



Identification de Cellules Sanguines

Projet PYnt of Blood

Jessica Planade, Julia Canac, Richard Bonfils, Frédéric Navez

Formation Bootcamp, Mars 2023

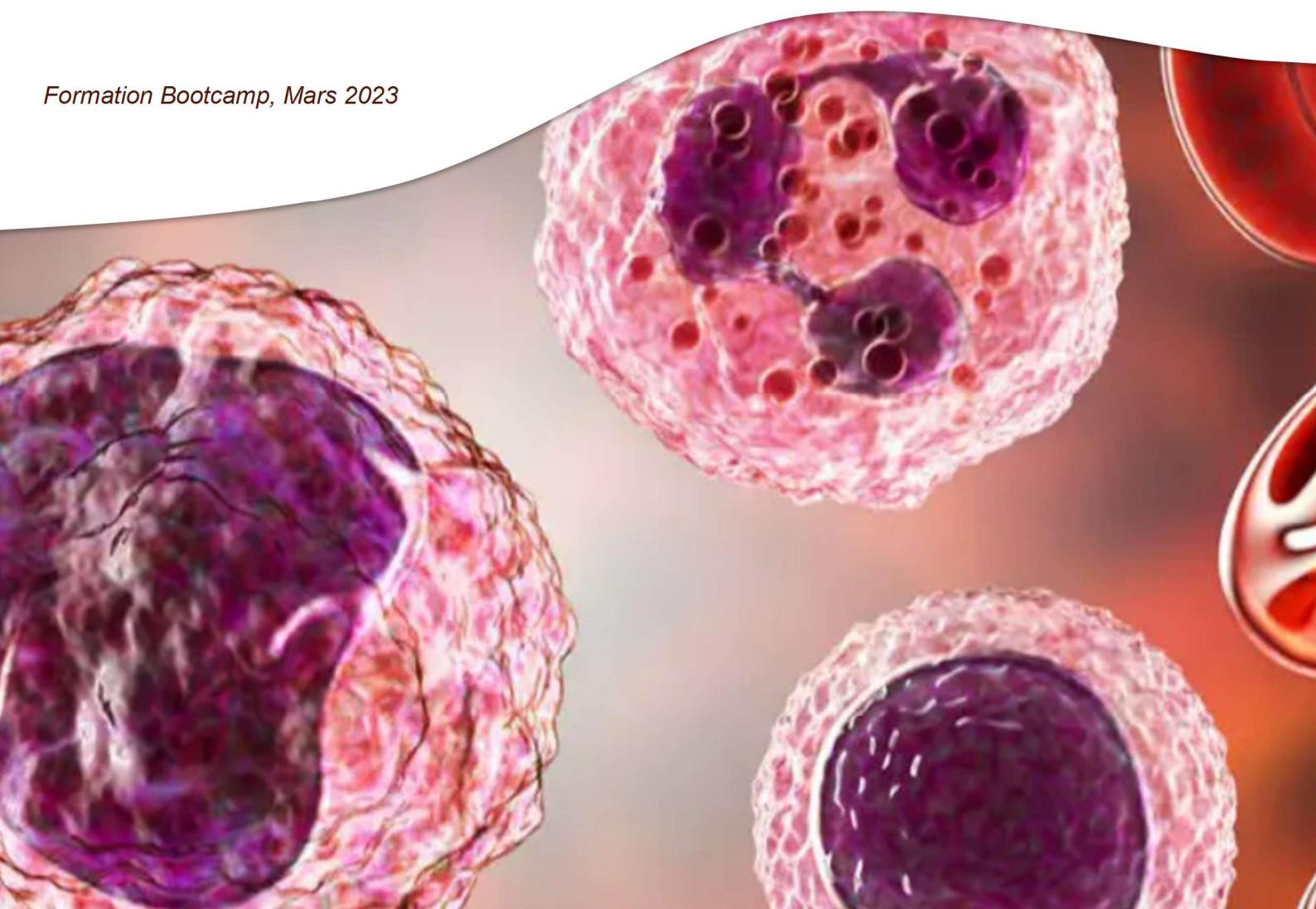


TABLE DES MATIERES

Introduction au projet.....	4
Contexte.....	4
Objectifs	5
Compréhension et manipulation des données.....	5
Cadre	5
Notions générales d'hématologie	5
Les datasetS disponibles.....	7
Premier DATASET: “Barcelona Mendeley data”	7
Second SET: Acute Promyelocytic Leukemia	7
Troisième set: Leukemia dataset (Milan)	8
DATASET complémentaire : le DATASET test	8
Pertinence	9
Data Visualisation.....	9
Résumé.....	9
Visualisations et Statistiques	11
Preprocessing	17
Machine Learning (ML).....	20
Méthode des K plus proches voisins, KNN	20
Régession logistique	24
Combinaison avec VotingClassifier	26
Random Forest (Forêts aléatoires).....	27
Conclusion	27
Deep Learning (DL).....	28
Réseaux de neurones convolutifs (CNN).....	28
Architecture lenet.....	29
CNN 1.....	30
CNN 1 - Variant	31
CNN avec des couches d'activation variées	32
Utilisation des réseaux de neurones pour classer les images à partir de données réduites ...	33
Transfer Learning (TL).....	36
EfficientNet.....	36

VGG16	38
DenseNet121	39
(Google) Inception V3	40
DEVELOPPEMENT.....	42
Annexes.....	42
Annexe 1 : Extrait étude (Zahra Mousavi Kouzehkanan, 2022),	42
Annexe 2 : Résumé des performances des modeles.....	44
References et bibliographie	50

INTRODUCTION AU PROJET

CONTEXTE

Le diagnostic d'un cancer nécessite plusieurs examens dont l'un des plus importants est l'examen histologique. Un prélèvement ou une biopsie est effectué et une lame est observée par microscopie, ce qui permet de confirmer ou non la présence d'anomalie, identifier les cellules saines ou touchées et poser définitivement le diagnostic.

Un dépistage réalisé tôt permet de meilleures chances de survie. Cependant, le dépistage est un processus lourd, long et pouvant être sujet aux erreurs. L'intelligence artificielle peut aider à l'analyse d'image. Un modèle bien entraîné peut obtenir des résultats au moins aussi fiables qu'un professionnel de santé et dans un temps réduit, par l'analyse de plusieurs milliers d'images en simultané. Cette aide, la confirmation du diagnostic étant laissée au professionnel, est d'autant plus utile dans le contexte de tension des ressources en personnel hospitalier.

Dans le cas de l'hématologie, qui concerne ce projet, la numération et formule sanguine est réalisée de manière routinière par des automates utilisant les technologies d'impédance électrique et de cytométrie de flux. En cas de résultats anormaux, l'observation de lames est souvent nécessaire pour identifier et / ou confirmer l'anormalité.

L'importance du modèle qui sera mis en place va résider, dans un premier temps, dans sa capacité à différencier le type des cellules présentes dans le sang, puis dans un second temps, à déterminer si une cellule est cancéreuse ou non.

Hypothétiquement, en plus de ces deux objectifs, un modèle bien entraîné à reconnaître des cellules saines et malades pourrait mettre en lumière des paramètres encore inconnus ou simplement non encore quantifiés entre caractéristiques morphologiques des cellules et propriétés (type de cellule, état sain ou malade, éventuellement l'âge, etc.).

OBJECTIFS

L'objectif primaire est de choisir et d'entraîner un modèle adapté à la distinction des différentes modalités de la catégorie « cellule sanguine » à partir de photographies de bonne qualité (cellules à identifier relativement centrées, image peu/pas bruitée) d'échantillons préparés pour analyse (noyaux cellulaires colorés). On privilégiera le dataset Barcelone pour cela.

L'objectif secondaire serait ensuite d'entraîner un modèle capable d'identifier si un type cellulaire présente une anomalie et donc s'il y a présence ou non d'un cancer et quel type cellulaire est touché.

Ce projet vise à développer un modèle de computer vision afin d'identifier les différents types de cellules du sang. Des expériences ont été menées sur un ensemble de 17092 images de cellules sanguines avec leurs sous-types. Une architecture basée sur des algorithmes de Convolutional Neural Network (CNN) sera construite, par la suite, pour classer automatiquement les cellules sanguines. En amont, l'utilisation de modèle de Machine Learning nous permettra de donner un premier aperçu et d'orienter notre travail vers de l'apprentissage profond.

COMPREHENSION ET MANIPULATION DES DONNEES

CADRE

Selon les objectifs primaires et secondaires, des dataset différents vont être utilisés.

NOTIONS GENERALES D'HEMATOLOGIE

Il nous paraît utile de résumer les différentes catégories de cellules sanguines existantes et les différentes lignées de cellules :

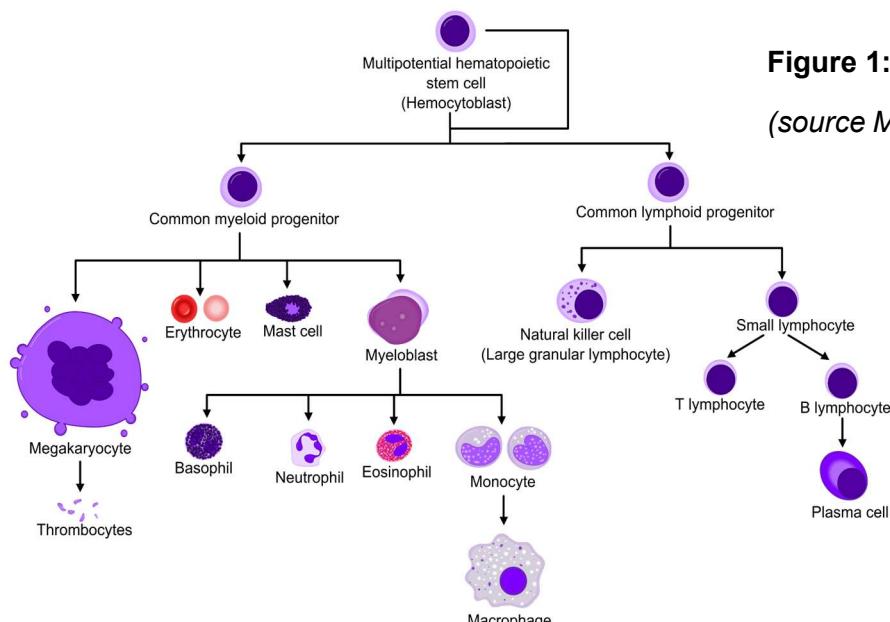


Figure 1: Hématopoïèse
(source Myélocyte - Wikipedia)

Les myélocytes sont un stade du développement normal des granulocytes, un type de globules blancs. Les granulocytes comprennent les neutrophiles, les éosinophiles et les basophiles, qui sont responsables de la lutte contre les infections et participent à la réponse immunitaire du corps.

Les myélocytes correspondent au stade du développement des granulocytes faisant suite aux promyélocytes, en amont du processus de développement. Les myélocytes sont caractérisés par la présence de granules spécifiques dans leur cytoplasme, qui contiennent des enzymes et d'autres substances qui aident la cellule à fonctionner.

D'autre part, les granulocytes immatures (IG) sont une catégorie plus large qui comprend non seulement les myélocytes, mais aussi des stades antérieurs du développement des granulocytes, tels que les promyélocytes et les métamyélocytes. Les granulocytes immatures présentent les caractéristiques suivantes lors de l'observation au microscope : une taille de cellule plus grande, une structure nucléaire moins condensée et des granules moins développés dans le cytoplasme par rapport aux granulocytes matures.

Dans certains contextes cliniques, le terme "granulocytes immatures" peut être utilisé de manière interchangeable avec "myélocytes", mais strictement parlant, les myélocytes sont un stade spécifique du développement des granulocytes, tandis que les granulocytes immatures englobent une gamme plus large de stades, y compris les promyélocytes et les métamyélocytes.

Par ailleurs, nous indiquons ci-dessous les principales caractéristiques morphologiques des différentes catégories de cellules :

- Granulocytes : ils présentent de petits grains dans le cytoplasme et autour du noyau ;
- Neutrophiles : leur couleur est plus pâle que les globules rouges qui l'entourent. Le noyau est scindé en 3 parties arrondies accrochées les unes aux autres. Ils font en moyenne deux fois la taille des globules rouges ;
- Eosinophiles : ils sont plus rouges et ont un noyau scindé en 2 parties rondes régulières ;
- Basophiles : ils sont en général violet foncé, et deux fois plus grands que les globules rouges. Leur noyau est difficilement différenciable du fait de leur couleur ;
- Lymphocytes : de couleur violette, ils sont de taille similaire aux globules rouges et présentent un halo ;
- Monocytes : leur noyau est en forme de rein et ont une taille qui va de quatre à cinq fois celle des globules rouges.

Bien qu'il soit attendu que les modèles d'apprentissage puissent détecter par eux même ces caractères spécifiques, nous les gardons en mémoire pour référence et future considération.

LES DATASETS DISPONIBLES

3 DATASETs sont référencés dans la présentation du projet :

PREMIER DATASET: "BARCELONA MENDELEY DATA".

Ce dataset provient d'un article scientifique "A dataset for microscopic peripheral blood cell images for development of automatic recognition systems". C'est un dossier contenant des images au format .jpg et de taille 363x360 en RGB. Les images, de taille normalisée, sont réparties dans 8 dossiers différents, correspondant aux catégories de cellules suivantes: basophiles, éosinophiles, érythroblastes, granulocytes immatures (IG), lymphocytes, monocytes, plaquettes et neutrophiles.

Les IG (Immature Granulocytes) sont les précurseurs des granulocytes, caractérisées par la présence de granules dans leur cytoplasme et jouant un rôle en cas d'inflammation ou d'infection. Ils contiennent les myélocytes.

Ce dataset permet d'étudier toutes les cellules de la lignée des globules blancs.

Toutes les images sont issues de patients sains (absence de maladies, infections, allergies). Ce dataset va donc permettre l'entraînement d'un modèle à la reconnaissance des cellules du sang

SECOND SET: ACUTE PROMYELOCYTIC LEUKEMIA

Ce dataset contient un fichier au format .csv "master.csv" avec 5 colonnes. La première correspond aux identifiants du patient, la seconde au diagnostic, la troisième à la cohorte, la quatrième à l'âge, et la dernière au sexe.

La colonne diagnostic contient uniquement des diagnostics de cancer qualifié de APL pour Acute Promyelocytic Leukemia ou AML pour Acute Myelocytic Leukemia. Ces cancers touchent des cellules pro-myéloïdes ou myéloïdes.

En plus de ce fichier .csv, il est également fourni une banque de 25915 images réparties en 23 catégories :

- | | | | |
|-------------------|---------------------|----------------|----------------------|
| ➤ Arifact | ➤ no lineage spec | ➤ Smudge cells | ➤ Lymphocyte |
| ➤ Band | ➤ Eosinophils | ➤ Thrombocyte | ➤ variant |
| ➤ neutrophils | ➤ Erythroblast | ➤ aggregation | ➤ Metamyelocyte |
| ➤ Basophil | ➤ Giant thrombocyte | ➤ Promonocyte | ➤ Monocyte' |
| ➤ Blast | ➤ Lymphocyte | ➤ Promyelocyte | ➤ Myelocyte |
| ➤ Unsigned slides | ➤ Plasma cells | ➤ Segmented | ➤ Patient_100 |
| ➤ Unsigned | ➤ Prolymphocyte | ➤ neutrophils | ➤ Young Unidentified |

Ce dataset pourrait permettre d'entraîner un modèle à la détection des cellules saines ou cancéreuses. La catégorie ‘Artifact’ comprend 26 images considérées comme des artefacts. Il pourrait être très intéressant, dans un second temps, d'entraîner le modèle dessus afin de diminuer le nombre de faux positifs. Un total de 10127 images non annotées (25 “Unidentified”, 10100 “Unsigned slides”, et 7 “Young Unidentified”) pourrait être classées par le modèle une fois celui-ci entraîné de manière satisfaisante.

TROISIEME SET: LEUKEMIA DATASET (MILAN)

Ce dataset rassemble des images de cancers lymphoblastiques. Ce type de cancer est diagnostiqué par la présence trop importante de lymphocytes. Il est attendu que ce dataset ne comprenne que des images de lymphocytes

Sont disponibles 108 images au format .jpg, associées chacune à un fichier texte au format .xyc reportant les coordonnées des barycentres des blastes. Ces images sont notées XXX_1.jpg ou XXX_0.jpg pour qualifier sur l'image si un patient est malade (1) ou non (0).

Sont également disponibles 260 images au format .tif qui représentent en leur centre un lymphocyte cancéreux ou non. Comme pour les images au format .jpg, les patients sains ou malades sont indiqués par respectivement 0 ou 1 à la fin du nom du fichier.

Ce dataset pourrait permettre d'entraîner le modèle à la détection des lymphocytes sains ou cancéreux.

