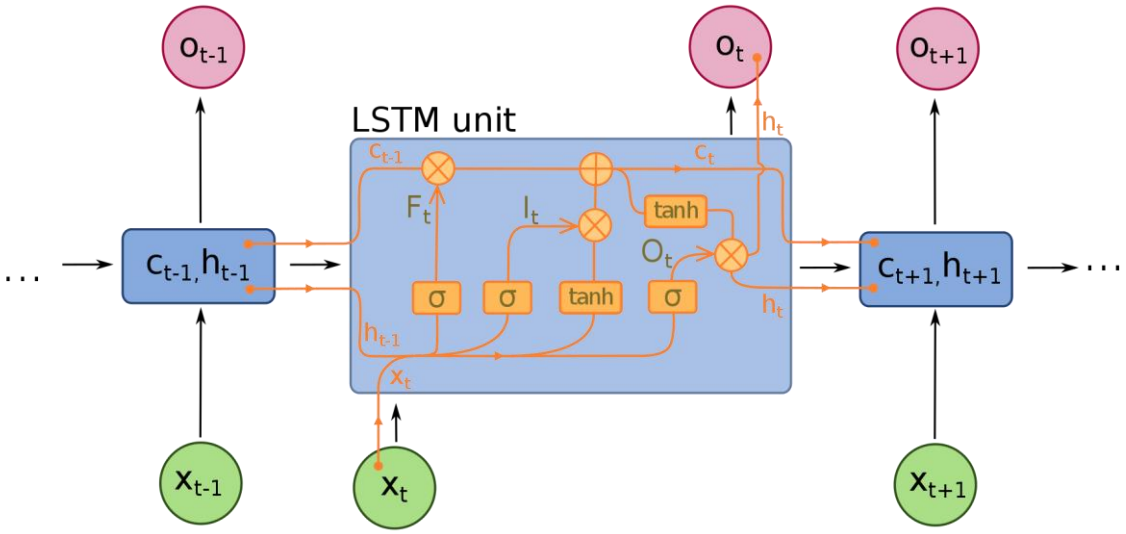

Figure 8 architecture de LSTM

- **GAN**

Un GAN est un modèle génératif où deux réseaux sont placés en compétition dans un scénario de théorie des jeux. Le premier réseau est le générateur, il génère un échantillon (ex. une image), tandis que son adversaire, le discriminateur essaie de détecter si un échantillon est réel ou bien s'il est le résultat du générateur. Ainsi, le générateur est entrainé avec comme but de tromper le discriminateur.[4]

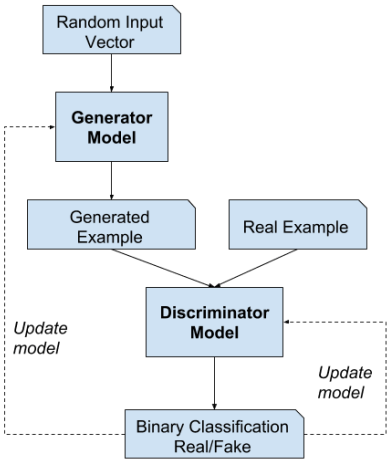

Figure 9 architecture de GAN

[3] Réseau de neurones récurrents — Wikipédia (wikipedia.org)
[4] Réseaux antagonistes génératifs — Wikipédia (wikipedia.org)

3. Caractéristiques du dataset : Recursion Cellular Image Classification

C'est un dataset des images numériques des cellules prises par un microscope électronique. Il contient 36517 images pour l'entrainement et 19899 images pour les tests. Chaque images est de dimension (512, 512, 6). Autrement dit, chaque image contient 6 canaux. Le dataset est de taille totale égale à 48.94GO.

Le problème à résoudre consiste à classifier ces images selon le type de siRNA. Le dataset contient 1108 classes. Avec le dossier des images, on a un fichier CSV qui contient des informations sur les fichiers de chaque image. (Chaque image est repartie sur 6 fichiers).

Exemple :

Id_code	Experiment	Plate	Well	sirna
HEPG2-01_1_B03	HEPG2-01	1	B03	Sirna_250

- Id_code : l'identifiant de l'image
- Experiment : le nom de l'experience
- Plate : le numéro du plat utilisé
- Well : la position dans le plat
- Sirna : la classe à prédire (la sortie)

Les fichiers sont une structure spécial, par exemple : pour récuperer l'image codé « HEPG2-01_1_B03 » , il faut aller au dossier HEPG2-01. Puis, aller au dossier « Plate1 » . Ensuite, on doit lire les images dont le nom commence par « B03 ». Enfin, on doit construire une matrice de dimension (512,512,6) où la 3ème dimension contient les 6 fichiers des canaux de l'image.

