



DataScientest • com

Rapport Technique d'évaluation

Classification des cellules sanguines - Leukopy

Promotion Mars 2021

Participants :

Mathieu Sarrat

Marie-Anne Mawhin

Laleh Ravanbod

Yahia Bouzoubaa

Contexte

Les globules blancs, plaquettes et globules rouges sont des composants cruciaux des réponses immunitaires. La présence de précurseurs des cellules sanguines dans la circulation est une indication importante d'infections ou de leucémies (Figure 1). Ainsi, plusieurs pathologies peuvent être détectées par une analyse hématologique, par exemple l'identification et la classification des leucocytes, des plaquettes et des érythrocytes est cruciale dans le diagnostic de l'anémie régénérative. Alors que plusieurs instruments de comptage automatique tels que le cytomètre en flux ou l'hémocytomètre fournissent des informations sur les comptes sanguins, l'évaluation qualitative de la cellule par imagerie est souvent nécessaire. Par exemple, en clinique, les frottis sanguins sont indiqués en cas de SIDA, de septicémie, de défaillance d'organe, de leucocytose inexplicable, d'anémie ou de leucémie suspectée.

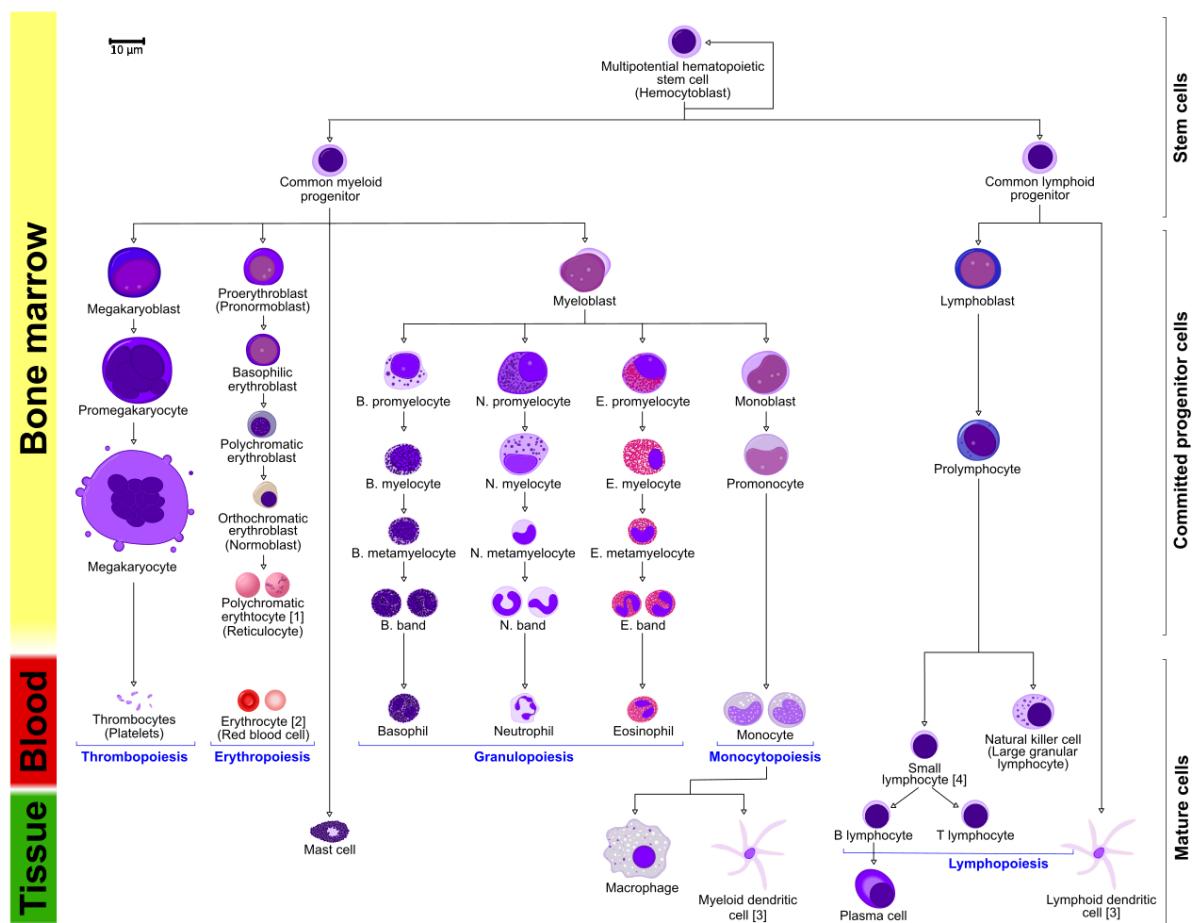


Figure 1 : Cellules sanguines périphériques et précurseurs.

L'identification manuelle des cellules sanguines périphériques est longue et coûteuse et nécessite la présence d'un spécialiste formé. De plus, la classification de certains précurseurs, comme les myélocytes et métamyélocytes, est sujette à de nombreuses controverses inter-spécialistes. Dans des contextes de recherche médicale, où les frottis sanguins sont souvent utilisés pour évaluer la qualité d'une préparation cellulaire, l'identification manuelle est un challenge. Enfin, l'évaluation manuelle est sujette à des résultats erronés. L'analyse assistée par ordinateur des frottis sanguins et l'identification des cellules anormales fourniraient une aide cruciale pour la recherche et les cliniciens, en termes de coût, de temps et de qualité.

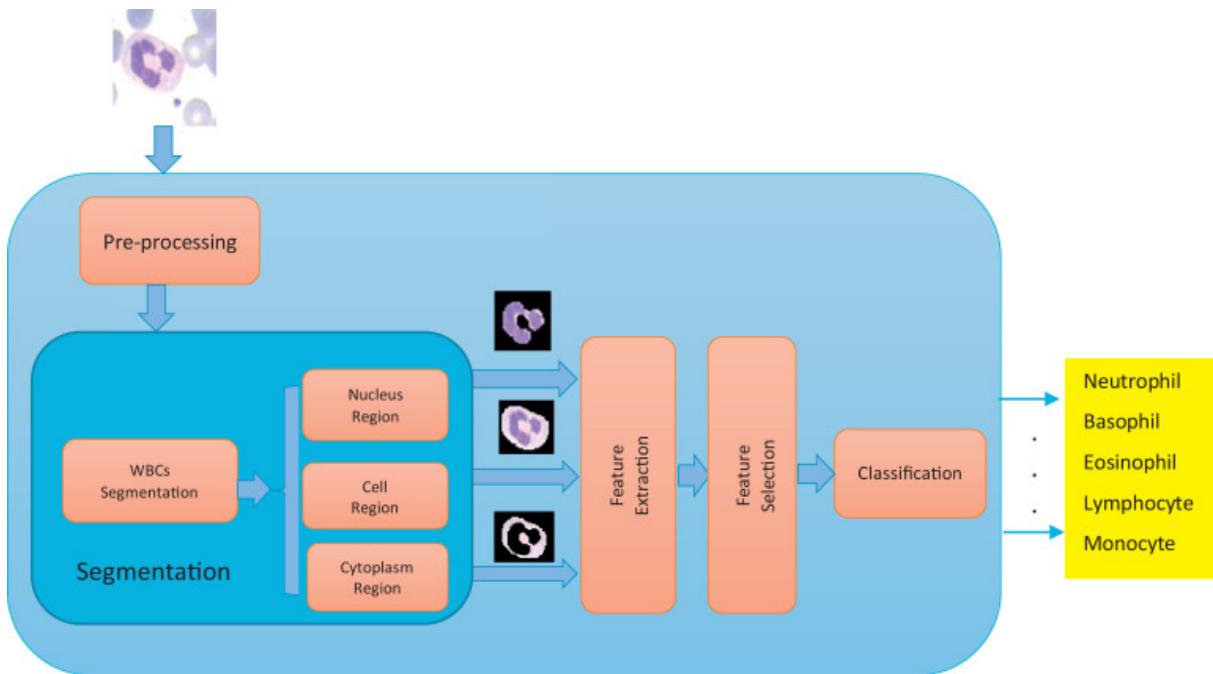


Figure 2: Identification des cellules sanguines selon un workflow classique.¹⁾

Traditionnellement, les systèmes de reconnaissance des globules blancs reposent sur la segmentation, la séparation des composants, l'extraction de caractéristiques et la classification des globules blancs par des modèles de shallow machine learning (Figure 2). Ce type de méthode est difficile à généraliser, notamment, par la variabilité des colorations de Romanowsky et des systèmes d'acquisition, et par l'exigence non négligeable de sa conception en termes de pré-processing et d'extraction des features. Dans ce contexte, l'utilisation du deep learning prend tout son sens.