DATASET COMPLEMENTAIRE : LE DATASET TEST

Le besoin s'est fait ressentir après les premières constructions de modèles de confronter le modèle qui avaient des bons scores, à des données extérieures au data set (Barcelona) utilisé pour l'apprentissage et la validation, afin de tester le degré de généralisation.

Une étude réalisée à Téhéran en 2021 se focalisant sur les leucocytes offre l'avantage de mettre à disposition une quantité significative d'images plus diversifiées et déjà labellisées pour 5 classes sur les 8 utilisées à Barcelone.

Il a été décidé d'utiliser la totalité des images en tant que test data. Pour pallier à l'absence d'images sur les 3 classes absentes, quelques images (3 à 4) jugées caractéristiques (mais toujours non standardisées) ont été manuellement trouvées sur des sites de département d'histologie et ajoutées dans les répertoires (Erythroblast, Immature Granulocytes (IG), et Platelet) afin d'avoir un résultat pour tous les types de cellules.

Seule la taille des images a été retouchée en 360 par 360 pour être comparables avec l'échantillon d'entraînement.

Le dernier paragraphe de la partie data visualisation fournit quelques données descriptives du dataset analysé de manière similaire au dataset d'entraînement.

PERTINENCE

Dans l'optique de mettre au point un modèle performant, il est nécessaire de réaliser chaque étape de pré-processing, de data visualisation, de machine learning et de deep learning de manière conscientieuse. Nous avons donc choisi de nous concentrer sur le data set 1 pour la construction d'un modèle de reconnaissance des cellules sanguines

L'utilisation du data set 2 pour la reconnaissance de cellules malades, n'est envisagé que si le temps de travail restreint par le format bootcamp le permet. Ainsi, le reste du document va être axé sur le dataset 1.

Dans ce dataset, certaines catégories comportent beaucoup plus d'images que d'autres : il y aura sans doute un biais lors de l'apprentissage, à cause de la différence des tailles d'échantillons.

DATA VISUALISATION

Les données des dataset contiennent énormément de variables par le nombre de pixels RGB présents sur chacune des photos. Cependant, grâce aux étapes de pré-processing, il va être possible de ne sélectionner que les pixels importants dans la reconnaissance des cellules.

RESUME

Le premier set de données sur lequel nous avons choisi de travailler est très propre : pas de doublon, pas d'images non annotées, cellules marquées par coloration dans les échantillons avant photographie. Mais les images sont nombreuses et lourdes (360x360x3 pixels en général). De plus, sur certaines images plusieurs cellules, de type différent, sont visibles. Enfin, une image est endommagée et doit être éliminée du set de données.

Quelques étapes de pré-processing sont donc tout de même nécessaires :

1. Lors de la mise en commun des données de travail, nous avons constaté que des problèmes d'upload des données avaient pu générer la création de copies de certaines images. Nous avons donc ajouté à notre code des cellules de nettoyage visant à supprimer les copies « ordinateurs » porteuses d'un numéro entre parenthèses dans le nom de fichier. Cette étape est une sécurité et ne gaspille pas trop de temps de traitement.
2. L'image problématique est contenue dans le dossier “neutrophil”, donc son élimination se fait dans le code à l'aide de la méthode pop grâce à une vérification automatique du type de fichier contenu dans ce dossier spécifique. Une élimination des fichiers non lisibles lors du chargement des images sur tous les dossiers chargés pourra facilement être mise en place en généralisant, lorsque nous voudrons incorporer de nouveaux sets, au cas où le set de

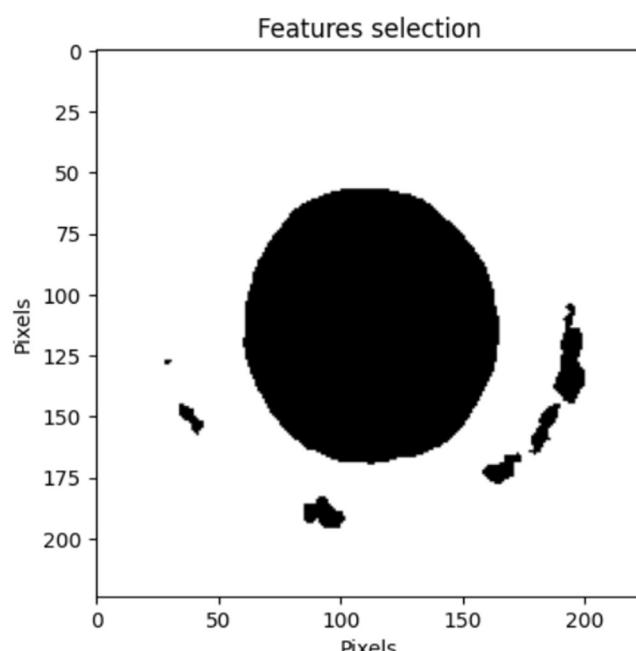
données entrées comporterait plus de fichiers corrompus, mais pour l'instant le code (plus rapide) disponible convient.

3. Des dictionnaires contenant chacun la liste des noms des images ont été créés, de même qu'un dataframe de 17092 lignes contenant 6 colonnes : le nom des images, le code de type cellulaire, la famille cellulaire, la hauteur de l'image, la largeur de l'image, le code de la sous-catégorie de type cellulaire (permettant de redécouper certaines familles de cellules). Ces dictionnaires ont permis une première étude des données.

Ci-dessous une capture d'écran de l'affichage des dernières lignes de ce tableau :

df_barco.tail()						
	image	code_g	cell_type	height	width	code_spe
17087	SNE_995183.jpg	NEU	neutrophil	363	360	SNE
17088	SNE_99568.jpg	NEU	neutrophil	363	360	SNE
17089	SNE_995695.jpg	NEU	neutrophil	363	360	SNE
17090	SNE_995874.jpg	NEU	neutrophil	363	360	SNE
17091	SNE_999519.jpg	NEU	neutrophil	363	360	SNE

4. Un dictionnaire contenant une image moyenne, par catégorie, de toutes les autres, a ensuite été généré. Pour se faire il a fallu redimensionner certaines images qui étaient de taille différente : la méthode resize de cv2 a été utilisée.



```
df_bar["height"] > 363]
```

	image	code_g	cell_type	height	width	code_spe
30	BA_127671.jpg	BA	basophil	369	366	BA
32	BA_128084.jpg	BA	basophil	369	366	BA
70	BA_162483.jpg	BA	basophil	369	366	BA
74	BA_165215.jpg	BA	basophil	369	366	BA
91	BA_178345.jpg	BA	basophil	369	366	BA
...
1678	NEUTROPHIL_941537.jpg	NEU	neutrophil	369	366	NEUTROPHIL
1679	NEUTROPHIL_971896.jpg	NEU	neutrophil	369	366	NEUTROPHIL
1680	NEUTROPHIL_982898.jpg	NEU	neutrophil	369	366	NEUTROPHIL
1681	NEUTROPHIL_985581.jpg	NEU	neutrophil	369	366	NEUTROPHIL
1682	NEUTROPHIL_993818.jpg	NEU	neutrophil	369	366	NEUTROPHIL

250 rows × 6 columns

5. Une première étape de seuillage sur des images type de chaque catégorie a permis de mesurer la taille moyenne de la surface projetée des cellules, en pixel, pour chaque type cellulaire.
6. Une analyse en composantes principales a été réalisée sur l'ensemble des images du dataset afin de générer un fichier de travail plus léger contenant des features importantes pour les étapes d'entraînement (réduction des données). Des dataframes au format csv ont été créés pour chacune des catégories d'image.

VISUALISATIONS ET STATISTIQUES

Le dataset Mendeley_Barcelonat contient 8 classes de cellules saines:

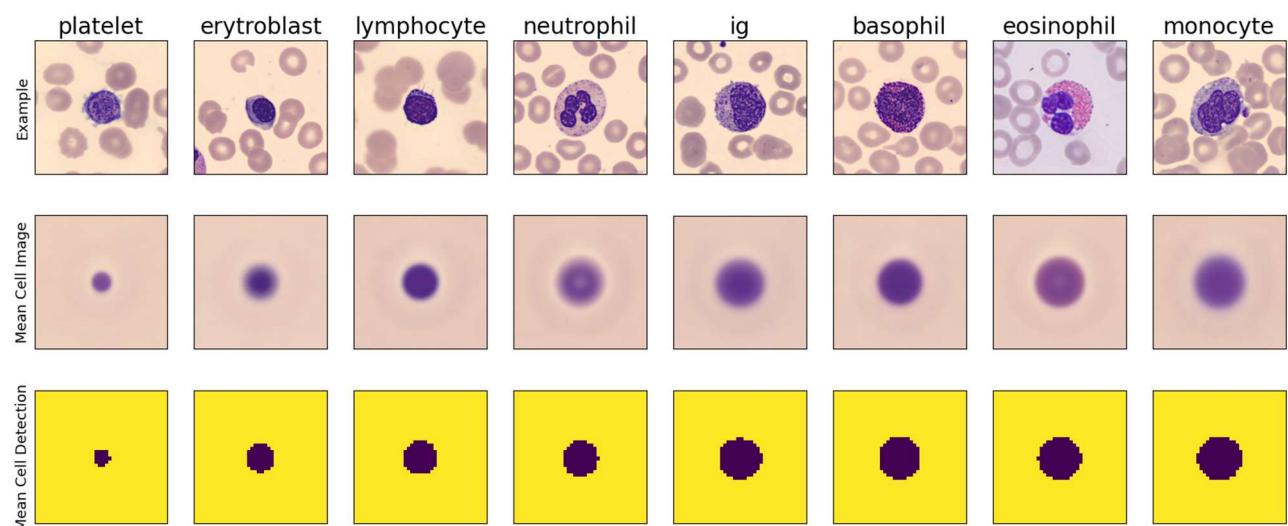
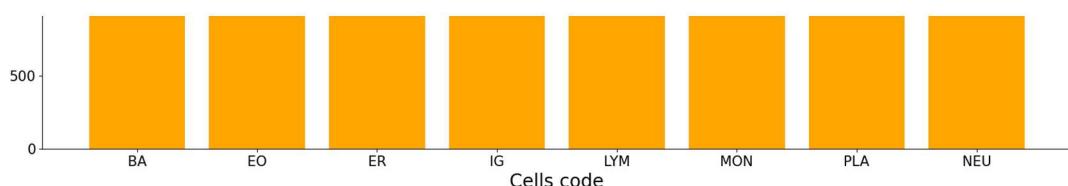
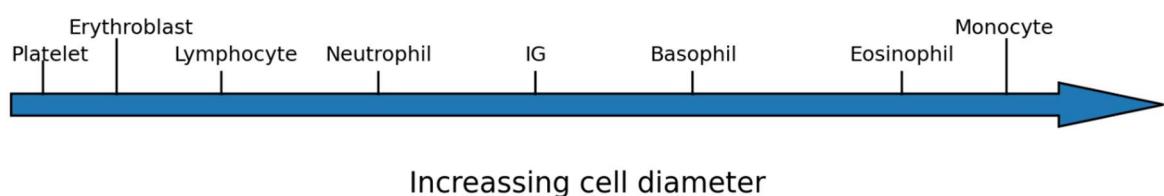
- Basophile (BA),
- Eosinophile (EO),
- Erythroblast (ER),
- Immature Granulocyte (IG),
- Lymphocyte (LM),
- Monocyte (MON),
- Plaquette (PLA)
- Neutrophile (NEU).

Au total, 17092 images de cellules sont réparties comme suit : 3330 NEU, 1823 EO, 2890 IG, 2348 PLA, 1551 ER, 1420 MON, 1218 BA et 1214 LYM.

En moyenne, il y a par classes 1974 images avec un écart-type de 747.

Comme dit précédemment, il y a un déséquilibre des classes. Le graphique ci-dessous illustre ce phénomène.

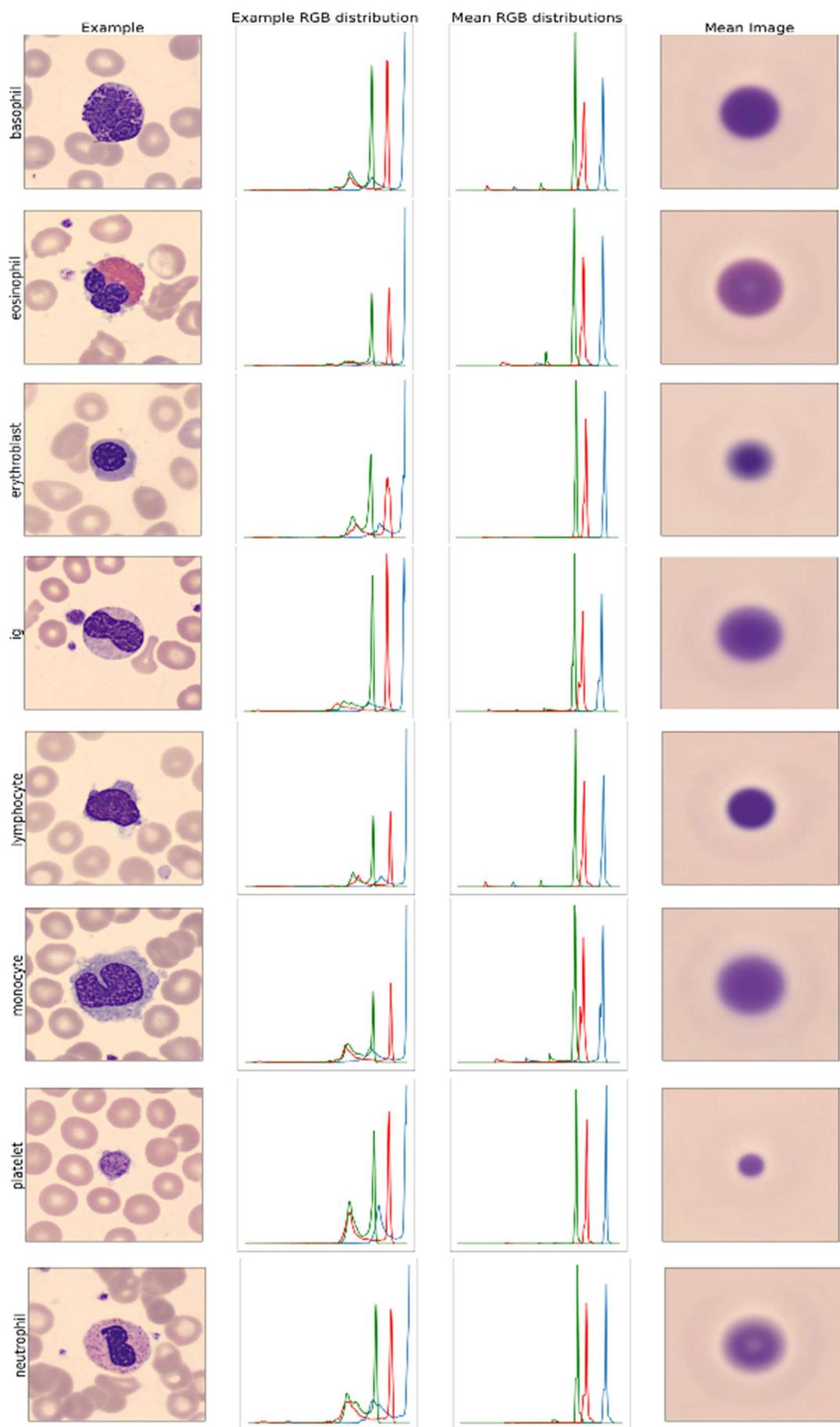
Concernant les cellules, selon le type cellulaire nous allons observer des formes et des tailles de cellules et noyaux différents. A partir des images moyennées, il a été calculé la taille des cellules en fonction du nombre de pixels. Ci-dessous, sont présentes trois figures, la première représente une cellule de chaque catégorie prise au hasard, la seconde représente les cellules "moyennées" et la dernière est une frise qui classe les cellules en fonction de leur taille.



Enfin, il a aussi été réalisé une analyse sur la répartition RGB des images, premièrement sur les images brutes, en prenant une cellule par type, puis sur les images “moyennées”.