Difficultés rencontrées avec ce dataset

Pour concevoir un CNN qui fait la classification de ces images, on a utilisé Colab qui offre 100GO de stockage et 10GO de RAM ainsi qu'un accès aux CPU et GPU pour entrainer les modèles. Malheureusement, on a rencontré quelques difficultés notamment :

- Le dataset est trop volumineux aux point où le stockage offert par Colab n'est pas suffisant pour télécharger le dataset en format « zip » puis l'extraire donc on aura besoin du double de la taille du dataset voire plus de 97GO.
- On a essayé d'extraire le dossier qui contient les images d'entrainement uniquement pour ne pas consommer tous l'espace de stockage et puis diviser les images pour l'entrainement et le test. Donc, on a utilisé 70% du 36517 images (25561 images) pour entrainer le modèle et 30% (10995) pour le test.
- Puis, quand on a essayé d'entrainer le modèle, Colab a crashé car il n'a pas pu lire tous les images dans la RAM, donc on a essayé de réduire la dimensionnalité des images à (100, 100, 6) et prendre une partie des 70% images prises tout à l'heure pour ne pas trop perdre de l'information.

- L’entrainement du modèle a marché mais la précision était « 0.2 »
- On a remarqué que les classes ne sont pas balancées. Alors, on a décidé d’appliquer la méthode d’augmentation des données « data augmentation » qui consiste à générer des nouvelles images à partir des images existantes en appliquant des opérations sur les images comme : rotation, agrandissent, changement de luminosité… et attribuer les mêmes classes des images originales aux nouvelles images afin d’augmenter la taille du dataset. Mais on appliquant cette méthode, la mémoire n’est pas suffisante donc ça n’a pas marché.

C’est pour ça, on a utilisé un autre dataset d’un problème un peu similaire « classification des images des cellules infectées par Malaria »

4. Caractéristiques du dataset «Cell images for detecting Malaria »

Vous pouvez trouver le dataset sur ce lien : https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria

Ce dataset contient 27558 images où 13779 images sont pour des cellules infectées par Malaria et 13779 pour des cellules non infectées. Donc c’est un problème de classification binaire.

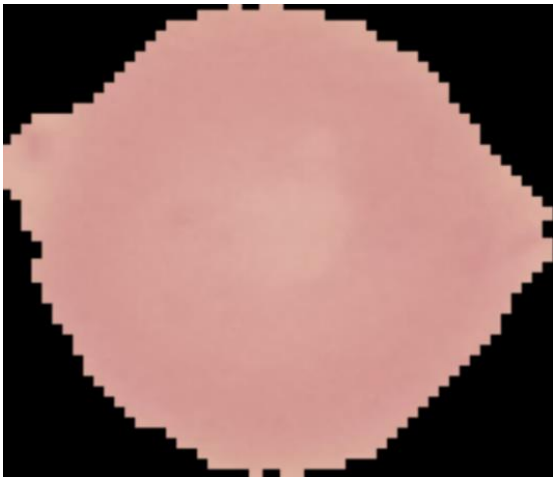

Figure 10 exemple d'une cellule non infectée

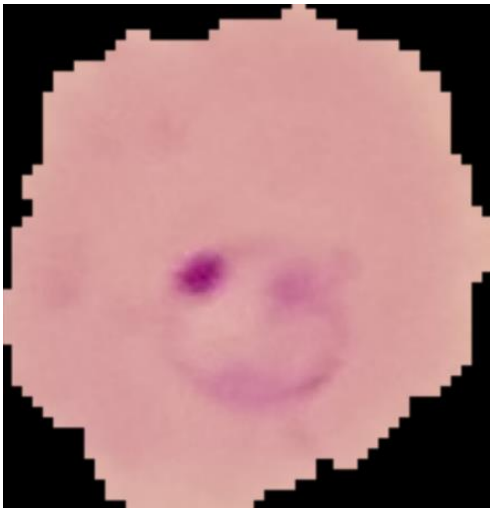

Figure 11 exemple d'une image d'une cellule infectée

Chaque image de ce dataset est de dimension (100,100,3). La taille totale du dataset est 708GO.

5. Classification des images du dataset via CNN et explication de l'architecture

Afin de faire une classification binaire de ce dataset, on est inspiré de l'architecture du CNN LeNet-5 qui est un CNN réalisé par Yann LeCun, Leon Bottou, Yosuha Bengio et Patrick Haffner dans les années 1990 pour la reconnaissance des caractères écrits à la main qui est un problème similaire à celui qu'on a traité.