Objectifs

L'objectif principal de cette étude est de développer un réseau de neurones profonds capables de classifier les cellules saines du sang en 8 ou 11 classes principales de cellules sanguines directement à partir d'images :

- neutrophiles (segmentés) - SNE
- éosinophiles - EO
- basophiles - BA
- lymphocytes -LY
- monocytes -MO
- granulocytes immatures (métamyélocytes, myélocytes, promyélocytes) et “band” - IG ou séparés - MMY, MY, PMY, BNE
- plaquettes - PLT
- érythroblastes - ERB

Un objectif secondaire est de classifier les cellules anormales, leucémiques, cependant par manque de ressources cet objectif a été écarté.

Une revue de la littérature a mis à jour plusieurs publications sur la classification des cellules sanguines par machine learning. La plupart des références utilisent des méthodes de segmentation, une phase d'extraction

¹ "White blood cells identification system based on convolutional deep" <https://pubmed.ncbi.nlm.nih.gov/29173802/>. Accessed 27 Sep. 2021.

de features suivie de classifieur soit deep soit shallow². Cependant, plusieurs publications sont apparues récemment : les modèles présentés reposent sur le deep learning uniquement¹, avec des données pré-segmentées³ limitées à 5-6 classes^{4,5,6} ou un dataset unique^{7,8,9}. Ces articles scientifiques nous ont permis de mettre au point un workflow de base, notamment pour l'augmentation de données et le choix des modèles, plutôt de type vgg. Cela nous a permis aussi de déterminer des pistes d'amélioration, comme par exemple dans la généralisation limitée des modèles entraînés sur un seul jeu de données¹⁰.

L'équipe a quelques bases en biologie et analyse d'images. Marie-Anne est à l'origine du projet, elle a un doctorat en immunologie et elle a donc des connaissances sur les cellules sanguines bien qu'elle ne soit pas experte en hématologie et cytologie. Mathieu est agrégé de physique et a un doctorat en physique des plasmas, cette problématique est donc nouvelle pour lui. Laleh est ingénieur-docteur en automatique et pour elle aussi le sujet est nouveau. Toutefois, ayant une fille en 6ème année de médecine, elle a pu en profiter pour avoir des explications supplémentaires sur la problématique. Dans le cadre de ce projet, elle a contacté le site de diffusion des données Raabin pour leur indiquer que leur données n'étaient pas accessibles. Heureusement, cette action a fixé le problème.

² "Deep learning approach to peripheral leukocyte recognition - PLOS." 25 Jun. 2019, <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0218808>. Accessed 26 Sep. 2021.

³ "Fine-grained leukocyte classification with deep residual learning for" <https://www.sciencedirect.com/science/article/pii/S0169260718300567>. Accessed 26 Sep. 2021.

⁴ "White blood cells detection and classification based on regional" <https://www.sciencedirect.com/science/article/pii/S0306987719310680>. Accessed 26 Sep. 2021.

⁵ "Fine-grained leukocyte classification with deep residual learning for" <https://www.sciencedirect.com/science/article/pii/S0169260718300567>. Accessed 26 Sep. 2021.

⁶ "Detection of subtype blood cells using deep learning - ScienceDirect." <https://www.sciencedirect.com/science/article/pii/S1389041718303760>. Accessed 26 Sep. 2021.

⁷ "White blood cell differential count of maturation stages in ... - PLOS." 11 Dec. 2017, <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189259>. Accessed 26 Sep. 2021.

⁸ "Recognition of peripheral blood cell images using convolutional" <https://www.sciencedirect.com/science/article/pii/S0169260719303578>. Accessed 26 Sep. 2021.

⁹ "Human-level recognition of blast cells in acute myeloid leukaemia." 12 Nov. 2019, <https://www.nature.com/articles/s42256-019-0101-9>. Accessed 26 Sep. 2021.

¹⁰ "Generalizability in White Blood Cells' Classification Problem | bioRxiv." 28 Jul. 2021, <https://www.biorxiv.org/content/10.1101/2021.05.12.443717v3>. Accessed 26 Sep. 2021.

Data

Cadre

Cellules saines

1. Normal_Barcelone :

Nous avons commencé notre étude avec un jeu de données nettoyé par des experts pathologistes de l'Hôpital Clinique de Barcelone. Ce dataset constitue plus de 17000 images de cellules sanguines issues de frottis de donneurs sains et colorées par May-Grunwald Giemsa (MGG) (Table 1)¹¹.

2. Normal_CVB :

Nous avons aussi trouvé un petit dataset issu du blog de Cellavision et contenant 100 images de donneurs sains¹². Au vu de la taille de ce jeu de données, nous avons choisi de le garder comme validation et test de la **généralisabilité** de notre modèle (Table 1).

3. Normal_Raabin :

Nous avons ensuite eu accès à une base de données d'environ 15000 images, incluant les 5 principales classes de cellules sanguines¹³. Tous les échantillons sont sains mis à part quelques basophiles imagés d'un patient atteint de leucémie et viennent de trois laboratoires en Iran : Razi Hospital à Rasht, Gholhak Laboratory, Shahr-e-Qods Laboratory et Takht-e Tavous Laboratory à Téhéran.

Cellules leucémiques et saines

1. AML_Munich :

Nous avons utilisé le dataset AML Morphology Dataset qui contient 18,365 images de frottis sanguins labellisées par des experts et issues de 100 patients diagnostiqués avec une leucémie myéloïde aiguë et de 100 patients "sains" à l'Hôpital de Munich (Table 1)¹⁴.

2. APL_AML_Johns_Hopkins :

Nous avons aussi exploré un dataset de cellules sanguines cancéreuses de type leucémie myéloïde mais après une première exploration, beaucoup de labels se sont révélés erronés. Ce jeu de données a donc été écarté face à l'impossibilité de re-labelliser les données¹⁵.

¹¹ "A dataset of microscopic peripheral blood cell images for ... - NCBI." 8 Apr. 2020, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7182702/>. Accessed 18 Sep. 2021.

¹² "Fast and robust segmentation of white blood cell images by self" <https://www.sciencedirect.com/science/article/pii/S0968432817303037>. Accessed 18 Sep. 2021.

¹³ "Raabin-WBC: a large free access dataset of white blood ... - bioRxiv." 29 May. 2021, <https://www.biorxiv.org/content/10.1101/2021.05.02.442287v4>. Accessed 26 Sep. 2021.

¹⁴ "A Single-cell Morphological Dataset of Leukocytes from AML" 17 Feb. 2021, <https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=61080958>. Accessed 26 Sep. 2021.

¹⁵ "Deep Learning for Distinguishing Morphological Features of Acute" 5 Nov. 2020, <https://ashpublications.org/blood/article/136/Supplement%201/10/472209/Deep-Learning-for-Distinguishing-Morphological>. Accessed 18 Sep. 2021.