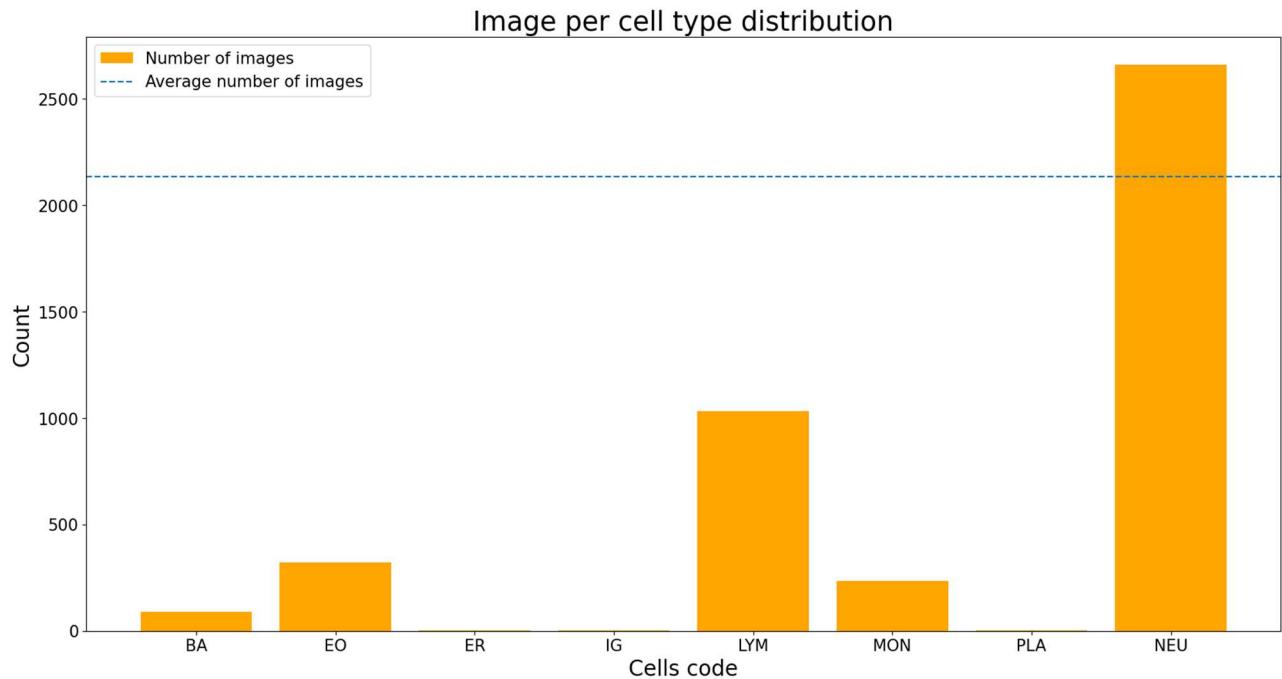
A partir de ces graphiques on peut voir que selon le type cellulaire il semble exister une répartition différente des couleurs RGB en termes de hauteurs de pics (notamment en ce qui concerne le ratio entre vert et bleu), et de distance entre les pics, avec parfois des sous-pics au sein d'une même couleur. Sur certains graphiques, notamment sur ceux des IG et éosinophiles, on peut voir qu'il y a une sorte de bruit de fond qui pourrait être dû au fond de chaque image.

Des différences subsistent sur les images moyennées entre certains types cellulaires

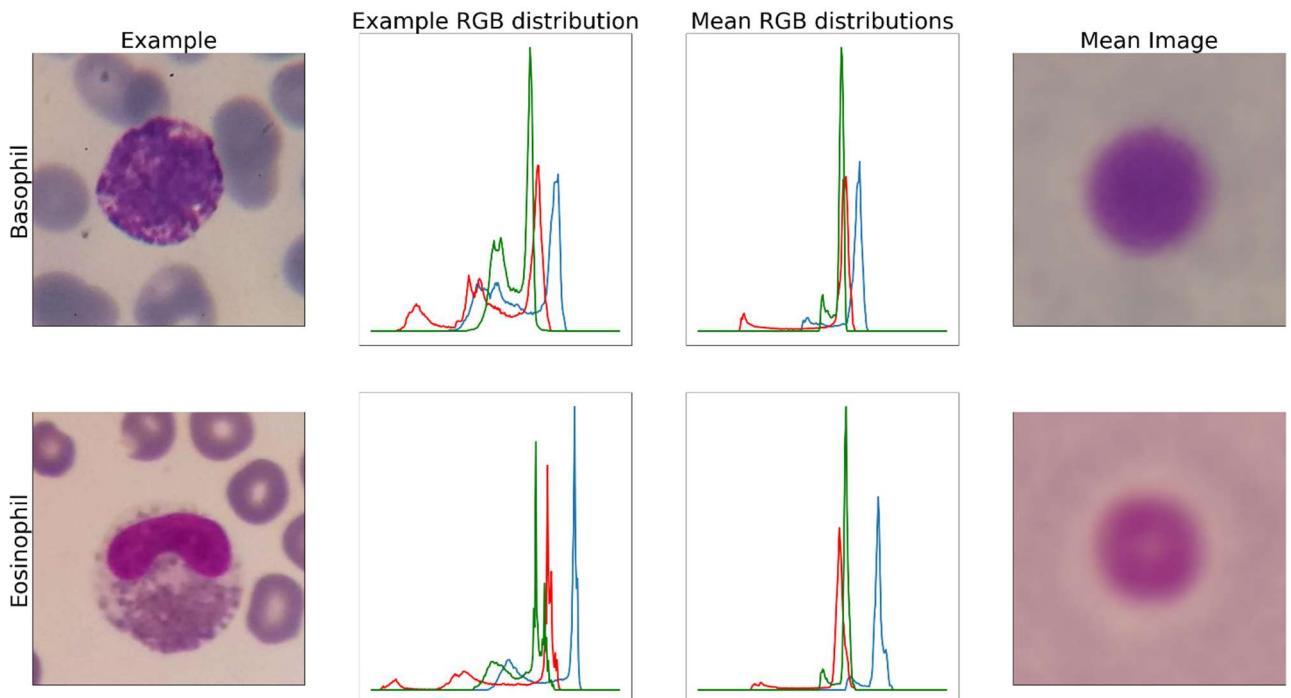


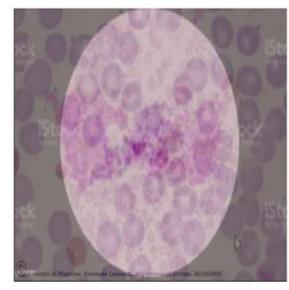
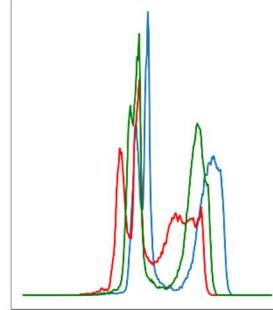
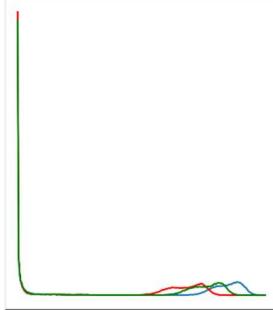
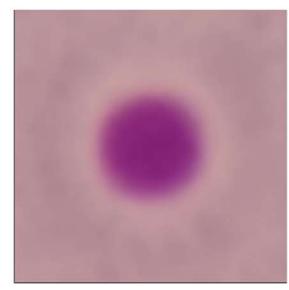
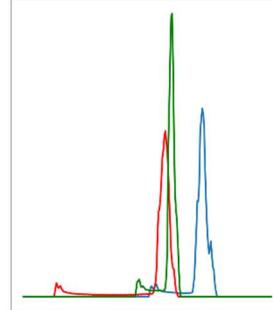
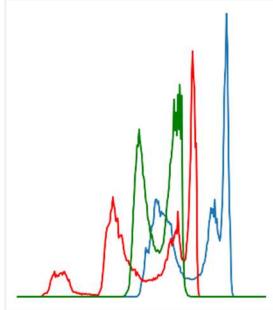
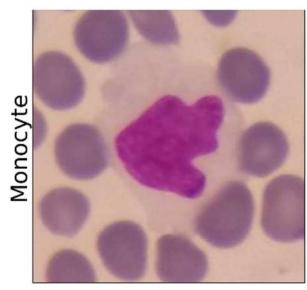
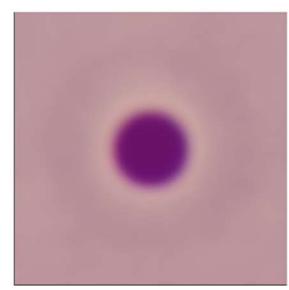
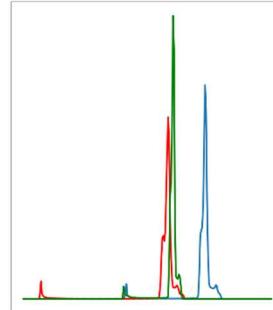
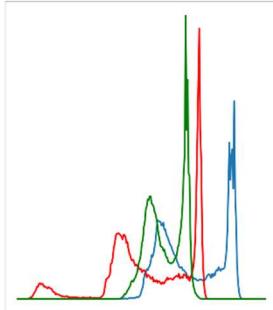
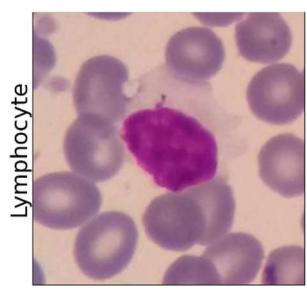
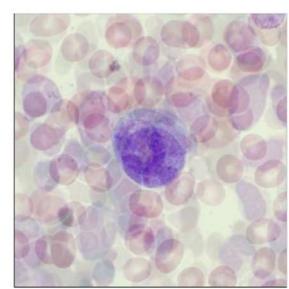
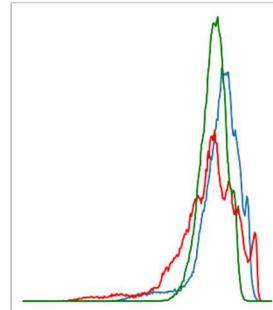
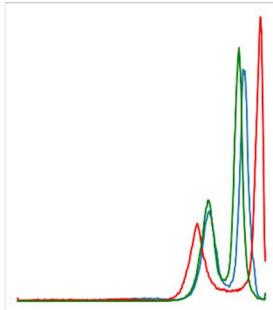
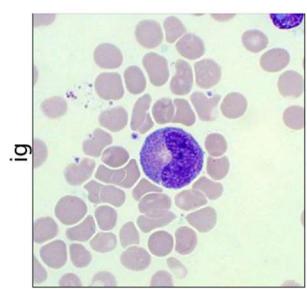
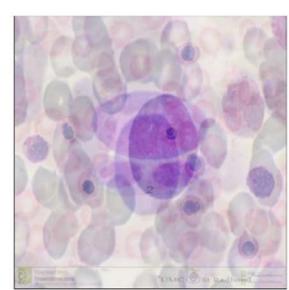
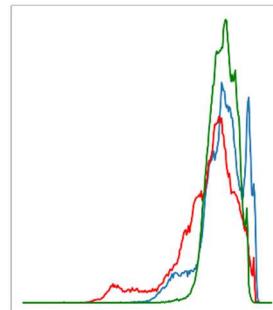
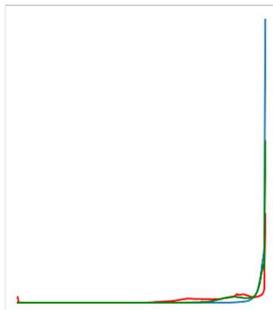
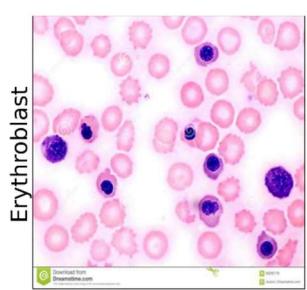
Concernant le dataset de test qui nous servira pour les modèles de Deep Learning.

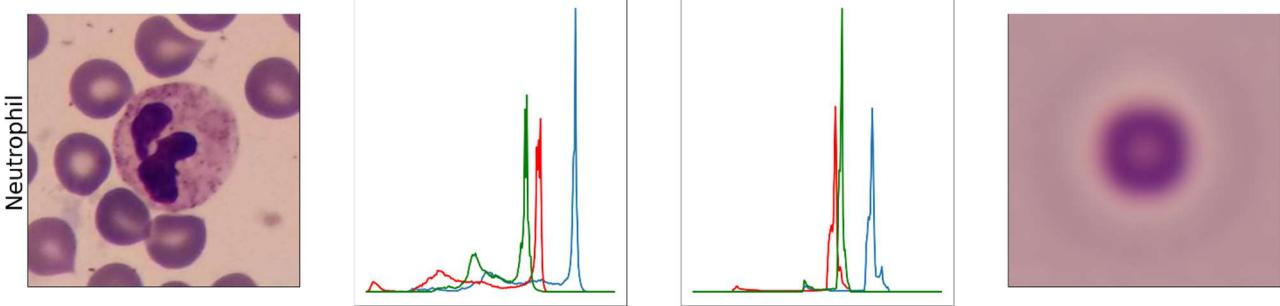
On y retrouve des photos de 1034 lymphocytes, 234 monocytes, 322 éosinophiles, 4 érythroblastes, 4 IG, 89 basophiles, 3 plaquettes et enfin 2660 neutrophiles. Il y a un important déséquilibre des classes mais ceci ne devrait pas poser problème étant donné que ce dataset ne servira pas à l'entraînement des modèles.



Pour chacune des classes, image et distribution RGB de respectivement une image prise au hasard et la moyenne des images.







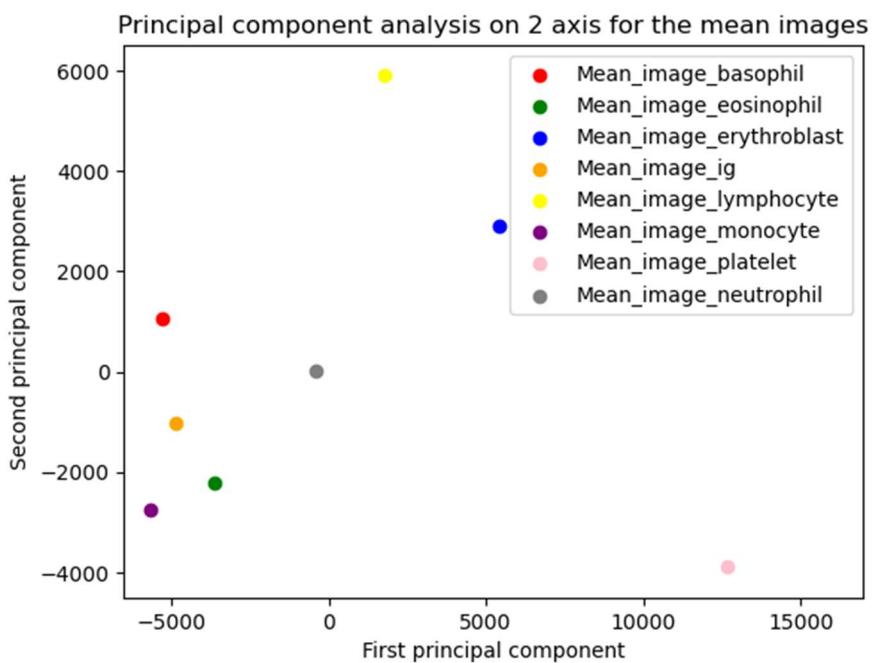
On voit bien ressortir sur cette figure les trois classes de cellules pour lesquelles seul quelques images ont été rajoutées à la main dans le dataset. L'effet de moyenne est insuffisant pour faire ressortir l'image type des cellules, tout comme des pics bien dessinés.

PREPROCESSING

Les 17092 images au format .jpg de la base de données utilisée pour la modélisation prennent beaucoup de temps à charger à chaque utilisation car composées chacune en moyenne de 360 par 363 pixels, sur 3 canaux RGB.

La PCA permet de faire une réduction intelligente des données, qui permettrait d'améliorer le temps de calcul pour la partie Machine Learning.

Dans un premier temps une PCA est réalisée sur les images moyennes obtenues lors de l'étape de visualisation des données. Le résultat pour les 2 composantes principales est présenté sur le graphique ci-dessous :



On constate qu'on a bien une discrimination entre les différentes cellules, nous allons donc utiliser cette stratégie pour réduire la taille de nos données tout en conservant leurs caractéristiques singulières.

Une transformation des données combinée entre une réduction non intelligente et une PCA est donc réalisée pour générer les fichiers de travail de l'étape suivante, le Machine Learning.

Tout d'abord, les images sont chargées une à une en vérifiant l'intégrité du fichier, avec les modifications suivantes (réduction directe) :

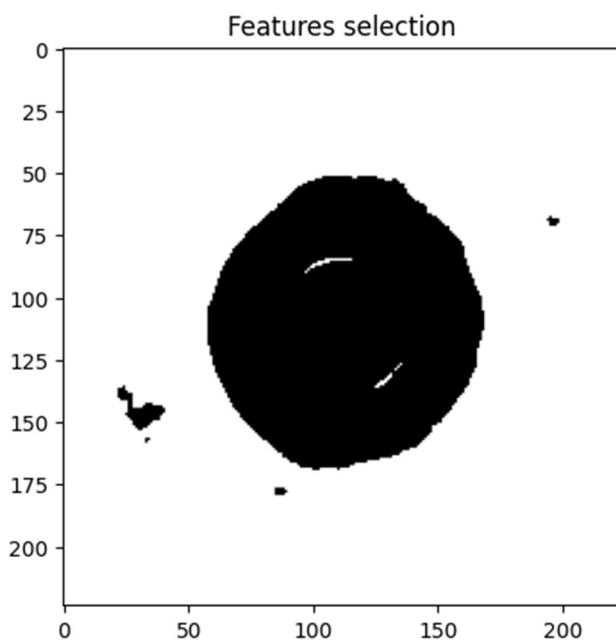
- somme des pixels sur les canaux de couleur afin d'obtenir des images en noir et blanc codées dans des tableaux à deux dimensions et non plus trois.
- réduction de la taille d'environ 40% avec l'utilisation de la fonction resize pour charger les images au format 224x224 pixels (soit 50176 features par image contre 392040 initialement).