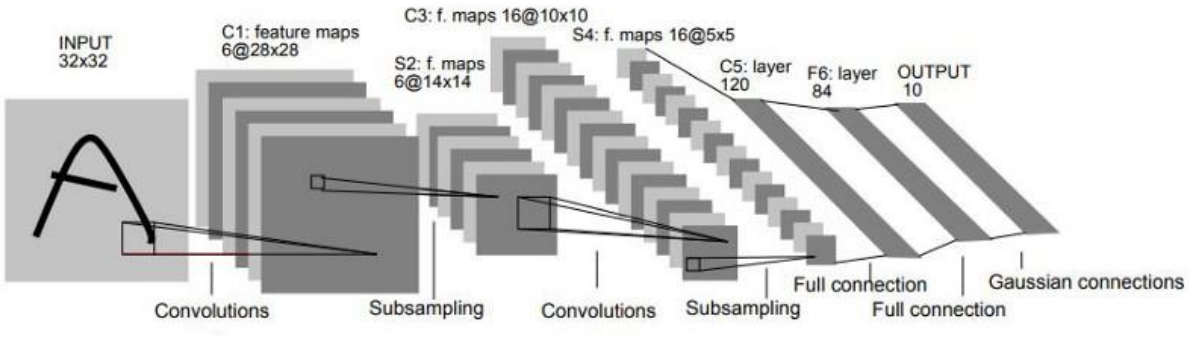

Figure 12 architecture de LeNet-5 [5]

[5] https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Figure 13 résumé de l'architecture de LeNet-5 [6]

Voici l'architecture de CNN qu'on a utilisé :

couche		Canaux	taille	activation
entrée	image	3	100*100	/
1	Convolution	32	3*3	ReLU
2	Max Pooling	/	2*2	/
3	Convolution	32	3*3	ReLU
4	Max Pooling	/	2*2	/
5	Convolution	32	3*3	ReLU
6	Max Pooling	/	2*2	/
7	Flatten	/	/	/
8	Fully Connected	/	512	ReLU
sortie	Fully Connected	/	1	sigmoid

Figure 14 architecture de CNN proposé

On a utilisé 3 couches de convolutions pour détecter les caractéristiques des images où chacune est suivie par une couche Max Pooling pour garder que les informations importantes et réduire la dimension.

Ensuite, on a utilisé une couche Flatten pour avoir une seule dimension. Puis, on a utilisé une couche entièrement connecté qui contient 512 neurones. Enfin, dans la couche de sortie, on a un seul neurone qui applique la fonction d'activation « sigmoid » car on est dans une classification binaire.

On a utilisé la fonction d'activation « ReLU » pour annuler les valeurs négatives car les images ne doivent contenir que des valeurs positives où nul.

[6] https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/

On a utilisé la méthode « Adam » pour optimiser les hyper-paramètres et la méthode « cross entropy » pour le calcul de l'erreur.

On a entrainé le modèle avec 5 epochs et un pourcentage de validation égal à 20%.

Nombre des hyper-paramètres : 1,658,817

Quand on a essayé de paralléliser l'entrainement avec PySpark, on a rencontré un problème de version où il n'était pas compatible avec la version de la librairie « tensorflow » qu'on a utilisé. Après avoir réglé ce problème, la RAM n'était pas suffisante pour l'entrainement.

Donc, on a opté pour une autre solution qui existe déjà dans la librairie « tensorflow » qui est appelé « Mirror strategy » et ça bien marché.

Lien du notebook : https://www.kaggle.com/code/medislamboumendjel/tp-bdm-malaria-cell

Résultats

Précision	0.9627
Erreur (loss)	0.1094
Précision de la validation	0.9562
Erreur de la validation	0.1323
Temps d'exécution séquentiel	643.25s
Temps d'exécution parallèle	625.86s