	Normal Barcelone	Normal CVB	Normal_Raabin	AML_Munich
origin	Anonymised healthy blood smears stained with MGG (automatic stainer, QC)	Healthy donors	73 blood smears from healthy donors	100 patients diagnosed with Acute Myeloid Leukemia at Munich University Hospital between 2014 and 2017 + 100 healthy donors
staining	MGG	unknown (probably Giemsa)	Giemsa	Giemsa
size	17092 RGB images of individual cells	100 RGB images	14514 RGB images	18365 RGB images
extension	.jpg	.bmp	.jpg	.tiff
size	360 x 363 px	300 x 300 px	575 x 575 px	400 x 400 px
Acquisition system	CellaVision	CellaVision	Olympus microscope + Samsung camera	M8 digital microscope / scanner
annotation	Expert pathologist	NA	Two independent expert pathologists	Expert at Munich, re-annotated 2 times to account for variability inter-expert.
Cell type	8 Neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes (metamyelocytes, myelocytes and promyelocytes), erythroblasts, platelets	naming: in class_label_CVB.csv, labels (1- 5) correspond respectively to neutrophil, lymphocyte, monocyte, eosinophil and basophil.	Neutrophils, eosinophils, basophils, lymphocytes, monocytes,	BAS Basophil EBO Erythroblast EOS Eosinophil KSC Smudge cell LYA Lymphocyte (atypical) LYT Lymphocyte (typical) MMZ Metamyelocyte MOB Monoblast MON Monocyte MYB Myelocyte MYO Myeloblast NGB Neutrophil (band) NGS Neutrophil (segmented) PMB Promyelocyte (bilobled) PMO Promyelocyte
Ref	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7182702/	https://ashpublications.org/blood/article/136/Supplement%201/10/472209/Deep-Learning-for-Distinguishing-Morphological	https://www.biorxiv.org/content/10.1101/2021.05.02.442287v4?ijkey=7a2390aa1717177c60a3b67203f28fdcf23208d4&keytype2=tf_ipsecsha	https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=61080958

Table 1: Description des datasets.

Pertinence

Avez vous eu à nettoyer et à traiter les données ? Si oui, décrivez votre processus de traitement.
Quelles particularités de votre jeu de données pouvez-vous mettre en avant ?

La variable cible dépend de la complexité choisie dans la classification des cellules, soit nous intégrons 8 classes :

- neutrophiles (segmentés) - SNE
- éosinophiles - EO
- basophiles - BA
- lymphocytes - LY
- monocytes - MO
- granulocytes immatures (métamyélocytes, myélocytes, promyélocytes, ‘band’)
- plaquettes - PLT
- érythroblastes - ERB

Soit nous intégrons 11 classes avec les granulocytes séparés : MMY, MY, PMY, BNE.

Dans un premier temps nous avons exploré le dataset “Normal_Barcelone”. Les classes ne sont pas équilibrées avec une prédominance de neutrophiles et éosinophiles et un plus faible nombre de basophiles et lymphocytes (Fig. 3).

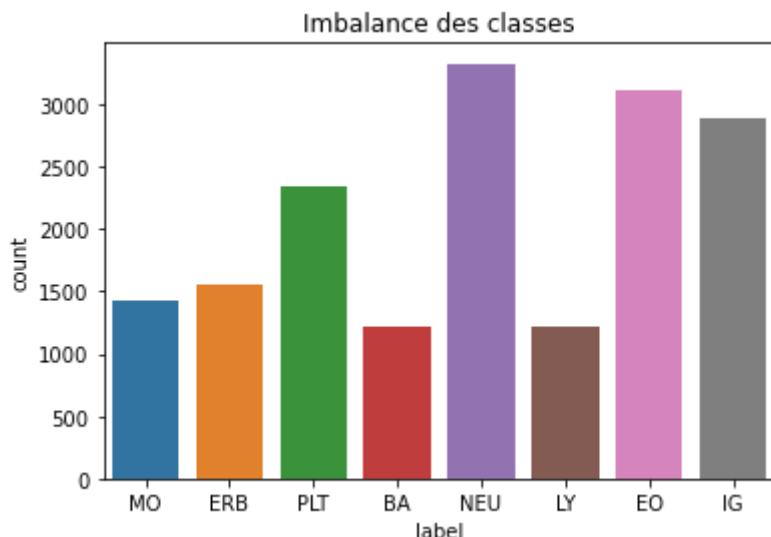


Figure 3: Distribution des classes dans le dataset de Barcelone.

En regardant les caractéristiques des cellules (Fig. 4), on peut remarquer des similitudes entre certains types de cellules : par exemple les monocytes et les granulocytes immatures ou les neutrophiles segmentés et bandes. On remarque aussi que certains éléments pourraient parasiter l'information importante, les globules rouges en fond et couleur de l'image.

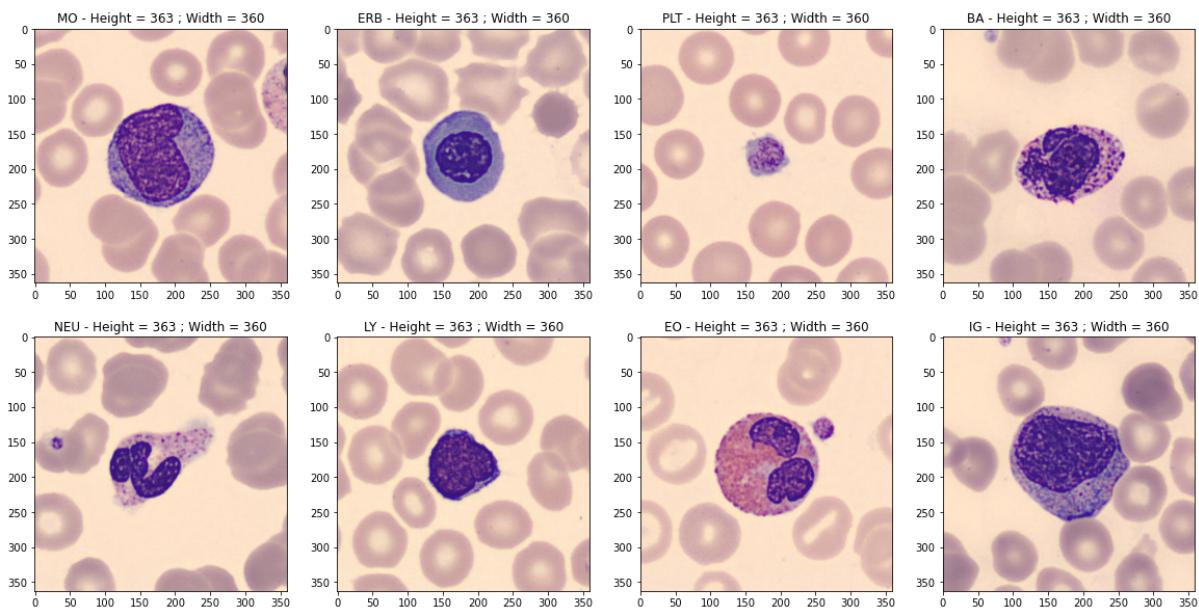


Figure 4 : Exemples des différents types cellulaires. La cellule d'intérêt présente généralement une forte coloration (bleue, violette ou rose selon le type de cellule) et se trouve au centre de l'image. Les petits objets de forme globalement arrondie situés autour sont des globules rouges.

Par une analyse de la luminosité de chaque image par classe de cellules, nous avons pu déterminer que **toutes les classes présentent des outliers**, essentiellement de **faible luminosité**, liés à la **taille du noyau** de la cellule d'intérêt, au **nombre de globules rouges en fond**, et à la **présence de cellules multiples**. De plus, les **cellules de petites tailles** comme les plaquettes et les érythroblastes ont une luminance plus haute. En utilisant des techniques de réduction de dimension telles que PCA, LDA ou UMAP (Fig. 5), nous avons confirmé que les plaquettes se distinguent bien du reste des cellules : cela pourrait s'expliquer par l'absence de noyau, qui conduit à une luminosité moyenne supérieure pour ces images.