2 étapes de réduction intelligente sont ensuite réalisées :

- la sélection des 20% de pixels les plus porteurs d'information aux yeux de l'algorithme SelectPercentile de scikit learn.

On obtient alors des « images » de 10035 features chacune (soit une réduction de 97,4%) par rapport à l'image initiale.

(Pixels conservés en noir sur l'image ci-dessous)



Cette première phase est particulièrement intéressante car elle permet de diminuer un biais des images que nous avions détecté lors de l'étape de l'exploration des données : les cellules, sur ce jeu de données, sont toujours centrées. Cette caractéristique pourrait facilement être reconnue et apprise par un modèle. En sélectionnant seulement les pixels composant la cellule en préprocessing, nous augmentons les chances que le modèle puisse reconnaître des cellules non centrées sur une image future.

- la sélection d'une partie des composantes principales sur les images pré-réduites. Cette étape est réalisée 3 fois, en conservant respectivement 70, 80, et 90% de l'information.

Dans la partie suivante, Machine Learning, nous ferons référence au jeu de données par le terme df70, df80 ou df90 selon le set réduit auquel on se réfère.

Les nouvelles « images » ainsi obtenues, de respectivement 22, 60, et 236 features, sont sauvegardées dans des fichiers .csv afin d'être utilisées par la suite pour le Machine Learning. On a donc finalement conservé environ 0,02, 0,05, et 0,18 % du nombre de features initial (et 0,04, 0,12 et 0,47% par rapport aux images après la réduction directe). La dernière colonne de chaque fichier .csv contient le label de l'image.

La réduction réalisée est très forte. Elle nous permet certes d'accélérer le temps de calcul, mais risque d'entraîner une perte de performance du modèle.

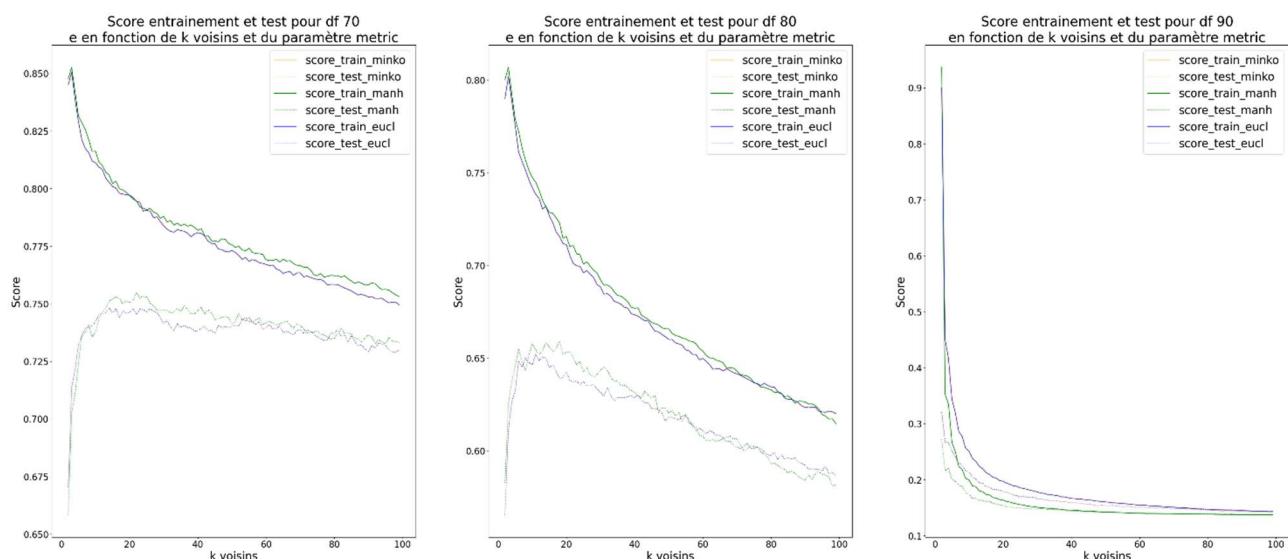
MACHINE LEARNING (ML)

Différents modèles d'apprentissage supervisés sont testés sur les données réduites. Notre problématique est de type classification nominale.

Avant de débuter l'entraînement des modèles testés, une étape de normalisation des données a été effectuée par l'utilisation de la fonction MinMax de la librairie scikit learn.

METHODE DES K PLUS PROCHES VOISINS, KNN

La méthode KNN est implémentée sur les 3 set de données, pca70, pca80, pca90, avec un nombre de voisins variant de 2 à 99, et deux métriques de calcul des distances entre points : « minkowsky » (noté minko par la suite), qui utilise le mode de calcul euclidien par défaut (une comparaison avec le mode « euclidean » montre que l'on obtient bien le même résultat), et « manhattan » (noté manh par la suite).



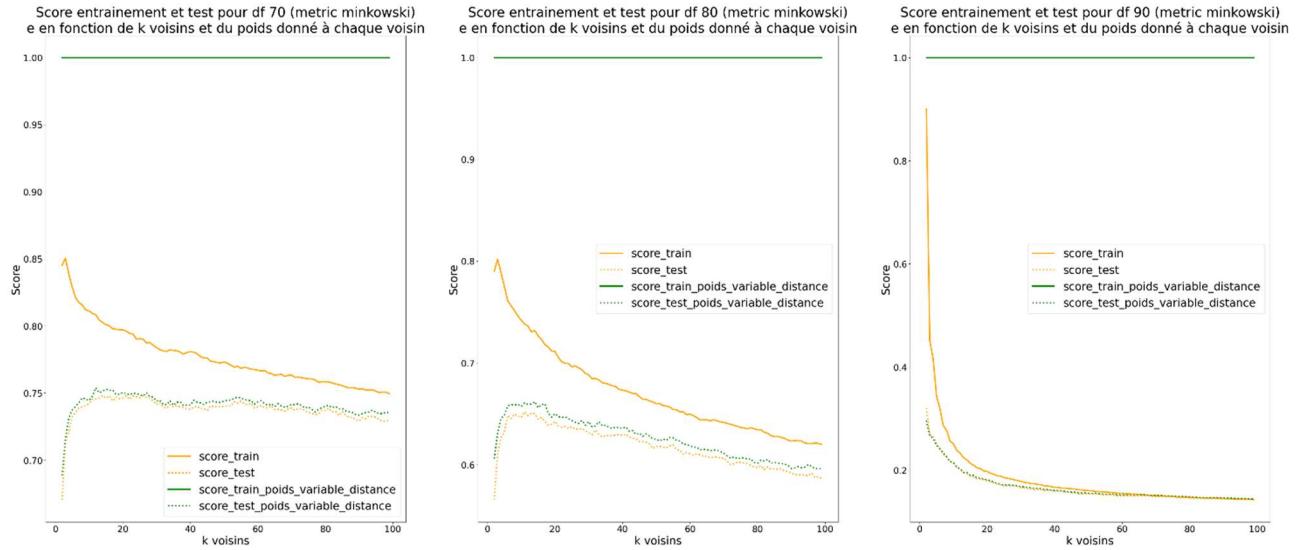
Le premier graphique ci-dessus illustre comme attendu le fait que :

- la métrique « minkowsky » est bien euclidienne (courbes exactement superposées)
- le score de précision est systématiquement inférieur pour le test par rapport à l'entraînement.

On constate que le mode de calcul euclidien donne des résultats meilleurs que manh avec un maximum de précision (test) à 0,748, 0,65, et 0,32 pour df70, df80 et df90 respectivement. Les résultats pour df90 sont nettement inférieurs à ceux obtenus avec les deux autres réductions. Il semblerait que KNN ne soit pas capable de fonctionner avec autant de features (pourtant seulement 236), ou bien que parmi les 236 features restantes, du bruit ait été sélectionné.

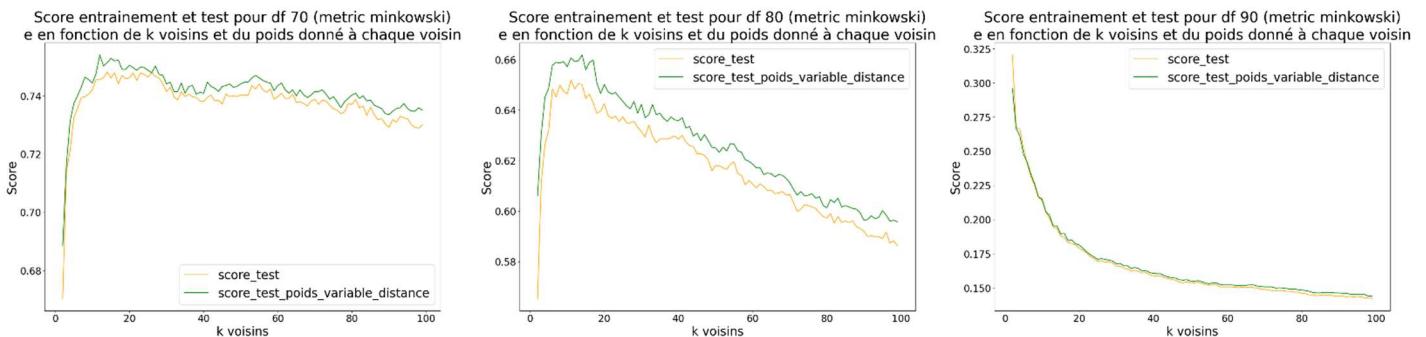
On se rappelle que les différentes classes ne sont pas représentées de manière égale dans les données, ce qui pourrait entraîner un biais dans la reconnaissance. Pour voir dans quelle mesure

ce paramètre joue, nous avons réentraîné le modèle avec la métrique minko, toujours avec un nombre de voisins variant de 2 à 99, en ajoutant un poids variable lié à la distance à chaque voisin. Plus un voisin est loin du point étudié, moins son influence est importante sur la décision.



On observe, sur les courbes ci-dessus, un overfitting dans le cas d'un entraînement où le poids des voisins est pondéré par leur distance au point recherché. C'est logique puisque lors de l'entraînement le point "recherché" appartient aux données d'entraînement donc il va lui même être le plus proche et être doté du plus grand poids. Sur le jeu de données test, en revanche, on voit que l'on a bien une amélioration du résultat, même si elle n'est pas extraordinaire.

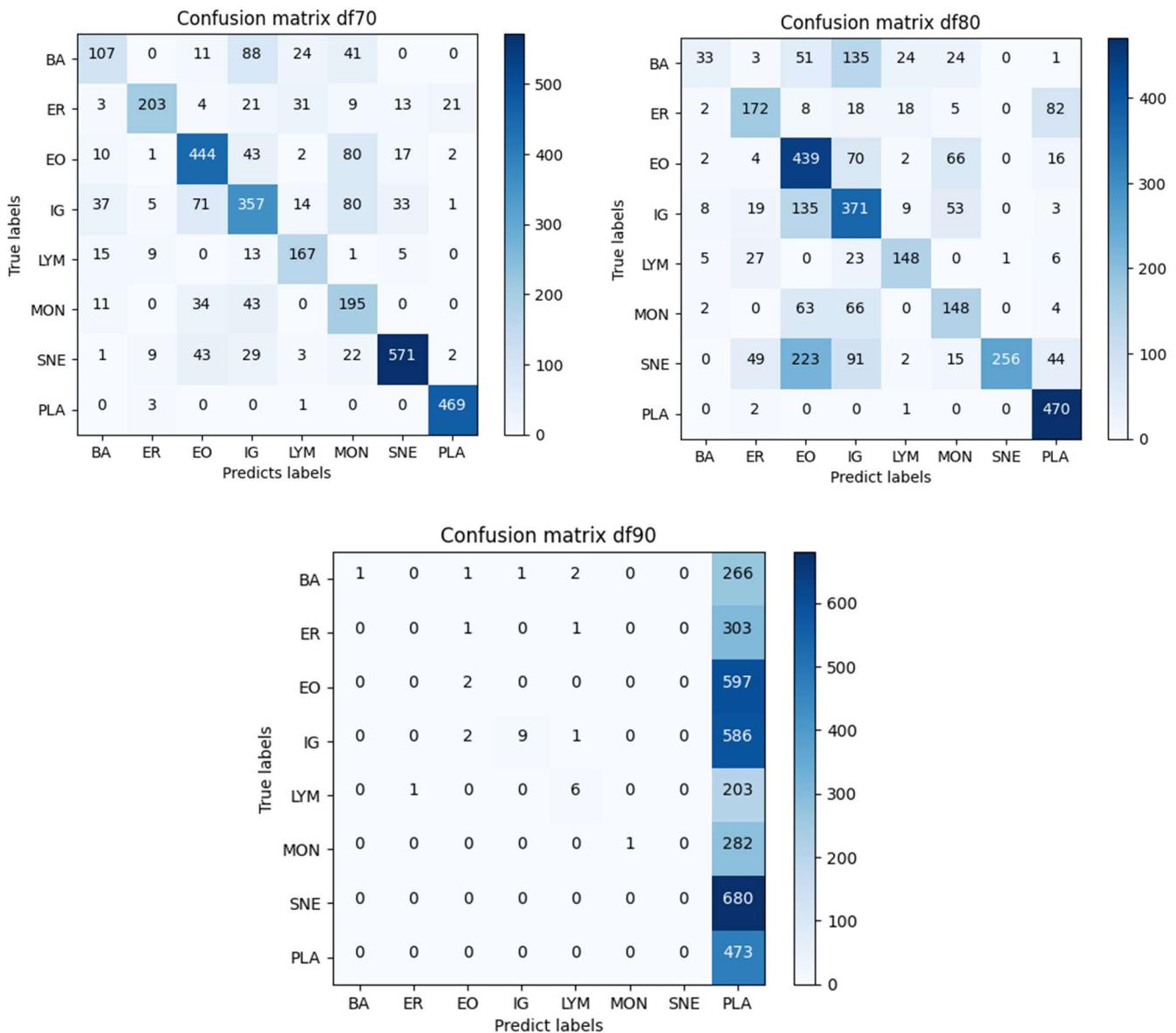
Pour plus de lisibilité, seules les courbes de score sur les données test sont présentées ci-dessous :



L'utilisation de poids permet bien d'améliorer les résultats, mais très faiblement, excepté pour df90. En effet le score de précision maximal (test) obtenu pour df70, df80, et df90 devient 0,75, 0,66 et 0,29 pour un nombre de voisin k égale à 12, 14 et 2, respectivement.

On a toujours un score un peu meilleur pour les données les plus réduites (possibilité d'une élimination de « bruit » par la réduction ?), mais la précision reste très similaire, ce qui est logique étant donnée la faible différence relative des set df70, 80 et 90 entre-eux par rapport à la différence avec le set d'origine.

Les KNN avec les poids ayant globalement de meilleurs résultats, des matrices de confusion ont été générées à partir de ces derniers. Voici les résultats obtenus pour chaque réduction :



Tout d'abord on peut constater que le modèle n'arrive pas à prédire avec le df90, tout est classé comme étant des plaquettes. Concernant les deux autres matrices, le label le mieux prédit est celui des plaquettes. Ceci peut s'expliquer par la grande différence entre la taille des plaquettes (très inférieure) et celles des autres cellules, ce qui les rend facilement identifiables. Cette observation est confirmée grâce au tableau de classification_report ci-dessous. Pour les plaquettes (classe 8), on obtient des précisions de 0.95 et 0.75 et un rappel de 0.99.