Figure 15 résultat du modèle

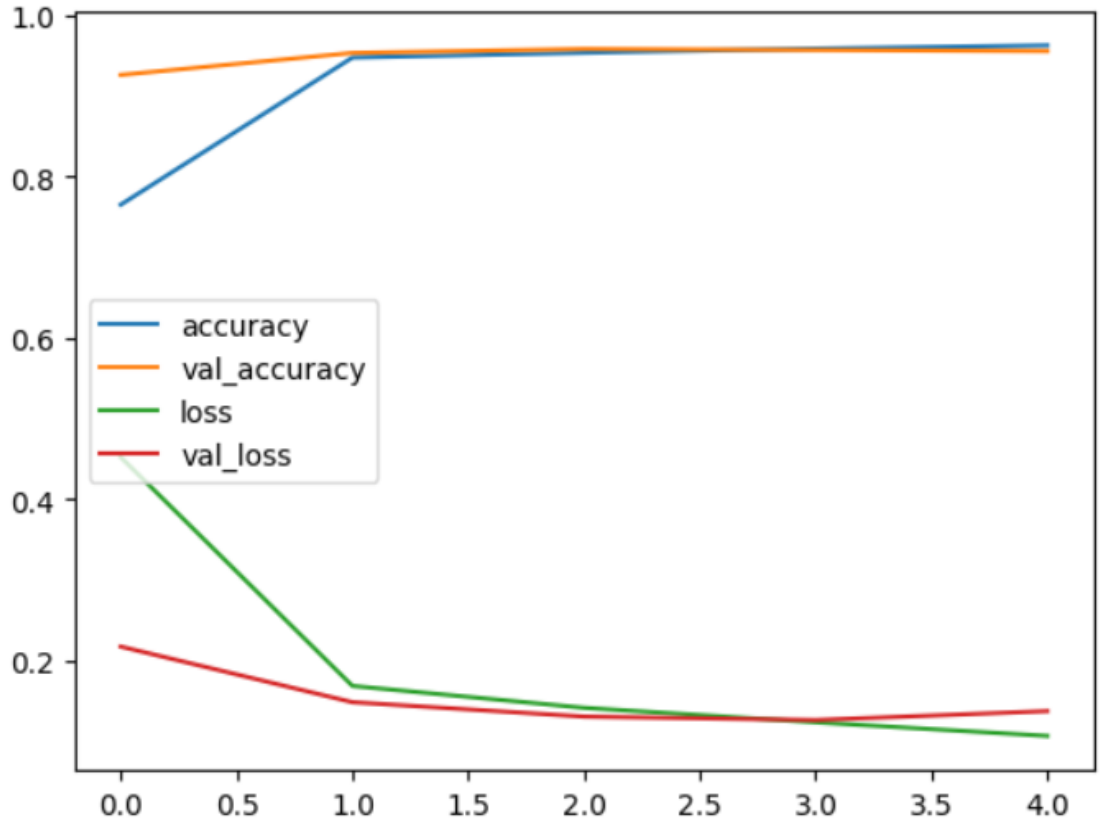

Figure 16 Evolution de la précision et l'erreur dans chaque epochs

Résultats du test

Précision	0.9562
Erreur (loss)	0.1324

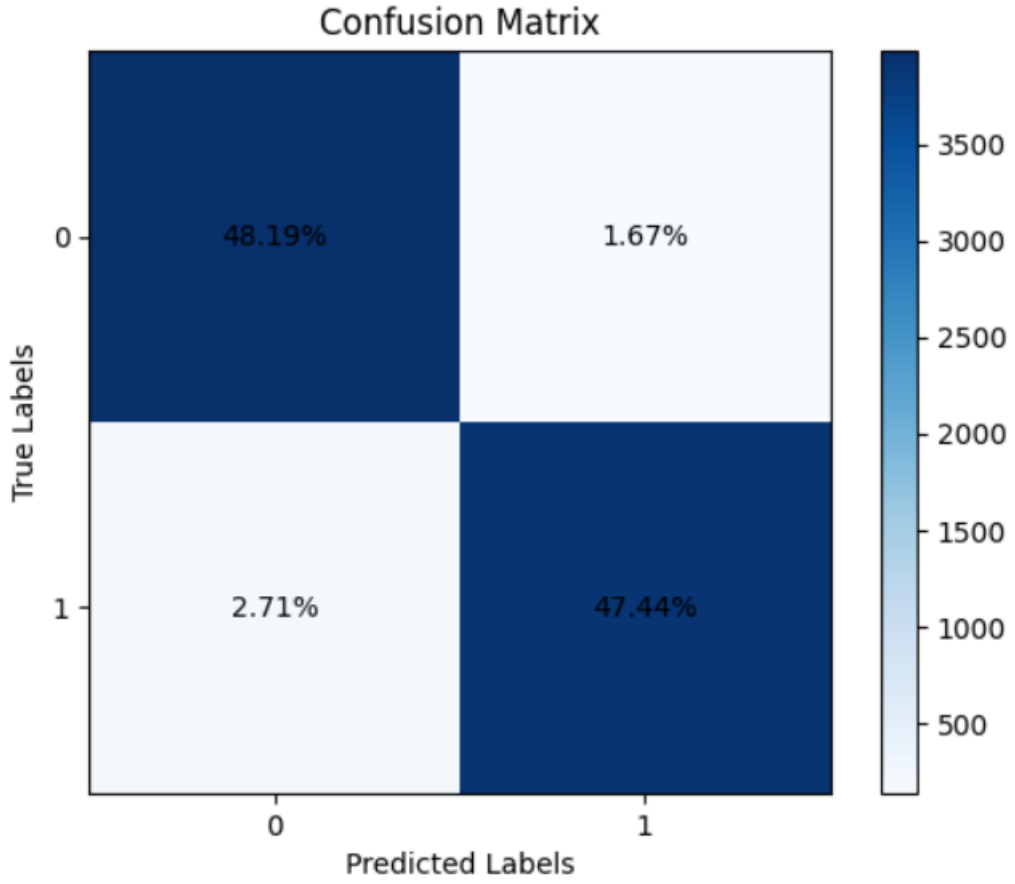

Figure 17 matrice de confusion du test