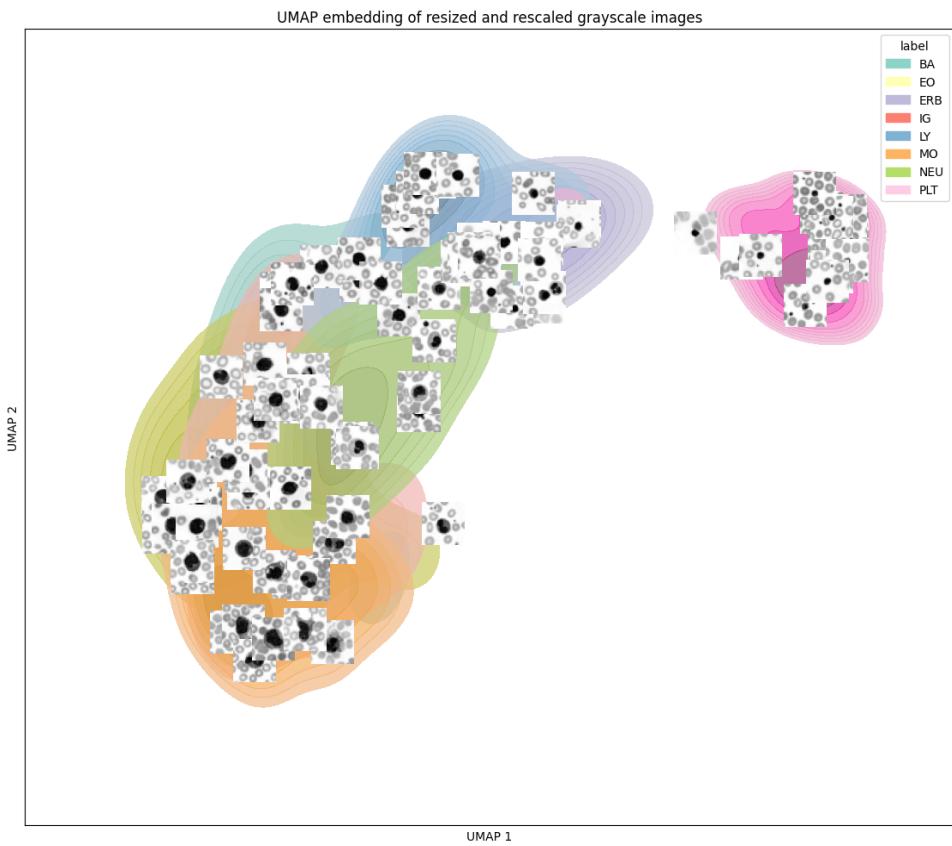


Figure 5: Réduction de dimension avec l'UMAP.

Pour pousser l'analyse, nous avons utilisé une **technique de déconvolution des couleurs** développée pour l'analyse histologique et la librairie employée utilise l'algorithme développé par G. Landini¹⁶. La déconvolution des couleurs est basée sur la loi de Beer-Lambert et peut séparer les images en trois canaux représentant l'absorbance de chaque coloration histologique : les globules rouges, les noyaux et le background. Des différences sont observées avec les plaquettes (PLT) et les érythroblastes (ERB) avec une intensité plus faible dans le canal noyau, plus haute dans le background et similaire dans le canal globules rouges, ce qui correspond bien à une taille plus petite (confirmée par une analyse de luminosité radiale des cellules moyennes, résultats non montrés dans ce rapport). Les basophiles (BA) présentent eux aussi un shift dans la canal background ce qui peut s'expliquer par leurs tailles et la présence d'agglomérats autour de ces cellules. Il n'y a donc probablement pas de biais de luminosité d'une classe de cellules à l'autre.

En termes de nettoyage, les images sont au format RGB et toutes n'ont pas la même taille. Nous avons donc dû **redimensionner les images**. Les données dont nous disposons pour les cellules saines dans le dataset de Barcelone sont de qualité et ne présentent pas de biais évident. Il faut toutefois ajouter qu'elles **ont été enregistrées avec le même analyseur, et processées dans le même laboratoire avec le même protocole** de coloration : **c'est une autre forme de biais**, qui pourrait induire un overfitting de notre modèle et **c'est pourquoi nous avons inclus d'autres datasets dans l'entraînement du modèle final**. Pour débuter, nous avons comparé plusieurs modèles en travaillant à partir du dataset de Barcelone. Après avoir sélectionné un modèle “final”, nous avons ensuite complété ce dataset avec les autres données que nous avons pu trouver.

¹⁶ "Colour deconvolution: stain unmixing in histological imaging - PubMed." 16 Jun. 2021, <https://pubmed.ncbi.nlm.nih.gov/32997742/>. Accessed 26 Sep. 2021.

Projet

Classification du problème

Avez vous utilisé d'autres métriques de performances qualitative ou quantitative) ? Si oui, détaillez.

Nous avons un **problème de classification d'images** à 8 ou 11 classes (en fonction de la complexité). Il s'agit de **computer vision**. La métrique choisie est le **F1-score pour chaque classe considérée individuellement** (le dataset étant déséquilibré, en plus de precision et recall) et l'**accuracy pour la performance globale** (toutes classes confondues, le déséquilibre n'étant toutefois pas énorme). Nous avons aussi tenté de comprendre ce que faisaient nos modèles avec Grad-CAM, qui permet d'analyser les régions de l'image auxquelles le modèle accorde de l'importance.

Choix du modèle & Optimisation

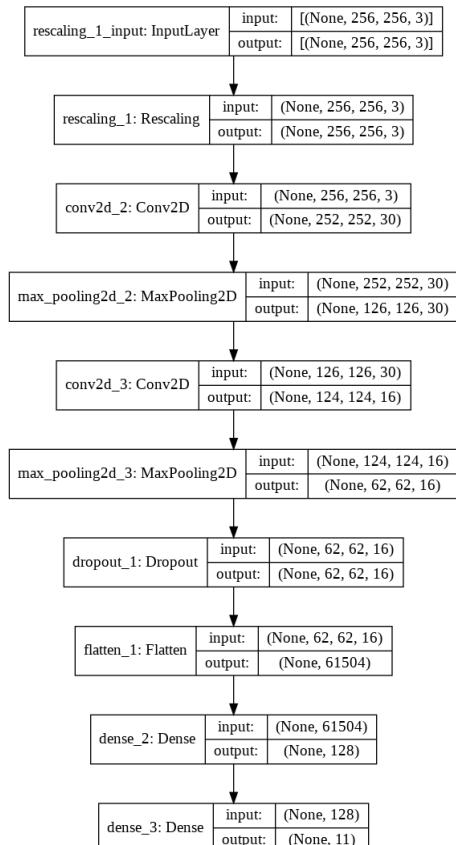
Plusieurs modèles ont été essayés au cours des premières itérations du projet. La plupart de ces approches reposent sur les Réseaux Neuronaux Convolutifs, fréquemment utilisés dans le domaine de la classification d'images et connus pour obtenir de bonnes performances. Une approche plus originale, utilisant un modèle de VisionTransformer, a également été testée.

LeNet

LeNet : baseline

Ce modèle, proposé par Yann LeCun en 1989, est l'un des tout premiers modèles de deep learning construit pour la classification d'images. Du fait de sa simplicité, il nous a servi à **établir une baseline** pour les données de Barcelone **mais aussi à nous familiariser avec les concepts de base** des CNN, Google Colab et le calcul sur GPU. **L'architecture du modèle LeNet utilisé est représentée sur la figure ci-contre, à droite.**

Le jeu de données a été divisé en trois ensembles : un jeu de test (2544 images), un jeu de validation (1730 images) et un jeu d'entraînement (12677 images). Les images ont été redimensionnées en 256*256*3 (RGB) et traitées par batchs de 32, sans augmentation de données. Ci-dessous, **la distribution des classes des images pour le jeu d'entraînement :**



EO	PLT	SNE	BNE	ERB	MO	LY	BA	MY	MMY	PMY
2352	1743	1263	1212	1162	1068	946	903	833	734	461

On compare sur le graphique ci-dessous l'accuracy globale mesurée sur le jeu d'entraînement (bleu) et sur le jeu de validation (orange) durant l'entraînement. On voit que le modèle over-fitte de manière importante les données, l'écart d'accuracy mesurée sur les deux jeux de données étant de plus en plus important au fur et à mesure de l'entraînement. **L'accuracy sur le jeu de validation plafonne aux alentours de 79%** (79,31% sur le jeu de validation, 79,83% sur le jeu de test).