Correspondance des codes et numéros de classe pour la pca :

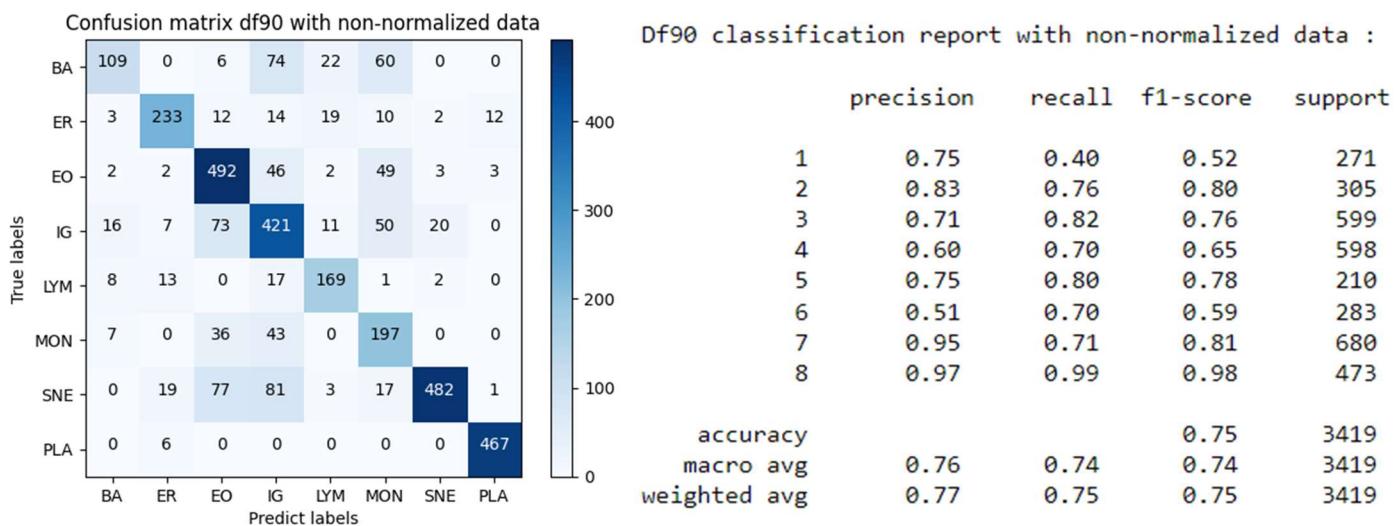
- BA : 1
- ER : 2

- EO : 3
- IG : 4
- LYM : 5
- MON : 6
- SNE : 7
- PLA : 8

Df70 classification report :					Df80 classification report :				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.58	0.39	0.47	271	1	0.63	0.12	0.20	271
2	0.88	0.67	0.76	305	2	0.62	0.56	0.59	305
3	0.73	0.74	0.74	599	3	0.48	0.73	0.58	599
4	0.60	0.60	0.60	598	4	0.48	0.62	0.54	598
5	0.69	0.80	0.74	210	5	0.73	0.70	0.71	210
6	0.46	0.69	0.55	283	6	0.48	0.52	0.50	283
7	0.89	0.84	0.87	680	7	1.00	0.38	0.55	680
8	0.95	0.99	0.97	473	8	0.75	0.99	0.86	473
accuracy			0.74	3419	accuracy			0.60	3419
macro avg	0.72	0.71	0.71	3419	macro avg	0.65	0.58	0.57	3419
weighted avg	0.75	0.74	0.74	3419	weighted avg	0.66	0.60	0.58	3419

Les basophiles sont les cellules les moins bien classées avec un rappel de 0.39 et 0.12 pour respectivement df70 et df80. Il semblerait qu'ils soient souvent classés comme étant des IG. Cela peut s'expliquer par le fait que les basophiles et les IG ont des tailles proches, avec d'ordinaire une taille plus faible pour les IG. Globalement, les cellules sont mieux différencierées et classées dans le df70 que le df80.

Le KNN a également été effectué sur données non normalisées. Comparé aux KNN précédent on obtient des scores de précision supérieur avec des précisions entre 0,78 et 0,76. En particulier, on constate que l'algorithme fonctionne sur les données du df90 et qu'on n'observe pas l'effondrement de précision que l'on a pu observer avec les données normalisées.



Les métriques s'améliorent grandement dans le cas des données non normalisées par rapport aux données normalisées. La normalisation ici semble masquer de manière importante des informations.

Par ailleurs, on observe également qu'une diminution de l'information lorsque l'on augmente le nombre de features (résultats meilleurs sur df70 que sur df80), le modèle est moins capable de différencier les cellules.

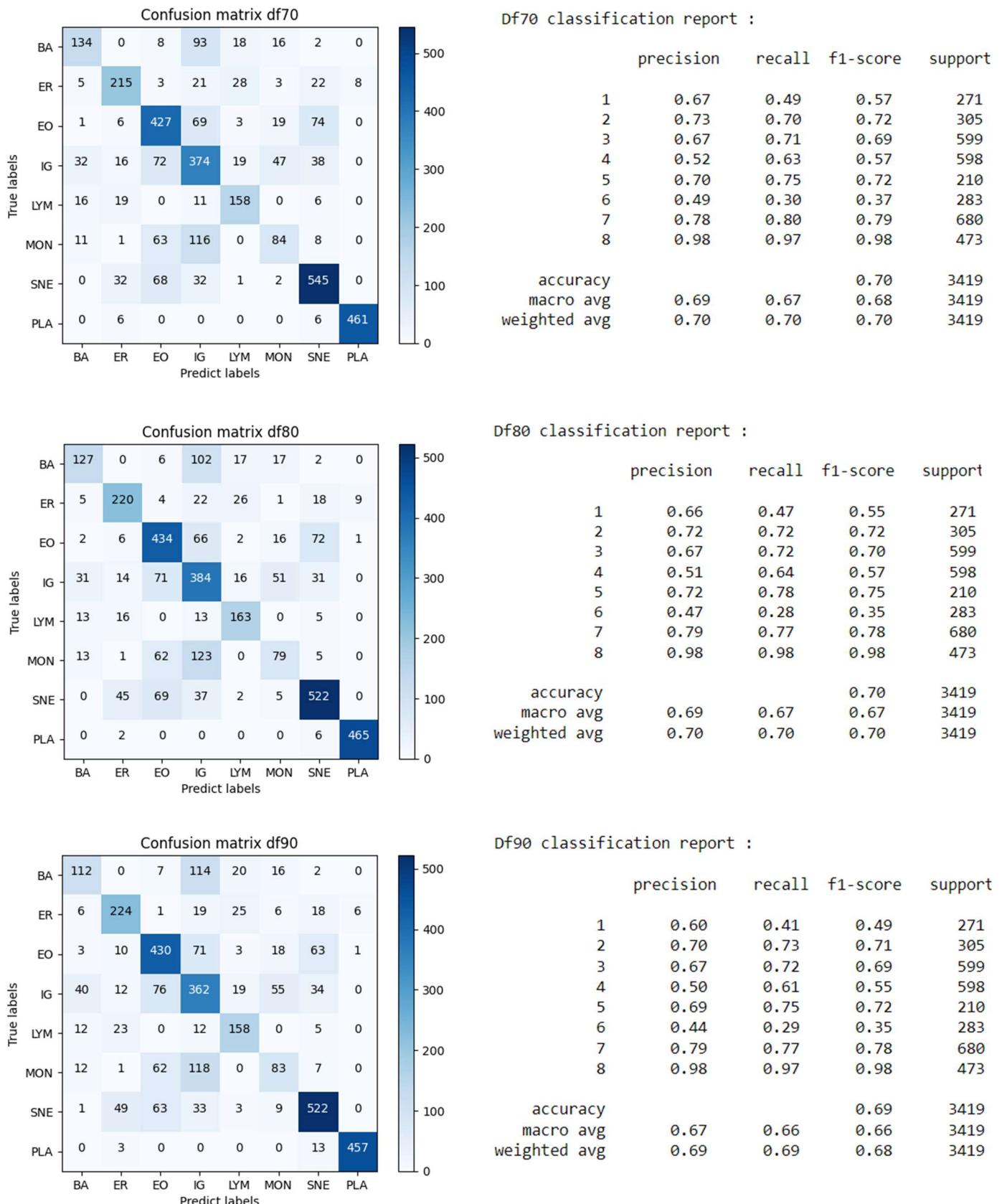
REGRESSION LOGISTIQUE

Afin d'optimiser l'utilisation du modèle LogisticRegression de scikit learn, l'outil GridSearchCV est mis en place en amont, pour sélectionner les paramètres du classifier.

Le modèle est ainsi entraîné et comparé sur les 3 jeux de données df70, df80, et df90,

- pour deux algorithmes d'optimisation,
 - « liblinear » qui travaille en comparaison « un contre tous » mais est optimal pour les petits dataset
 - « lbfgs » qui gère les fonctions de perte de type cross-entropy « un contre un » (paramètre « multi_class » en mode « auto » qui bascule sur « multiclass » pour « lbfgs »), avec une régularisation variable (C),
- avec 2 paramètres de découpe pour la validation croisée de GridSearch (cv) : 3 et 15.

Les résultats donnent des valeurs de précision absolument identiques entre cv=3 et cv=15. Elles sont comprises entre 0,69 et 0,70, avec les meilleures obtenues pour df80 et df70 avec les paramètres solver = 'lbfgs' et C=100 . Ci-dessous sont présentés les matrices de confusion et les classification_report obtenus pour chacun des dataframes :



On peut voir que comme avec le KNN, les basophiles ne sont pas très bien classés et sont, là aussi, souvent confondus avec des IG. Cependant, comparé au KNN, la régression linéaire n'arrive pas très bien à classer les monocytes avec des rappels de 0.28 et 0.29 pour les df80 et df90. Ils sont majoritairement classés comme étant des IG et dans une moindre mesure comme des éosinophiles. Les éosinophiles sont les cellules qui sont les plus proches en tailles des monocytes. Entre les IG

et les monocytes, il semble y avoir des similitudes notamment concernant la proportion entre la taille de la cellule et celle du noyau ainsi que le fait que des granulosités sont présentes dans les deux.

COMBINAISON AVEC VOTINGCLASSIFIER

Les deux modèles précédemment utilisés donnent des résultats assez moyens, avec une précision qui n'atteint jamais 0,8. Nous avons tenté de les combiner pour voir si les forces de chacun des deux modèles pouvaient se recouper afin d'améliorer le résultat final.

Les deux modèles KNN et régression logistique sont entraînés avec les hyperparamètres optimaux obtenus sur le df70 et combinés avec 3 ratios de poids différents et une classe de vote « soft ».

Le tableau ci-dessous présente les performances obtenues ensuite avec un GridSearch (cv=15) :

	params	mean_test_score	std_test_score	rank_test_score
0	{'weights': [0.8, 0.2]}	0.772545	0.013069	1
1	{'weights': [0.5, 0.5]}	0.772326	0.012499	2
2	{'weights': [0.3, 0.7]}	0.750677	0.011228	3

La meilleure précision est de 0,77 et est obtenue avec le plus grand poids donné au modèle KNN. Or, dans nos tests précédents, ce modèle KNN obtenait une précision de 0,78. La combinaison des modèles n'a donc pas permis, dans ce cas-ci, d'améliorer le score.

Concernant la détection des cellules, les basophiles sont encore une fois les cellules les moins bien classées avec un rappel de 0,47. Les plaquettes sont les cellules les mieux détectées et classées avec un f1-score de 0,98.

Vating classifier classification report :

	precision	recall	f1-score	support
1	0.62	0.47	0.53	271
2	0.87	0.72	0.79	305
3	0.74	0.76	0.75	599
4	0.60	0.61	0.61	598
5	0.69	0.77	0.73	210
6	0.49	0.64	0.56	283
7	0.87	0.85	0.86	680
8	0.97	0.99	0.98	473
accuracy			0.75	3419
macro avg	0.73	0.73	0.73	3419
weighted avg	0.75	0.75	0.75	3419

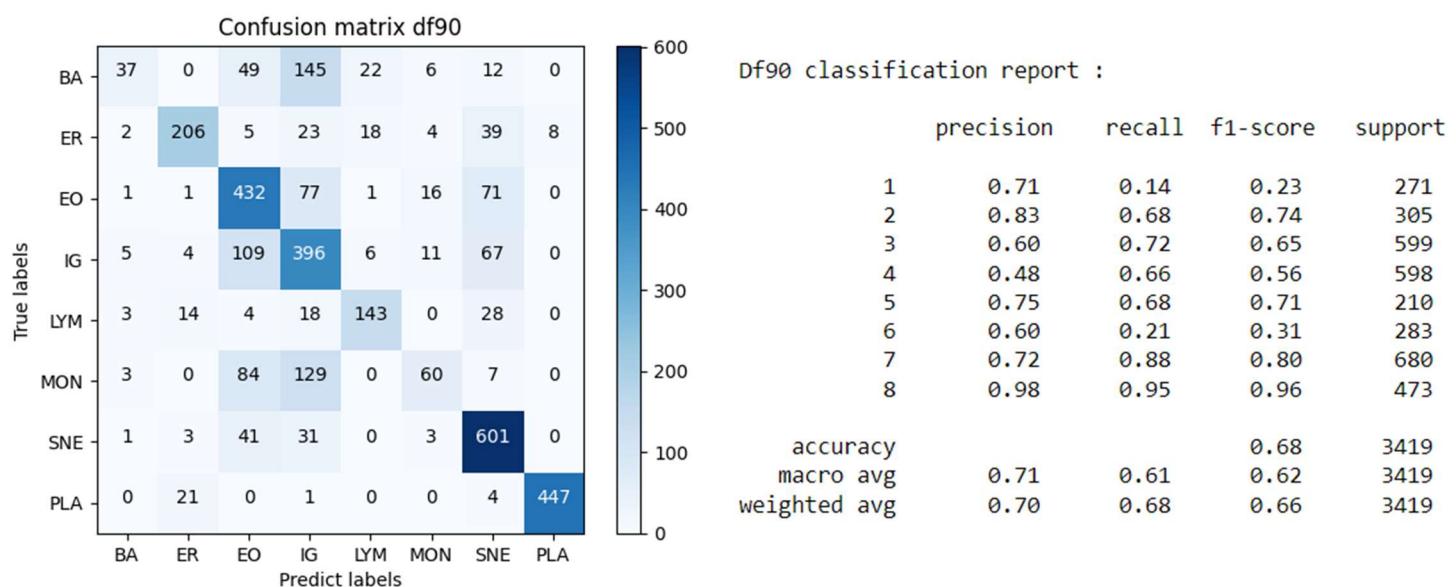
RANDOM FOREST (FORETS ALEATOIRES)

Les forêts aléatoires sont des modèles d'ensemble, qui semblent donc particulièrement appropriés à notre jeu de données.

Les performances de précision sur les données test donnent, pour df70, df80 et df90, respectivement : 0,76, 0,75 et 0,68.

A nouveau, on observe une performance plus faible pour les données moins réduites.

Comme avec les modèles précédents la classe la moins bien classées est celle des basophiles avec des rappels à 0,47, 0,38, et seulement 0,14 pour respectivement df70, df80 et df90. La seconde classe la moins bien classées est encore une fois les monocytes avec un premier rappel à 0,48 pour df70 et ça descend à 0,21 pour df90.



Une recherche des paramètres optimaux pour le modèle random forest a été effectuée, sur différentes valeurs d'estimateur, de profondeur pour l'arbre et sur deux types de criterion : gini et entropy.

CONCLUSION

En conclusion, les résultats obtenus à ce stade avec les modèles de Machine Learning ne sont pas complètement satisfaisants, même si la précision est bien au-dessus du score aléatoire (1/8).

La classe des basophiles est toujours la classe la moins bien détectée et classée quelle que soit la réduction appliquée. Cependant, plus la réduction diminue moins cette classe est bien catégorisée et on voit apparaître un autre type cellulaire mal classés, les monocytes.

Cela pourrait être dû :

- à des limitations sur les modèles :
 - o comme un choix inappropriate de modèle, mais comme plusieurs ont été testés, cette option paraît peu probable;
 - o une optimisation « manuelle » insuffisante, mais à nouveau, plusieurs outils de sélections ont été utilisés
- à des limitation des données comme
 - o un déséquilibre du jeu de donnée, mais ce paramètre a été testé et ne semble pas avoir beaucoup d'influence dans notre cas,
 - o un nombre insuffisant d'images,
 - o ou encore une réduction trop importante des données causant une perte d'information, mais les résultats plus faibles pour le jeu de données moins réduit semblent contredire cette idée.
- enfin, aux caractéristiques des cellules elles-mêmes. Certaines cellules présentent des caractéristiques similaires. C'est le cas par exemple des basophiles et des IG qui sont relativement proches en termes de taille. Les monocytes sont proches en tailles des éosinophiles et leur noyau présente des granulosités tout comme les IG. Les neutrophiles sont également similaires aux IG par leur noyau qui semble présenter 3 petits nodules. De plus la proportion de cytoplasme dans les deux types cellulaires semble proche. Ces similitudes dans les cellules se retrouvent dans les résultats des modèles. Moins la réduction est forte plus les basophiles sont classés comme étant des IG, quel que soit le modèle. Les monocytes sont classés comme étant des IG ou des éosinophiles, notamment avec les modèles de Random Forest et de Régression Linéaire. Les neutrophiles sont classés comme des éosinophiles avec le modèle KNN dès df80.

Passer sur des modèles autonomes de Deep Learning pourra sans doute améliorer les performances.

DEEP LEARNING (DL)

Le deep learning est un domaine du machine learning. Il est basé sur des réseaux de neurones artificiels afin de gérer de grandes quantités de données en ajoutant des couches au réseau. De manière simplifiée, il s'agit du machine learning fonctionnant grâce à des réseaux de neurones profonds. Dans un premier temps, nous allons repartir des données brutes pour entraîner les modèles et nous comparerons ensuite avec les résultats sur données réduites.