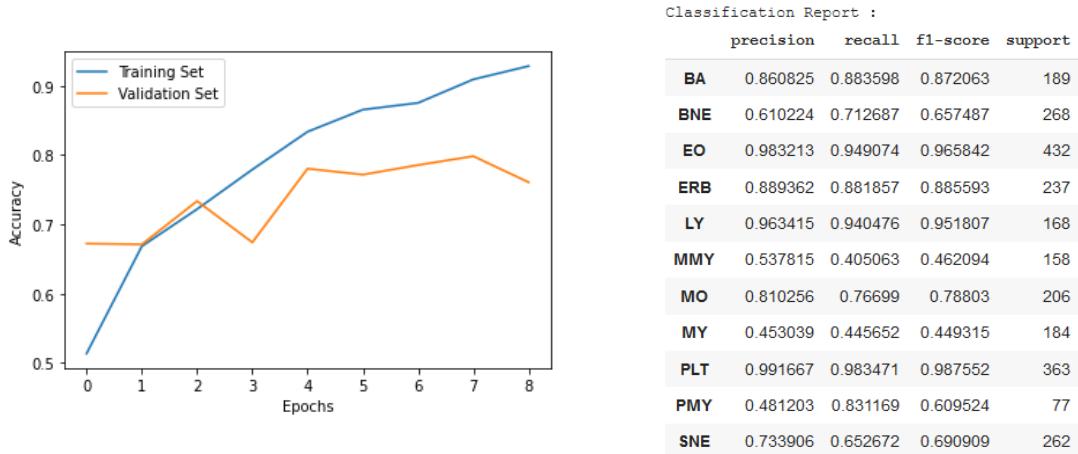


Figure 7 : Gauche : accuracy globale du modèle LeNet sur les jeux d'entraînement et de validation. Droite : rapport de classification pour le jeu de test : précision, rappel, f1-score de chaque classe. La colonne support indique le nombre d'images dans chaque classe (labels réels).

La lecture du rapport de classification (Figure 7) nous permet de mettre en évidence de très bons F1-Scores pour les éosinophiles (cellules roses, EO), sur les lymphocytes (LY) ainsi que sur les plaquettes (PLT). En revanche, le modèle est médiocre pour les MMY, PMY et MY, ainsi que pour les SNE/BNE.

Regardons maintenant la matrice de confusion (Figure 8) obtenue pour le jeu de test afin de déterminer quelles sont les confusions les plus fréquentes commises par le modèle : sur l'axe vertical nous représentons les labels réels des images ; les labels prédits se trouvent quant à eux sur l'axe horizontal. Chaque case de la matrice contient le pourcentage d'images du label X classées dans la catégorie Y : la somme des valeurs d'une ligne vaut 1.

Ainsi, on peut lire sur la première ligne que 88,4% des images de BA ont été classées BA par le modèle. La couleur de chaque case est représentative du pourcentage d'images qui s'y trouve : **un modèle qui classe bien les images implique une matrice de confusion dont seule la diagonale est sombre**.

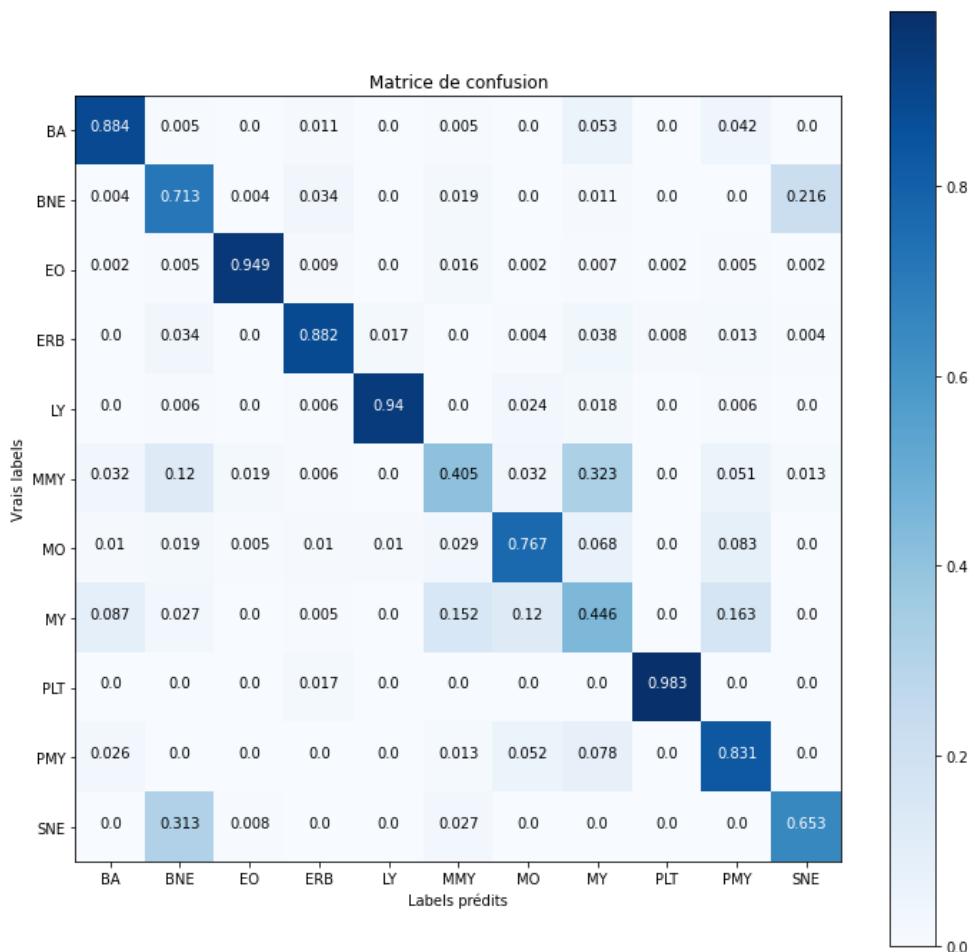


Figure 8 : Matrice de confusion sur le jeu de test. Modèle LeNet, 11 classes.

Ce n'est clairement pas le cas avec le modèle LeNet. Sans entrer dans les détails (nous le ferons plus loin), soulignons les principales erreurs du modèle :

- **confusion entre BNE et SNE** (31,3% des SNE sont classés comme BNE, 21,6% des BNE sont classés comme SNE);
- **confusion entre MY et MMY, et dans une moindre mesure avec les PMY (bien mieux détectés) et les monocytes (MO).**

Il apparaît que **les MY sont les plus difficiles à classifier**, dans le sens où on trouve un pourcentage important de ces images dans pas moins de 5 classes.

LeNet : considérations sur le label des cellules

À titre d'expérience et de comparaison, **entraînons maintenant le modèle sur 8 classes**, comme c'est le cas dans l'article accompagnant le dataset Barcelone Normal : pour cela, nous allons regrouper PMY, MY et MMY dans une même classe (IG pour Immature Granulocyte) et SNE et BNE dans la classe NEU (Neutrophil).

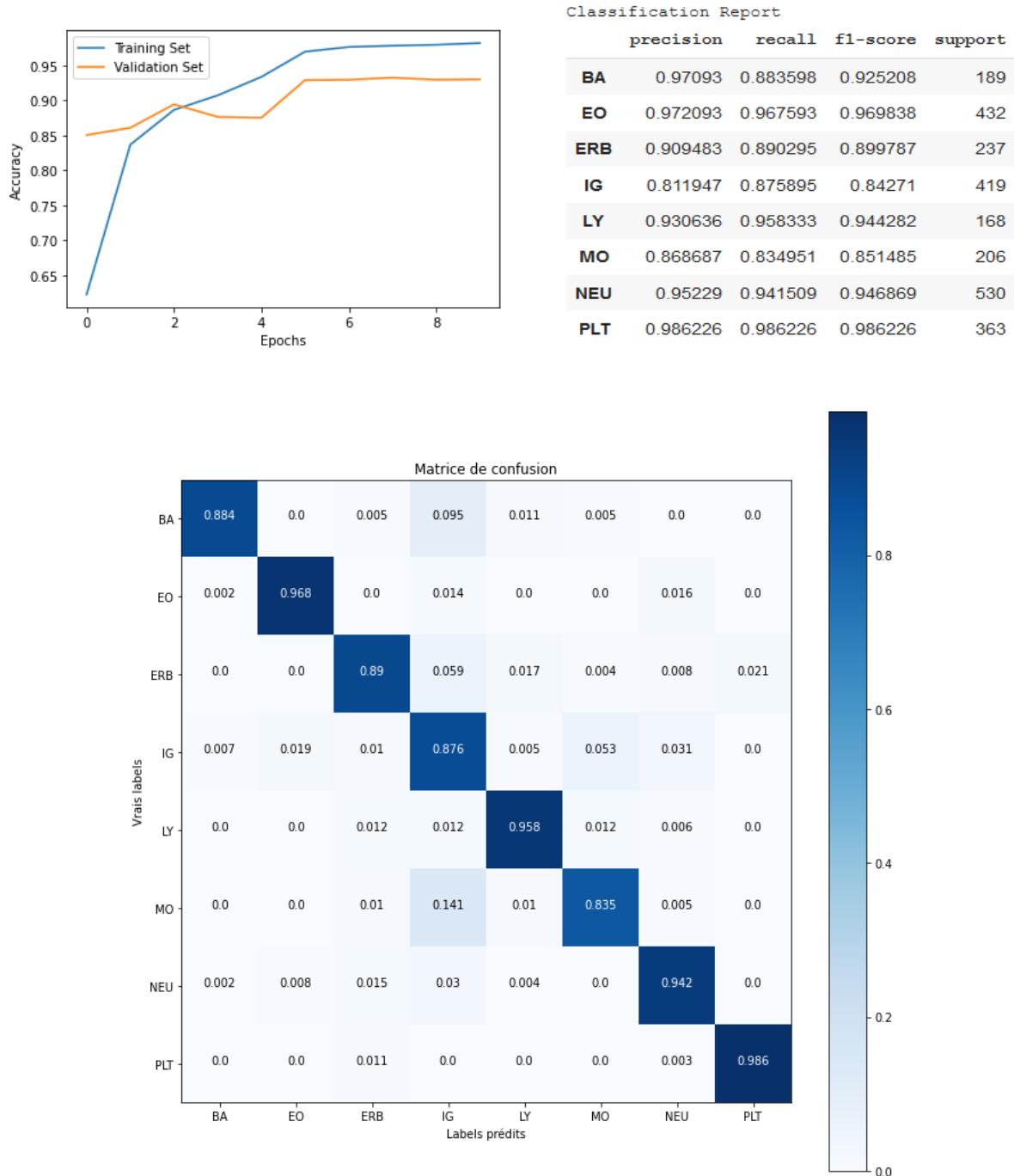


Figure 9 : Modèle LeNet, 8 classes. Résultats pour le jeu de test.

Le résultat de cette expérience est une **amélioration drastique des performances apparentes du modèle LeNet** (Figure 9), puisque nous obtenons une accuracy globale de 92 ~ 93% sur les jeux de test et de validation (soit plus de 10% d'amélioration par rapport au cas à 11 classes, **et pourtant avec exactement les mêmes données** dans chacun des trois jeux). Le résultat est légèrement moins bon que dans l'article Le surapprentissage apparaît nettement réduit (99% sur le jeu d'entraînement) et le modèle classe légèrement mieux d'autres types de cellules (BA et MO).

De ceci, il y a plusieurs commentaires à faire :

- nous passons d'un F1-Score inférieur à 70% à un score supérieur à 94% pour l'ensemble des neutrophiles. L'augmentation du F1-Score pour les IG (MY, MMY et PMY regroupés) est tout aussi spectaculaire, et s'accompagne d'une amélioration importante pour les MO. **Cela semble aller dans le sens d'une confusion entre d'une part les sous-classes de neutrophiles : SNE/BNE et d'autre part entre les sous-classes d'immatures (MY, MMY et PMY)**. Cette confusion peut provenir de **deux causes : le modèle qui peine à distinguer les cellules ou des erreurs lors du labelling des images**. Nous avons en tout cas **deux "super-classes" (IG et NEU) qu'il est difficile de subdiviser** ;
- le choix des classes à décrire correctement et le labelling en amont du modèle peuvent donc avoir un impact très important sur le résultat final et **conduire à une augmentation / à une diminution artificielle des performances d'un modèle**. Il est ainsi très simple de "mettre la poussière sous le tapis" sans s'en rendre compte : il convient donc d'être prudent et de **réfléchir au rôle de chaque sous-classe de cellule** pour savoir si oui ou non il est pertinent de considérer deux classes ou bien une seule.

En réalité, les neutrophiles segmentés (SNE), les éosinophiles et les basophiles font partie d'une catégorie de globules blancs appelés "granulocytes", selon la capacité qu'elles ont à absorber certains types de colorants (neutres, acides ou basiques). **Chacune de ces sous-catégories a sa fonction spécifique dans l'organisme** : les éosinophiles sont par exemple impliqués dans la lutte contre les parasites et dans le mécanisme des allergies. Il est donc **pertinent de chercher à les décrire correctement**.

Ces cellules sont des cellules arrivées à maturité, qui circulent dans le sang et dont la concentration change lorsqu'il y a une pathologie. Avant d'arriver à cette maturité, elles passent par plusieurs stades d'évolution, comme on peut le voir sur les Figures 1 et 10 : promyélocyte (PMY), myélocyte (MY) puis métamyélocyte (MMY) et enfin granulocyte de bande (dont les BNE, qui sont des neutrophiles de bande). Cette évolution se produisant de façon continue, il est possible de trouver des cellules "entre deux stades", ce qui complique la tâche de labelling et de classification.

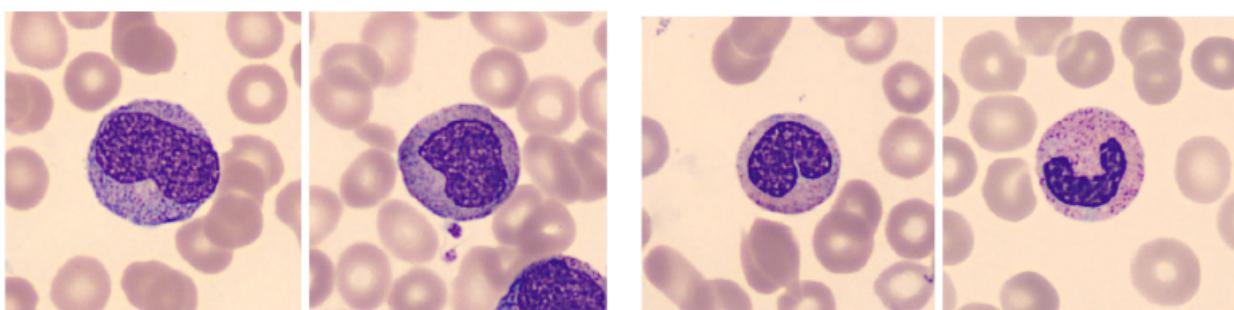


Figure 10 - De gauche à droite, un promyélocyte, deux métamyélocytes et un BNE.