RESEAUX DE NEURONES CONVOLUTIFS (CNN)

Nous avons entraîné plusieurs réseaux de neurones convolutifs CNN à l'aide du module Keras avec Python dans le cadre d'une classification d'image de cellules de sang. Pour cela, nous avons utilisé

un ensemble d'images composé de 17092 images. Ces images ont ensuite été divisées en un ensemble d'entraînement et de validation afin d'alimenter le réseau de neurones convolutifs. Les étapes de préprocessing du dataset ont été réalisées grâce à Tensorflow. Les images sont dimensionnées au format 360*360 pixels.

Parmi les grandes familles des réseaux de neurones profonds, nous trouvons les réseaux de neurones convolutifs appelés CNN. Ils sont axés sur des applications comportant des motifs répétitifs dans différents domaines de l'espace de modélisation, notamment dans la reconnaissance d'image et vidéo, les systèmes de recommandation, et le traitement du langage naturel.

Les CNN sont composés de couches convolutionnelles et de couches de pooling peuvent être ajustées en utilisant des paramètres.

- *Couche convolutive*

Les couches convolutives consistent à appliquer un filtre de convolution à l'image pour détecter des caractéristiques de l'image. Une image passe à travers une succession de filtres créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Les filtres de convolution sont mis à plat et concaténées en un vecteur de caractéristiques. Les filtres passent à travers une fonction d'activation non linéaire (le plus souvent rectified Linear unit (relu) qui consiste à remplacer les nombres négatifs des images filtrées par des zéros).

- Couche de pooling

Elle consiste à réduire progressivement la taille de l'image en ne gardant que les informations les plus importantes.

Avec la couche pooling, la quantité de paramètres et de calcul dans le réseau sont réduits, et cela va permettre de contrôler le sur-apprentissage.

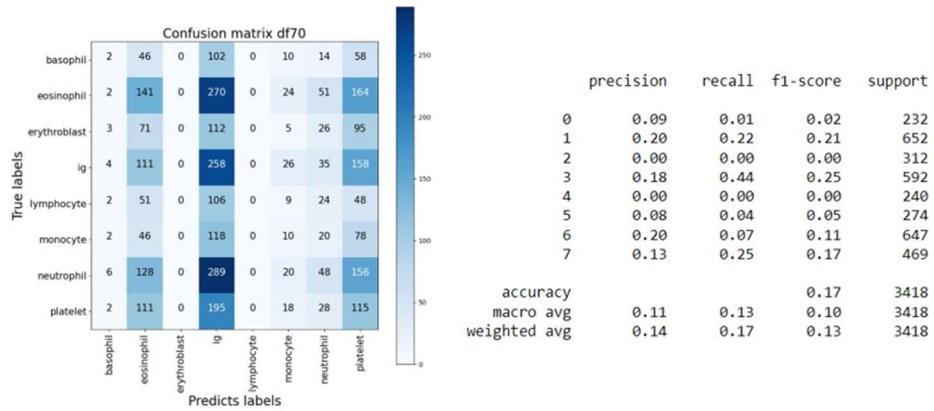
ARCHITECTURE LENET

Ce premier CNN minimaliste est composé de 5 couches :

- Une couche de convolution avec la fonction d'activation 'relu';
- Une couche de maxpooling ;
- Un dropout ;
- Un flatten ;
- Deux couches denses : une avec la fonction d'activation Relu, l'autre avec la fonction d'activation 'softmax'.

Le modèle est compilé avec la fonction de perte 'sparse categorical crossentropy', l'optimizer adam et l'accuracy comme métrique. L'entraînement du modèle a été effectué sur 5 epochs.

La matrice de confusion et le classification report sont présentés ci-dessous.



La mesure globale de la performance du modèle affiche un résultat de 0.17. Les cellules sont prédictes pour les classes eosinophiles, Ig et neutrophiles.

CNN 1

Ce CNN combine trois paires de convolution et maxpooling suivi d'un flatten, une couche dense, un dropout et une dernière couche dense. La fonction d'activation des couches de convolution et de la première couche dense est 'ReLU'. La fonction d'activation de la dernière couche dense est 'softmax'.

Comme précédemment le modèle a été combiné avec la fonction de perte 'sparse categorical crossentropy', l'optimiseur 'adam' et la métrique 'accuracy'. Ici, le modèle a été entraîné sur 15 epochs.

De plus, le modèle a été entraîné sur plus d'images. En effet, il a été appliqué aux images un ImageDataGenerator qui a permis de faire varier l'étirement, le zoom, ainsi que de retourner horizontalement certaines images. Ceci permet de d'augmenter la diversité de notre set d'images. On obtiendra donc peut-être une meilleure généralisation du modèle.

Avec ce CNN on obtient une accuracy de 0,95 et une val_accuracy supérieure à 0,99. Ce phénomène peut être dû à l'utilisation du dropout. Lors de l'entraînement du modèle une partie des fonctionnalités est remise à zéro. (source en référence). Dans notre cas cette partie représente 50% (Dropout à 0.5). Cependant, lors de la phase de validation, la totalité des fonctionnalités est utilisée. Ainsi le modèle est plus robuste sur les données de validation et donc l'accuracy est meilleure.

Lorsque l'on essaie de généraliser le modèle au dataset de test, l'accuracy s'effondre à 0,07. La quasi-totalité des cellules sont classées comme étant des éosinophiles. On obtient la même matrice de confusion et le même classification_report que pour le CNN précédent, malgré l'augmentation du nombre de couches de convolution et la diversifications des données. A posteriori, on peut aussi craindre que le déformement (shear range = 0.2) provoqué par l'étirement est provoqué la perte de caractéristiques importantes de certaines catégories de cellules.

L'utilisation du modèle généré à 10 époques (plutôt que 15 époques), qui correspond au début du plateau des performances n'améliore qu'à la marge la précision sur le Test Dataset avec une valeur de 0.09 (inférieure à la valeur du hasard), donc on peut conclure à une absence de généralisation.

CNN 1 - VARIANT

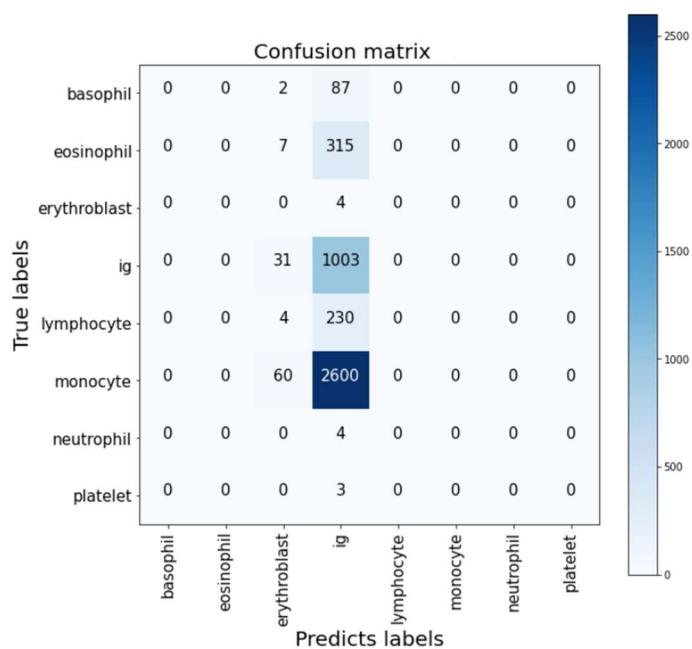
Ce CNN est construit de manière similaire au précédent. Quelques différences, listé ci-dessous, ont permis d'améliorer un petit peu la performance :

Le nombre de paramètres à entraîner est plus réduit pour le variant, 7 037 504, par rapport au premier modèle excessivement lourd (30 388 424, liés notamment à une couche dense de 128 neurones en fin de cycle).

20 couches ont été utilisées contre 15 pour le premier modèle.

En fine, on obtient une accuracy de 0,99 et une val_accuracy de 0,92. Les résultats de validation sont bons, mais des valeurs aussi élevées pourraient révéler un overfitting assez important.

Ce CNN obtient des résultats de prédiction sur le dataset de test meilleurs que les deux précédents CNN. On obtient une accuracy générale de 0,16, qui, tout en restant insuffisante, est légèrement supérieure à la distribution au hasard (0,12). Ici, ce sont les IG qui obtiennent le meilleur f1-score à 0,38 avec une précision de 0,24 et un rappel de 0,97. Ces valeurs ainsi que la matrice de confusion ci-dessous nous montrent que la précision de 0,16 obtenue pour ce modèle est une fausse amélioration. Le modèle CNN1 - Variant classe presque toutes les images en tant qu'IG (granulocyte immature). Il se trouve que c'est une classe contenant beaucoup d'images, d'où la précision globale qui augmente artificiellement.



	precision	recall	f1-score	support
0	0.00	0.00	0.00	89
1	0.00	0.00	0.00	322
2	0.00	0.00	0.00	4
3	0.24	0.97	0.38	1034
4	0.00	0.00	0.00	234
5	0.00	0.00	0.00	2660
6	0.00	0.00	0.00	4
7	0.00	0.00	0.00	3
accuracy			0.23	4350
macro avg	0.03	0.12	0.05	4350
weighted avg	0.06	0.23	0.09	4350

CNN AVEC DES COUCHES D'ACTIVATION VARIEES

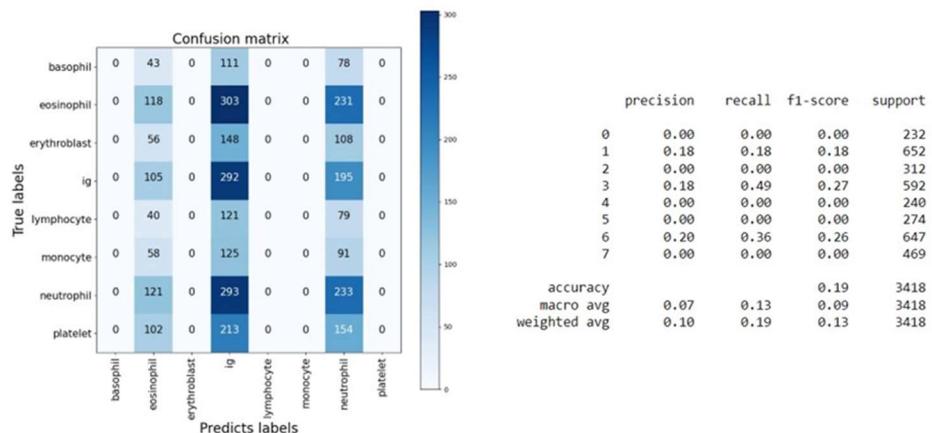
Deux réseaux CNN mettant en œuvre une stratégie différente sont construits :

Ils sont constitués de séries de couches de convolution – maxpooling – dropout, avant passage par une dernière couche cachée Dense et, enfin, la couche (Dense) de sortie.

Ils diffèrent en particulier des précédents par l'utilisation de dropout entre chaque couche de convolution et par l'utilisation de fonctions d'activation différentes d'une couche de convolution à l'autre. L'idée derrière cette tentative était de renforcer la robustesse à l'aide des dropout récurrents, et de tester si la variation dans la fonction d'activation pourrait créer une format de compensation pour les faiblesses de l'une ou l'autre.

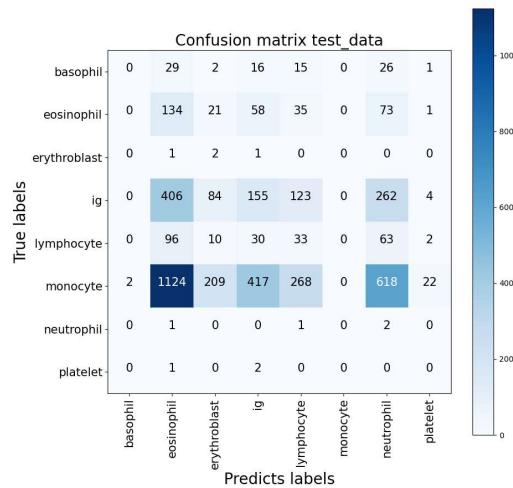
Le premier modèle comporte 4 couches de convolution, activées respectivement par les fonctions LeackyReLU, Relu, tanh, puis Relu. Il entraîne plus de 70000 paramètres:

On observe qu'au bout de quelques epochs, les performances de validation et d'entraînement divergent. Le modèle atteint un plateau et l'entraînement supplémentaire ne semble rien apporter, hormis un risque d'overfitting accru. Le modèle, à l'issu de cet entraînement, affiche une précision d'entraînement de 0,81 et de validation de 0,76 (sur 10 epochs). La matrice de confusion sur ces données d'entraînement est présentée ci-dessous :



La mesure globale de la performance du modèle affiche un résultat de 0.19. Les cellules sont prédites pour les classes éosinophiles, Ig et neutrophiles. Ces résultats sont assez similaires aux résultats du modèle précédent.

La matrice de confusion pour les données ‘Test’ est présentée ci-dessous :



La généralisation est légèrement meilleure que les autres modèles.

Le second modèle est construit en dupliquant la série de couche de convolutions, ce qui permet de faire chuter le nombre de paramètres à entraîner à 17000, afin de voir si diminuer le nombre de paramètres en entrée de la première couche dense permet d'améliorer les performances : cela n'a pas fonctionné.

UTILISATION DES RESEAUX DE NEURONES POUR CLASSEZ LES IMAGES A PARTIR DE DONNEES REDUITES

Les données ont été réduites pour manipulation par les algorithmes de machine learning sur des temps acceptables. Nous souhaiterions comparer les performances des algorithmes de deep learning sur ces données-là, par rapport aux données brutes.

Nous avons réalisé 3 tests à partir des données partiellement réduites :