Leur présence dans le sang est révélatrice d'une inflammation ou d'une infection, comme si l'organisme ayant besoin de défenseurs supplémentaires envoyait les "bleus" en cours de formation aider leurs homologues matures. Il est donc **important de distinguer les cellules immatures des cellules matures** (et donc d'avoir une classe "IG") mais il est peut-être possible de se passer de distinguer MY/MMY/PMY/BNE. Ajoutons qu'aucun des datasets à notre disposition ne distingue les éosinophiles et les basophiles immatures des éosinophiles et basophiles matures (peut-être parce que la présence de ces cellules matures dans le sang est déjà rare en comparaison d'autres classes).

Nous avons cependant **choisi de distinguer toutes les sous-classes** (donc 11 classes au total) : les modèles fondés sur les CNN sont capables de classer correctement dans un nombre bien plus important de classes (cf. le dataset ImageNet), nous préférons donc garder du détail dans notre modélisation. Nous avons aussi décidé **d'afficher les probabilités d'appartenance aux classes** plutôt que la seule classe dominante afin de **fournir une prédiction moins "exclusive"** et de permettre au spécialiste utilisant le modèle de trancher.

LeNet : effet d'une augmentation de données, nouvelle baseline

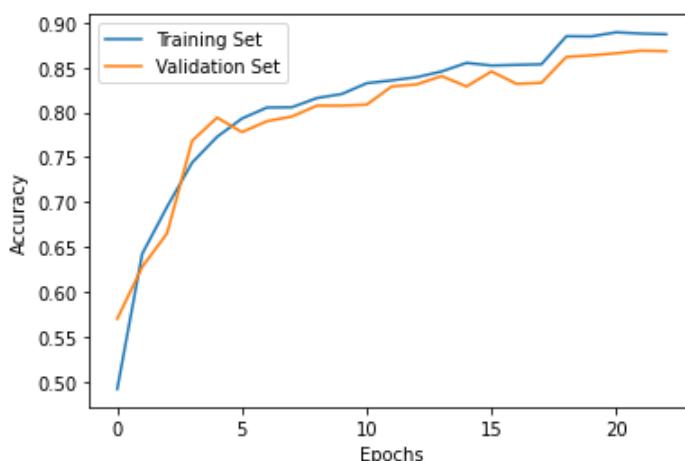
Le surapprentissage du modèle est important (17% d'écart en accuracy globale entre le jeu d'entraînement et le jeu de validation) et doit être réduit. La première stratégie envisagée pour le réduire - dans ce modèle mais aussi dans les suivants - est d'implémenter de l'augmentation de donnée : en transformant chaque image du jeu d'entraînement à chaque epoch selon des règles prédéfinies, on s'assure que le modèle apprend à chaque fois sur de nouvelles images. On espère ainsi accroître sa capacité à généraliser, ce qui doit se traduire par une accuracy globale sur les jeux de validation et de test :

- plus élevée,
- plus proche de celle mesurée pour le jeu d'entraînement

On utilise pour cela la classe `ImageDataGenerator` de Keras pour introduire des transformations simples sur les images du jeu d'entraînement : rotation, symétrie axiale.

En utilisant le même modèle que précédemment (LeNet, 11 classes) et les mêmes jeux de données, on mesure un fort impact bénéfique de l'augmentation de données sur la performance du modèle :

- **l'accuracy globale** mesurée sur les jeux de validation et de **test bondit** de 79% à **85~86%**
- l'accuracy globale mesurée sur le jeu d'entraînement diminue à **86~87%** : le **surapprentissage a été fortement réduit** (on passe de 17% d'écart à ~2% seulement, et avec une augmentation de données très basique).



Regardons plus en détail les effets de cette augmentation en comparant les métriques obtenues pour chacune des 11 classes modélisées par LeNet (Figure 11):

Classification Report - Train					Classification Report - Test				
	precision	recall	f1-score	support		precision	recall	f1-score	support
BA	0.925078	0.984496	0.953863	903	BA	0.914141	0.957672	0.935401	189
BNE	0.775269	0.594884	0.673203	1212	BNE	0.815385	0.593284	0.686825	268
EO	0.997021	0.996173	0.996597	2352	EO	0.99536	0.993056	0.994206	432
ERB	0.982159	0.947504	0.96452	1162	ERB	0.969432	0.936709	0.95279	237
LY	0.972603	0.975687	0.974142	946	LY	0.936782	0.970238	0.953216	168
MMY	0.72404	0.69346	0.70842	734	MMY	0.608434	0.639241	0.623457	158
MO	0.919192	0.937266	0.928141	1068	MO	0.885714	0.902913	0.894231	206
MY	0.715356	0.687875	0.701346	833	MY	0.698718	0.592391	0.641176	184
PLT	0.997702	0.996558	0.99713	1743	PLT	0.994521	1	0.997253	363
PMY	0.789247	0.796095	0.792657	461	PMY	0.684783	0.818182	0.745562	77
SNE	0.70836	0.87886	0.784452	1263	SNE	0.722561	0.90458	0.80339	262

Figure 11 : Comparaison des métriques de précision, rappel et F1-Score pour chacune des 11 classes modélisées par LeNet. Gauche : résultats mesurés pour le jeu d'entraînement. Droite : résultats mesurés pour le jeu de test.

Il apparaît que le surapprentissage est le plus important pour le trio MY/MMY/PMY (entre 5 et 8% d'écart entre les F1-scores mesurés sur les jeux de test et d'entraînement). Le problème a disparu pour certaines classes : PLT (plaquettes), EO (éosinophiles), SNE et BNE. Remarquons que **les MY/MMY (et dans une moindre mesure les BNE) sont les types de cellules que le modèle a le plus de mal à classer correctement** : le score de rappel, mesurant la capacité du modèle à détecter ces images, est mauvais (seulement 60% de ces images sont classées correctement). S'il détecte mal les BNE, le modèle est en revanche précis lorsqu'il classe des images comme BNE (précision = 81,5%).

L'augmentation de données semble en tout cas être une bonne stratégie pour réduire le surapprentissage mais aussi pour accroître les performances du modèle.

Ces résultats constituent la nouvelle baseline à améliorer :

- accuracy globale de 87% sur le dataset Normal Barcelone
- F1-scores de la Figure 11.
- surapprentissage à 2%

Afin de battre la baseline établie par LeNet, nous avons décidé de tester des modèles plus complexes :

- EfficientNet,
- VGG16 et VGG19,
- Xception
- VisionTransformer

Ces expérimentations sont détaillées dans la suite de ce document.

EfficientNet

EfficientNet est un CNN à l'architecture complexe bien plus complexe que LeNet. Ce modèle a été développé dans le but de fournir de **bonnes performances tout en réduisant le nombre de paramètres à entraîner** nécessaires pour l'obtenir. Il nous a donc paru intéressant d'essayer ce modèle.

EfficientNet - Data

Tout d'abord, et dans le but de pouvoir comparer ce modèle avec LeNet, VGG et Xception :

- les **jeux de données utilisés sont les mêmes que pour LeNet** : Normal Barcelone, avec le même découpage en trois ensembles : train/valid/test;
- on utilise la **même augmentation de données que pour LeNet** : rotation, symétrie axiale ;
- 11 classes sont modélisées.