Après réduction brute de 40% de la taille des images et 20% de réduction intelligente par sélection de pixels porteurs d'information (SelectPercentile), nous disposons de tables de pixels composées de 10035 features par image de départ. Afin de faire tourner un CNN construit pour des images sur ces données, un format artificiel d'image doit être donné : nous appliquons un reshape sur les données partiellement réduites afin de les réorganiser en matrices de 223 par 15 par 3 (car on a bien encore des canaux RGB après les étapes de réduction décrites). Nous avons conscience qu'a priori, l'image ainsi recréée est distordue par rapport à l'image d'origine. Nous ne pouvons donc pas être sûrs que les étapes d'entraînement des réseaux de neurones par convolutions permettront bien de faire ressortir des informations intéressantes. Le modèle sélectionné est le suivant :

```

model = Sequential()

model.add(Conv2D(filters=32,
                 kernel_size=(3,3),
                 padding='valid',
                 activation='relu',
                 input_shape=(223,15,3)))

model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))

model.add(Dropout(rate=0.5))

model.add(Flatten())

model.add(Dense(units=64, activation ='relu'))

model.add(Dense(units=n_class+1, activation='softmax'))

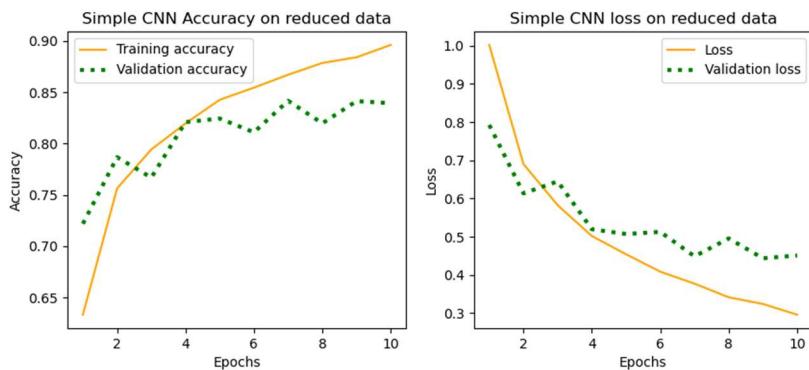
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

Il est très simple, composé d'une couche de convolution avec fonction d'activation "Relu", d'une couche de Pooling des pixels contenant la quantité d'information maximale, d'une couche de dropout forçant l'inactivation de 50% des poids, puis de deux couches denses, l'une de 64 neurones avec fonction d'activation "Relu" et la couche finale de 9 neurones avec activation "softmax" qui renvoie la probabilité d'appartenance à une classe. La compilation est réalisée avec la fonction de perte 'sparse categorical crossentropy', l'optimizer adam et l'accuracy (précision) comme métrique.

Il donne une précision de 0,93 pour les données d'entraînement, et 0,71 pour la validation après entraînement sur 5 epochs avec les données non réduites.

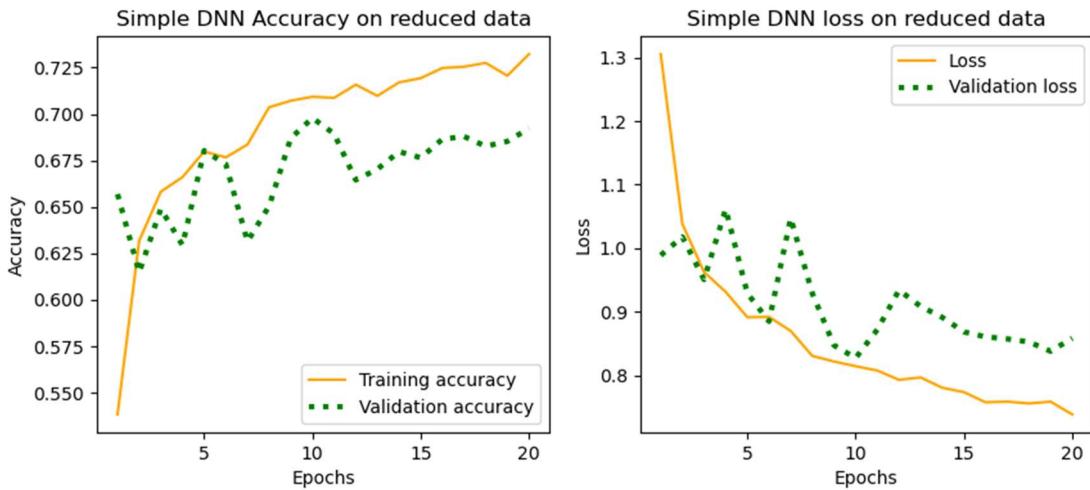
Les résultats obtenus sur ce modèle sur 10 epochs sont présentés sur le graphe ci-dessous. Les courbes d'évolution de la précision et de la fonction de perte n'ont pas encore atteint leur plateau, mais la validation, en revanche, stagne. Il n'y a donc pas d'intérêt à entraîner le modèle au-delà de ces 10 epochs. La précision de validation obtenue est de 0,84, ce qui est bien meilleur que dans le cas de l'entraînement sur données non réduites.



Nous avons donc également entraîné deux réseaux uniquement composé de couches denses, sur les données non redimensionnées (données tabulaires, donc).

L'architecture choisie pour le premier est basée sur l'extraction à l'identique des deux couches denses du modèle précédent. Il est donc uniquement composé de ces deux couches.

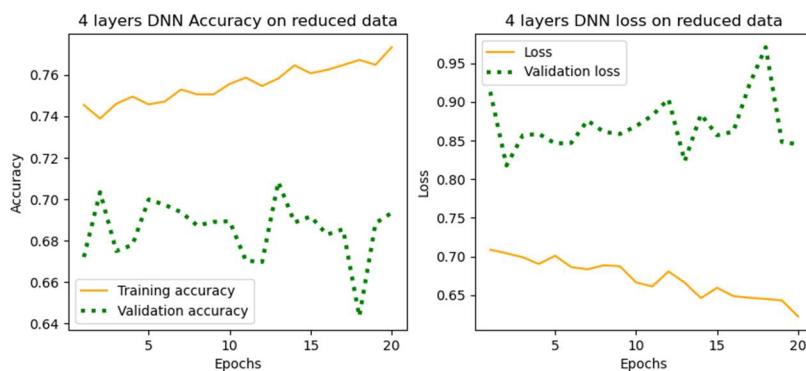
Les résultats obtenus sur ce modèle, pour 20 epochs, sont les suivants :



On voit que l'entraînement, et surtout la validation, sont nettement moins efficaces que dans le cas de la reconstruction de pseudo-images. Nous avons essayé de faire tourner un modèle un petit plus complet en rajoutant quelques couches, une couche dense de 128 neurones toujours avec la fonction d'activation "Relu", et la couche de Dropout à 50% que l'on avait dans le modèle CNN. Ce dernier modèle testé est donc composé comme suit :

```
inputs=Input(shape=10035,name='Input')
dense1=Dense(units=128,activation="relu",name='Couche_1')
couche2=Dropout(rate=0.5,name='Couche_2')
dense3=Dense(units=64,activation="relu",name='Couche_3')
dense4=Dense(units=n_class+1,activation="softmax",name='Couche_4')
```

Et les résultats obtenus, présentés sur le graphe ci-dessous, ne sont pas meilleurs que dans le cas précédent.



Le fait que les données en entrée soient toujours des pixels semble compter fortement : on se prive d'outils permettant de les traiter efficacement en les considérant comme de simples données tabulaires.

En conclusion, ces résultats sont nettement meilleurs que les résultats de ML en ce qui concerne la précision d'apprentissage, et significativement supérieurs pour la précision de validation aussi, puisque la barre des 0,8 est franchie. Ils sont également supérieurs à ceux de DL pour le même CNN, mais pour un nombre d'epochs supérieur. En revanche, l'entraînement est beaucoup plus rapide, donc cette méthode est très prometteuse.

Par ailleurs, ces essais d'utilisation de données réduites par sélection de features se rapprochent de la technique de features selection qui consiste à préparer les données en les faisant traiter par les premières couches d'un réseau de neurones avant de les soumettre à un algorithme de classification. Le test des 3 modèles générés ci-dessus sur les images du dataset test permettront de voir si cette méthode génère moins d'overfitting que l'utilisation simple de CNNs, et si cela vaut la peine d'essayer de mettre en place dans le futur une architecture de features selection conventionnelle.

TRANSFER LEARNING (TL)

La méthode de transfer learning consiste à utiliser le backbone d'un modèle pré-entraîné comme extraction de features, puis ajouter des couches Dense pour traiter le problème de classification.

Il s'agit à récupérer tout ou partie d'un modèle pré-entraîné et de l'utiliser comme un modèle initial auquel on ajoute un nombre réduit de paramètres (couches) avant de reprendre l'apprentissage. Le modèle pré-entraîné aide le modèle à ajuster les poids en fournissant une bonne initialisation de ceux-ci. Généralement les modèles pré-entraînés sont entraînés sur plusieurs tâches et sur une grande quantité de données, le modèle transféré a donc déjà une bonne capacité de représentation et ne nécessite que peu de données et moins de temps d'apprentissage pour être adapté à la tâche cible.

Les couches pré-entraînées sont figées, ce qui signifie qu'elles ne sont pas modifiées pendant le processus d'entraînement sur nos données. Cela permet de conserver les connaissances apprises et le travail se fait uniquement sur l'ajustement des nouvelles couches ajoutées.

Pendant l'entraînement, on peut cependant choisir de "dégeler" ("défreezer") des couches de la partie backbone afin d'affiner les poids du modèle : cette étape est nommée le fine-tuning.

Les hyperparamètres du modèle peuvent être ajustés pour obtenir de meilleures performances sur la tâche cible. Cela peut inclure des hyperparamètres tels que le taux d'apprentissage, la taille du lot (batch size) ou le nombre d'itérations.

EFFICIENTNET

Le modèle EfficientNet est particulièrement remarquable car il parvient à obtenir des performances élevées tout en maintenant une efficacité en termes de paramètres et de calculs.

Le modèle EfficientNet utilise un CNN pour extraire des caractéristiques de l'image en utilisant des couches de convolution. Il se compose généralement de plusieurs blocs de convolution, suivis de couches de mise en commun (pooling) et de couches entièrement connectées pour la classification finale.

Une autre caractéristique importante d'EfficientNet est l'utilisation de la régularisation de l'apprentissage appelée "DropConnect". Elle consiste à supprimer aléatoirement certaines

connexions entre les neurones pendant l'entraînement, ce qui permet de prévenir l'overfitting et d'améliorer la généralisation du modèle.

A ce modèle, ont été ajoutées : une couche de GlobalAveragePooling2D, deux paires de couches Dense et Dropout à 0,2, une couche Flatten et enfin une couche Dense. Les fonctions d'activation sont des 'relu' à l'exception de la fonction de la dernière qui est un 'softmax'. Le modèle est compilé avec la loss 'sparse categorical cross entropy', l'optimiseur 'adam' et la métrique 'accuracy'. Le modèle est entraîné sur 5 epochs.

On obtient une première accuracy de 0,91 et une val_accuracy de 0,93. Cependant, lorsque l'on évalue le modèle sur les données test, on descend à une accuracy de 0,0006 ce qui est très en dessous des résultats obtenus avec les CNN.

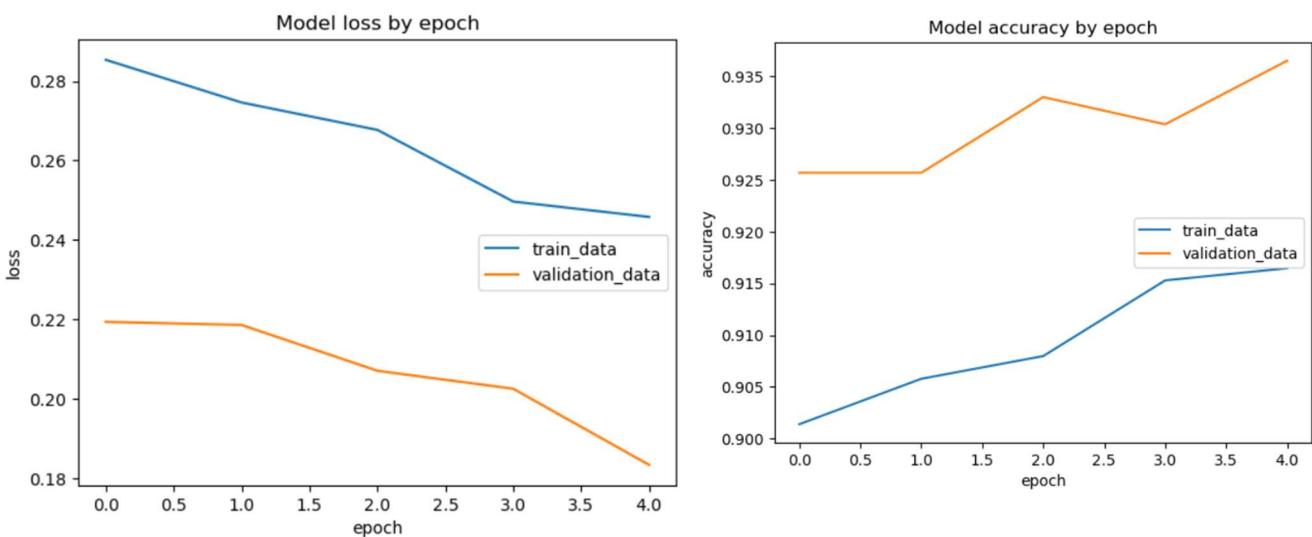
Nous avons donc défreezé 4 couches du modèle EfficientNet. Cette fois, après entraînement, on obtient une accuracy de 0,92 et un val_accuracy de 0,93. L'accuracy s'améliore également lorsque l'on évalue le modèle, elle passe à 0,015. Malgré cette amélioration, cela reste en deçà de ce qui a été obtenu avec les CNN.

Ce modèle est composé de deux couches de convolution en plus du MaxPooling. Cela permet de réduire la taille des caractéristiques tout en préservant les informations les plus importantes. Chaque couche convulsive possède une fonction d'activation relu, qui introduit la non-linéarité dans le modèle. La couche de Flatten n'a pas été ajouté contrairement au CNN précédent. Ce modèle est également compilé avec un optimiseur adam, une losses.SparseCategoricalCrossentropy pour la fonction de perte et enfin par l'accuracy correspondant à la métrique. Des callbacks ont été ajouté afin de contrôler le flux d'exécution d'un algorithme d'apprentissage automatique. Ils effectuent des actions spécifiques à des moments clés pendant l'entraînement d'un modèle.

Il est généralement souhaitable d'obtenir une performance élevée à la fois sur les données d'entraînement (accuracy d'entraînement) et sur de nouvelles données non vues auparavant (accuracy de validation). Cependant, ici l'accuracy de validation est supérieure à l'accuracy d'entraînement, ce qui peut sembler contre-intuitif. Les raisons qui expliquent cela peuvent être :

La taille du jeu de données d'entraînement est relativement petite, le modèle peut facilement surapprendre les données d'entraînement et obtenir une accuracy élevée sur celles-ci. Cependant, lorsque le modèle est évalué sur de nouvelles données (validation), il peut ne pas généraliser correctement et obtenir une accuracy inférieure.

Le surapprentissage se produit lorsque le modèle devient trop complexe par rapport à la quantité de données disponibles. Dans ce cas, le modèle peut mémoriser les exemples d'entraînement spécifiques au lieu d'apprendre des schémas généraux. Cela peut entraîner une accuracy élevée sur les données d'entraînement, mais une performance plus faible sur les données de validation, car le modèle ne généralise pas correctement.



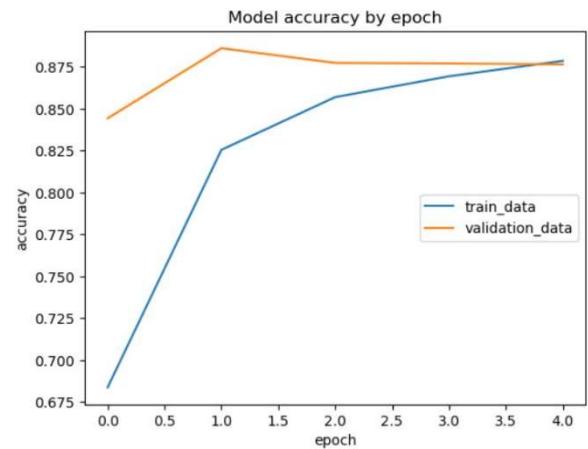
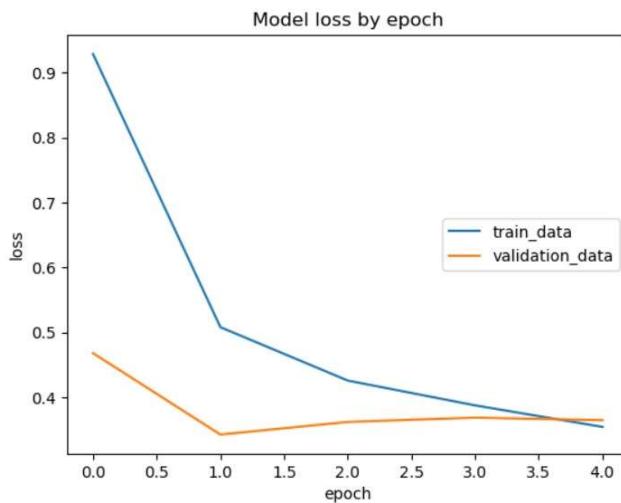
Layer (type)	Output Shape	Param #
<hr/>		
efficientnetb1 (Functional)	(None, 12, 12, 1280)	6575239
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 1024)	1311744
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 8)	4104
<hr/>		
Total params:	8,415,887	
Trainable params:	1,840,648	
Non-trainable params:	6,575,239	

VGG16

Le modèle VGG16, que nous avons ensuite utilisé, est un modèle pré-entraîné sur des images de la base ImageNet.

Nous ne devons pas utiliser d'autres transformations d'augmentation car le réseau VGG16 se comporte mal dans ce cas (c'est un réseau déjà pré-entraîné).

Layer (type)	Output Shape	Param #				
vgg16 (Functional)	(None, None, None, 512)	14714688				
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 512)	0				
dense_8 (Dense)	(None, 1024)	525312	recall	f1-score	support	Les cellules
dropout_5 (Dropout)	(None, 1024)	0	0.05	0.06	232	sont en
dense_9 (Dense)	(None, 512)	524800	0.19	0.20	652	partie
dropout_6 (Dropout)	(None, 512)	0	0.05	0.05	312	prédites
dense_10 (Dense)	(None, 8)	4104	0.20	0.18	592	
			0.08	0.10	240	
			0.20	0.19	274	
			0.13	0.13	647	
					469	
					3418	
			0.15			
			0.12			
			0.15			
			0.14			
Total params:	15,768,904					
Trainable params:	1,054,216					
Non-trainable params:	14,714,688					



l'ensemble des 8 classes de cellules. La prédiction est plus répartie pour ces différentes classes par rapport aux autres modèles cité précédemment. La mesure globale de la performance du modèle affiche un résultat de 0.15.

DENSENET121

Une version légèrement adaptée du DenseNet121 (composé des couches suivantes 117-conv, 3-transition, and 1-classification) est développée du fait des bonnes performances générales du modèle.

Afin de favoriser la généralisation de l'apprentissage, une augmentation de données est réalisée via le data generator avec les paramètres suivants, pour accomoder la reconnaissance d'images prises

avec une exposition, un éclairage ou un grossissement légèrement différents (zoom_range=0.2, horizontal_flip=True, vertical_flip=True, brightness_range=[0.8, 1.2])

```
Total params: 7,301,960  
Trainable params: 264,456  
Non-trainable params: 7,037,504
```

Les résultats obtenus sur 12 epochs sont comme suit

```
Epoch 12/12  
395/395:1895s 5s/step - loss: 0.1860 - acc: 0.9364 - val_loss: 0.2125 - val_acc:  
0.9316
```

Malheureusement, le modèle h5 n'a pas été correctement sauvegardé et l'apprentissage doit être réeffectué pour un test sur données externes.

(GOOGLE) INCEPTION V3

Inception V3 est un modèle qui a été entraîné sur les données ImageNet et qui atteint une précision de 78,1%. Le modèle est composé de couche de convolution, de pool moyen, de pool maximal, de concaténation, d'abandon et de couche entièrement connectées. La perte est calculée par un softmax.

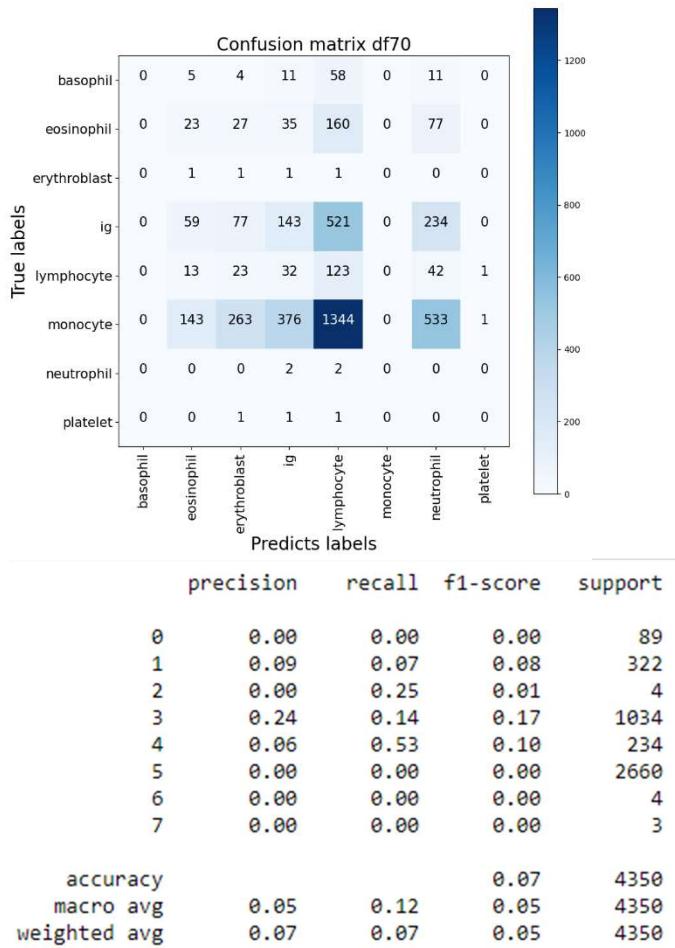
Un premier entraînement est réalisé sans defreezer de couche du modèle inception. Ajouté au modèle on trouve ensuite une couche de GlobalAveragePooling, suivi de deux paires de couche dense avec l'activation 'relu' et dropout et enfin une dernière couche dense avec la fonction d'activation 'softmax'. Le modèle est compilé avec la loss 'sparse categorical crossentropy', l'optimiseur adam, et la métrique 'accuracy'. Le modèle est ensuite entraîné sur 5 epochs.

A la fin de ce premier entraînement, on obtient une accuracy de 0,41 et ne val_accuracy de 0,44. Les résultats sont bien inférieurs à ceux des modèles convolutifs et au Transfer Learning précédent.

Ainsi un second entraînement a été réalisé en défreezant cette fois les quatre dernières couches du modèle Inception V3. Le modèle passe a une accuracy de 0,47 et une val_accuracy de 0,55.

Le modèle s'améliore seulement très légèrement comparé aux autres modèles de Transfer Learning qui ont été entraînés sur le même nombre d'epochs. InceptionV3 ne semble pas être un modèle adéquate pour nos données.

Concernant l'évaluation de la généralisation sur les données test Rabiin, le modèle obtient une accuracy de 0.07. Les cellules sont peu ou pas détectée par le modèle. A partir du classification report ci-dessous, on peut constater tout de même que le modèle parvient à mieux classer les lymphocytes (rappel à 0.53).



DEVELOPPEMENT

Malgré un travail assez conséquent, plusieurs points restent à explorer et/ou améliorer.

Tout d'abord notre dataset d'entraînement est assez déséquilibré. Il faudrait rééquilibrer le nombre d'images par catégorie de cellule au moment de l'entraînement des modèles de deep learning, soit par suppression d'un certain nombre d'images dans les catégories sur-représentées, soit en jouant sur le poids de chaque image dans le modèle (comme cela a été testé en machine learning). On pourrait également essayer de faire un oversampling.

Ensuite, nous avons réalisé une PCA sur les images moyennées que nous n'avons pas pu exploiter. Il serait intéressant de transformer les données brutes à l'aide de la PCA sur images moyennées, et ensuite de refaire au moins le modèle de machine Learning qui a obtenu les meilleurs scores.

Concernant plus spécifiquement la partie Machine Learning, une tentative d'étudier l'influence de la différence de taille des échantillons de données de chaque classe a été réalisée à l'aide de poids donnés à chaque classe. Un GridSearchCV sur 3 poids par classe avait été lancé, mais le temps de calcul s'est avéré trop long (arrêt par « keyboard interrupt »). Une amélioration du code pour palier à ce problème serait à prévoir.

Par ailleurs, nous n'avons pas encore eu le loisir de tester des modèles utilisant la “feature selection”. En effet, en utilisant un modèle de Deep Learning ou de Transfert Learning en amont sur les données, ensuite appliquer un modèle de Machine Learning sur des données pré-traitées et voir si cela permettrait d'améliorer la détection et la classification des cellules.

Une dernière étape intéressante aurait été de comparer la vitesse de calcul de chaque modèle. En effet, pour l'usage professionnel du modèle, il faut que celui soit performant mais également assez rapide, surtout dans le cas de diagnostic d'une maladie.

Enfin, la partie détection des différents cancers restent encore à faire. Lorsqu'un modèle sera efficace sur les images de cellules saines, il faudra ajouter la détection de cellules malades. Cela devrait permettre d'identifier une cellule malade ou saine mais également de connaître le type de la cellule.

ANNEXES

ANNEXE 1 : EXTRAIT ETUDE (ZAHRA MOUSAVI KOUZEHKANAN, 2022),

Tableau de résultats extrait de l'étude (Zahra Mousavi Kouzehkanan, 2022), reflétant les scores de prédiction sur 5 classes sur le dataset de première lecture (non incluant les plaquettes (PLA), immature granulocytes (IG) et erythroblasts exclus)

Test sur échantillons correspondant au test A du Raabin-WBC dataset (5 classes, hors platelet, ig, erythroblast)
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8782871/>

The results of different pre-trained models as well as Tavakoli et al.⁵⁰ on the test-A dataset.

Methods	Lymph			Mono			Neut			Eosi			Baso			Acc (%)
	P (%)	S (%)	F1 (%)	P (%)	S (%)	F1 (%)	P (%)	S (%)	F1 (%)	P (%)	S (%)	F1 (%)	P (%)	S (%)	F1 (%)	
ResNet18 ⁴⁴	98.84	99.23	99.08	96.96	95.30	96.12	99.66	99.36	99.5	95.77	98.45	97.09	100	100	100	99.06
ResNet34 ⁴⁴	98.66	99.52	99.09	96.93	94.44	95.67	99.85	99.14	99.49	94.67	99.38	96.97	100	100	100	99.01
ResNet50 ⁴⁴	98.47	99.52	98.99	97.36	94.44	95.88	99.74	99.40	99.57	96.94	98.45	97.69	100	100	100	99.10
ResNext50 ⁴⁵	98.01	100	98.99	99.53	91.45	95.32	99.74	99.55	99.64	97.85	98.76	98.30	100	100	100	99.17
MnasNet1 ⁴⁸	98.93	98.65	98.79	94.14	96.15	95.14	99.66	98.76	99.21	92.15	98.45	95.20	100	100	100	98.59
MobileNet-V2 ⁴⁷	98.85	99.32	99.08	96.93	94.44	95.67	99.66	99.40	99.53	96.97	99.38	98.16	100	100	100	99.12
DenseNet121 ⁴⁶	98.38	99.61	98.99	97.72	91.45	94.48	99.70	99.14	99.42	94.40	99.38	96.82	100	100	100	98.87
ShuffleNet-V2 ⁴⁹	98.09	99.32	98.70	96.49	94.02	95.24	99.74	99.36	99.55	97.85	98.76	98.30	100	100	100	99.03
VGG16 ⁴³	98.26	98.26	98.26	95.09	91.03	93.01	99.43	98.57	99	89.01	98.14	93.35	100	100	100	98.09
Tavakoli et al. ⁵⁰	97.23	95.07	96.14	84.87	86.32	85.59	98	95.60	96.78	72.24	91.30	80.66	96.59	95.51	96.05	94.65

Table 7

The number of samples in training data, test-A, and test-B.

Sets	Lymph	Mono	Neut	Eos	Bas
Training data	2427	561	6231	744	212
Test-A	1034	234	2660	322	89

Les réseaux de neurones convolutifs sont utilisés depuis plusieurs années avec des modèles de référence qui se sont succédés. 2 modèles de bases ont été utilisés également pour notre projet:

- VGG 16
- DenseNet121

ANNEXE 2 : RESUME DES PERFORMANCES DES MODELES

Résumé des différents modèles explorés pendant le projet : une mise à jour est en cours, et le tableau sera complété d'ici à la présentation

Name of model	Name of h5 file and hyperlink to G Colab	Type données d'entrée	Brève description (base model en cas de transfer learning)	Nom notebook sur github	Train Accuracy	Train Loss	Validation accuracy	Validation Loss	ML Scores	Metrics Performance :Numéro de l'éPOCH avec meilleure performance Test data sur échantillon externe : general accuracy best f1 (nom de la classe)	Temps de prédiction Test data	Commentaire sur performance
KNN données non normalisées										score Minko max pour 0.7, 0.8, 0.9 : 0.7859 0.76046 0.76046 pour k= 16 14 16		
KNN données normalisées										score max pour 0.7, 0.8, 0.9, courbe test minkowski : 0.75402 0.66189 0.29599		
KNN données normalisées										df70 Accuracy 0.74 F1 entre 0.47 et 0.97 selon classe		
KNN données normalisées										Df80 Accuracy 0.6 F1 entre 0.2 et 0.86 selon classe		

Name of model	Name of h5 file and hyperlink to Colab	Type données d'entrée	Brève description (base model en cas de transfer learning)	Nom notebook sur github	Train Accuracy	Train Loss	Validation accuracy	Validation Loss	ML Scores	Metrics Performance :Numéro de l'époch avec meilleure performance Test data sur échantillon externe : general accuracy best f1 (nom de la classe)	Temps de prédiction Test data	Commentaire sur performance
KNN données normalisées									Df90 Accuracy 0.14 F1 entre 0.0 et 0.14 selon classe			
KNN données non normalisées			CV3						Df90 Accuracy 0.72 F1 entre 0.45 et 0.96			
Régression linéaire		Df70	CV3						Accuracy 0.70 F1 entre 0.30 et 0.97			
Régression linéaire		Df80	CV3						Accuracy 0.70 F1 entre 0.28 et 0.98			
Régression linéaire		Df90	CV3						Accuracy 0.69 F1 entre 0.35 et 0.98			
Régression linéaire		Df70	CV15						Accuracy 0.70 F1 entre 0.37 et 0.98			
Régression linéaire		Df80	CV15						Accuracy 0.70 F1 entre 0.36 et 0.98			

Name of model	Name of h5 file and hyperlink to Colab	Type données d'entrée	Brève description (base model en cas de transfer learning)	Nom notebook sur github	Train Accuracy	Train Loss	Validation accuracy	Validation Loss	ML Scores	Metrics Performance :Numéro de l'époche avec meilleure performance Test data sur échantillon externe : general accuracy best f1 (nom de la classe)	Temps de prédiction Test data	Commentaire sur performance
Régression linéaire		Df90	CV15						Accuracy 0.69 F1 entre 0.35 et 0.97 (CV# à confirmer) Rapport à regénérer)			
Voting Classifier									Accuracy 0.75 F1 entre 0.53 et 0.98			
Random Forest		DF70							Accuracy 0.76 F1 entre 0.55 et 0.97			
Random Forest		DF80							Accuracy 0.75 F1 entre 0.48 et 0.97			
Random Forest		DF90							Accuracy 0.68 F1 entre 0.21 et 0.95			
CNN 1	Model 15 CNN RB.h5	Images 360x360 Data generator: Normalisé (/255), Shear 0.2, zoom .02	CNN from scratch Normalisé (/255), Shear 0.2, zoom .02	RBC CNN manuel acc 95 val acc99 epochs 15 et tests.ipynb	0.9545	0.1339	0.9940	0.0232		Epoch 15/15 Accuracy: 0.07 Best f1-score: 0.14 (eosinophil)	157s	Absence de généralisation évidente sur test de données externes..

Name of model	Name of h5 file and hyperlink to Colab	Type données d'entrée	Brève description (base model en cas de transfer learning)	Nom notebook sur github	Train Accuracy	Train Loss	Validation accuracy	Validation Loss	ML Scores	Metrics Performance :Numéro de l'époche avec meilleure performance Test data sur échantillon externe : general accuracy best f1 (nom de la classe)	Temps de prédiction Test data	Commentaire sur performance
CNN 1	Model_10 sur 15 CNN RB.h5	Images 360x360 data generator, Normalisé (/255), Shear 0.2, zoom .02	CNN from scratch	A venir	0.9312	0.1930	0.9312	0.0734		Epoch 10/15 Accuracy: 0.09 Best f1-score: 0.14 (eosinophil)	60s	La sélection de l'époque d'installation du plateau n'améliore pas significativement la généralisation.
CNN1 - VARIANT		Images va		CNN_3_EfficientNet cours.ipynb	VGG16_en	0.99		0.93		Epoch 20 Accuracy 0.16 Best f1 0.38 pour Ig, 0 pour autres catégories		Meilleur f1-score à 0,38 avec une précision de 0,24 et un rappel de 0,97. Le modèle ne détecte pas correctement les IG mais lorsqu'il les détecte il les classe correctement
LeNet variant				Architecture_LeNet.ipynb		0.939	0.1894	0.7156	224	Epoch 5 Accuracy 0.07 F1 0 sauf classe 1 à 0.14		Très faible généralisation
CNN couche activation variées				DL_donnees_reduites.ipynb		0.81		0.76		Epoch 10/10		
EfficientNet			EfficientNet avec ajout de couches	CNN_3_EfficientNet VGG16		0.908	0.27	0.933	0.20	Epoch 5 accuracy 0.06		
EfficientNet			EfficientNet avec 4 couches défreezé	CNN_3_EfficientNet VGG16		0.925	0.216	0.93	0.200	Epoch 5 Accuracy 0.15		

Name of model	Name of h5 file and hyperlink to Colab	Type données d'entrée	Brève description (base model en cas de transfer learning)	Nom notebook sur github	Train Accuracy	Train Loss	Validation accuracy	Validation Loss	ML Scores	Metrics Performance :Numéro de l'époche avec meilleure performance Test data sur échantillon externe : general accuracy best f1 (nom de la classe)	Temps de prédiction Test data	Commentaire sur performance
DenseNet121 modifié		Images 360x360 normalisées via data generator	DenseNet121 Datagenerator Normalisé / 255, horizontal flip, vertical flip, zoom .02, brightness range [0.8, 1.2]	TL DenseNet121	0.9364	0.1860	0.9316	0.2125		Epoch 12 Accuracy 0.07, F1max à 0.35	Non disponible, À venir	
DenseNet121 modifié, unfrozen	Non enregistré	Images 360x360 normalisées via data generator	DenseNet121, 4 couches unfrozen Datagenerator Normalisé / 255, horizontal flip, vertical flip, zoom .02, brightness range [0.8, 1.2]	À venir	À venir	À venir	À venir	À venir			Non disponible	
Google Inception V3	Non enregistré		Function activation softmax, loss sparse categorical crossentropy.	Pynt_of_Blood_Google_Inception_V3.ipynb	0.50		0.51			Epoch 5 Accuracy 0.052	1050s/epochs	
Google Inception V3	Non enregistré		4 dernières couches unfrozen	Pynt_of_Blood_Google_Inception_V3.ipynb	0.52		0.57					

REFERENCES ET BIBLIOGRAPHIE

al., Z. M. (2022). A large dataset of white blood cells containing cell locations and types, along with segmented nuclei and cytoplasm. *Scientific Reports (Sci Rep) nature research.* <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8782871/>

<https://keras.io/api/applications/densenet/>

"Higher validation accuracy, than training accuracy using Tensorflow and Keras" .
<https://stackoverflow.com/questions/43979449/higher-validation-accuracy-than-training-accuracy-using-tensorflow-and-keras>