Il est nécessaire d'utiliser la fonction de preprocessing spécifique à EfficientNet : c'est la couche d'entrée du modèle qui se charge de normaliser les pixels des images. Une "couche" de preprocessing existe dans la librairie Keras, mais elle n'a aucun effet. **Les modèles EfficientNet requièrent comme entrée des images sous forme de Tensor, dont la valeur de chaque pixel est comprise entre 0 et 255.**

EfficientNet - Modèle et résultats

L'approche choisie est celle de l'apprentissage par transfert (**Transfer Learning**) : on utilise un modèle préalablement entraîné sur un jeu de données A pour **classer les images d'un jeu de données B dont les classes sont différentes de celles de A**. Pour cela, on importe les poids optimisés pour le jeu de données A et on remplace les dernières couches du modèle (top layers) par des couches non-entraînées et adaptées au problème du jeu de données B (bloc de classification). On entraîne seulement ces nouvelles couches pour conserver l'expérience acquise par le modèle sur le jeu A : on "gèle" donc les couches n'appartenant pas au bloc de classification (voir schéma de principe ci-dessous).

Ici, le modèle est EfficientNet. Le jeu de données A est ImageNet, une base de données d'images contenant plusieurs millions d'échantillons regroupés en 1000 classes. En pratique, cela revient à **transférer les connaissances apprises par EfficientNet sur le dataset ImageNet** à notre modèle, puis à **mobiliser ces connaissances** (i.e. les features extraites) **sur un problème "légèrement" différent** : classer des images de cellules sanguines. L'un des avantages immédiats de cette approche est un gain de temps non négligeable lors de l'entraînement du modèle puisque seules les couches du bloc de classification font l'objet d'une optimisation.

Il est possible grâce à la librairie TensorFlow.Keras d'importer les poids d'EfficientNet optimisés pour ce jeu de données. Ensuite, on ajoute en sortie d'EfficientNet un bloc de classification essentiellement constitué de couches Denses (couche linéaire avec activation par la fonction ReLU par exemple). Les couches d'EfficientNet sont ensuite gelées, pour que seul le bloc de classification soit entraîné sur Barcelone Normal :

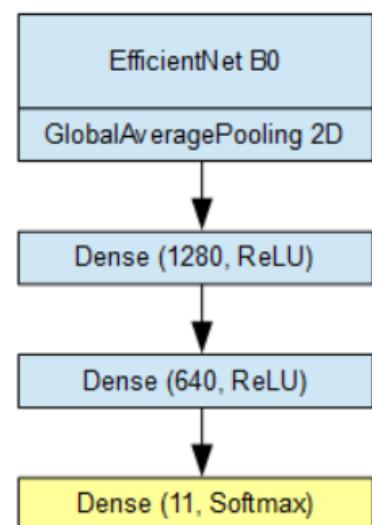
- EfficientNet extrait des caractéristiques (features) optimales pour le type d'images sur lesquelles il a été entraîné (ici ImageNet) ;
- le bloc de classification calcule des probabilités d'appartenance à 11 classes en utilisant ces caractéristiques et en considérant les labels de chaque image du jeu d'entraînement comme référence.

EfficientNet est en réalité une famille de modèles (les variantes étant nommées de B0 à B7) : le choix d'un modèle dans cette famille dépend de plusieurs contraintes, notamment la résolution des images. Nos images sont de format 360*360 pixels en RGB : nous sommes donc limités aux architectures d'EfficientNet conçues pour des images de résolution inférieure ou égale à 360*360. **Deux modèles ont donc été testés : EfficientNetB0 (Input Shape : (224, 224, 3)) et EfficientNetB3 (Input Shape : (300, 300, 3)).**

Structure et résultats du modèle B0 :

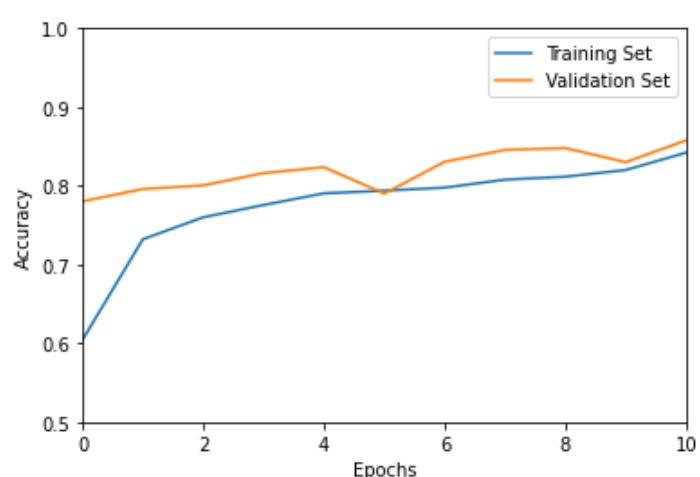
Le modèle “B0” est constitué :

- **du bloc d'extraction de features (base_model)** : il s'agit du modèle EfficientNetB0, privé de ses dernières couches (include_top = False) et entraîné sur la base de données ImageNet (weights = ‘imagenet’). Les poids importés sont gelés : ces couches ne seront pas entraînées ;
- **du bloc de classification** : il est constitué de deux couches Dense (units = 1280 et 640 neurones respectivement) activées par la fonction ReLU, et d'une ultime couche Dense (units = 11) activée par la fonction softmax. Cette dernière couche calcule les probabilités d'appartenance des images à chacune des 11 classes.¹⁷



On choisit **Adam** comme optimiseur, avec un **learning rate initial de 1e-3**. Ce dernier va varier au cours de l'entraînement grâce à un callback de type ReduceLROnPlateau.

Au bout de 11 epochs, le modèle obtient une accuracy globale de 0.8653 et de **0.8683** sur les jeux de validation et de **test** respectivement, pour une accuracy globale de **0.8712 sur le jeu d'entraînement**. Le modèle obtient la **même accuracy que LeNet, mais sans surapprentissage** ce qui est une amélioration.



¹⁷ D'autres structures pour le bloc de classification ont été testées, notamment un ensemble de 4 couches Denses (units = 2048, 1024, 512 et 11) avec ou sans couches de Dropout entre elles. Les performances en termes de métriques sont similaires, avec de faibles fluctuations sur les F1-score de chaque classe.

Pour pousser plus loin la comparaison avec LeNet, on peut comparer les rapports de classification obtenus avec ces deux modèles pour le jeu de test (gauche : B0 ; droite : LeNet) :

Classification Report - Test				
	precision	recall	f1-score	support
BA	0.898058	0.978836	0.936709	189
BNE	0.767347	0.701493	0.732943	268
EO	0.983529	0.967593	0.975496	432
ERB	0.969828	0.949367	0.959488	237
LY	0.942529	0.97619	0.959064	168
MMY	0.618421	0.594937	0.606452	158
MO	0.849315	0.902913	0.875294	206
MY	0.70122	0.625	0.66092	184
PLT	0.997214	0.986226	0.99169	363
PMY	0.625	0.844156	0.718232	77
SNE	0.799242	0.805344	0.802281	262

Classification Report - Test				
	precision	recall	f1-score	support
BA	0.914141	0.957672	0.935401	189
BNE	0.815385	0.593284	0.686825	268
EO	0.99536	0.993056	0.994206	432
ERB	0.969432	0.936709	0.95279	237
LY	0.936782	0.970238	0.953216	168
MMY	0.608434	0.639241	0.623457	158
MO	0.885714	0.902913	0.894231	206
MY	0.698718	0.592391	0.641176	184
PLT	0.994521	1	0.997253	363
PMY	0.684783	0.818182	0.745562	77
SNE	0.722561	0.90458	0.80339	262

Figure 12 :

comparaison des rapports de classification, jeu de test. Gauche : B0. Droite : LeNet.

On voit que **les performances sont sensiblement les mêmes** : si EfficientNetB0 semble mieux reconnaître les BNE et les MY (meilleur rappel, deux points faibles majeurs de LeNet), il est légèrement moins bon que LeNet pour EO, MO, MMY et PMY. Nous reviendrons ultérieurement (voir chapitre sur VGG-19) sur les résultats obtenus avec Grad-CAM pour tous nos modèles.

Quelles sont les principales erreurs de classement commises par le modèle ? La matrice de confusion va nous les indiquer :

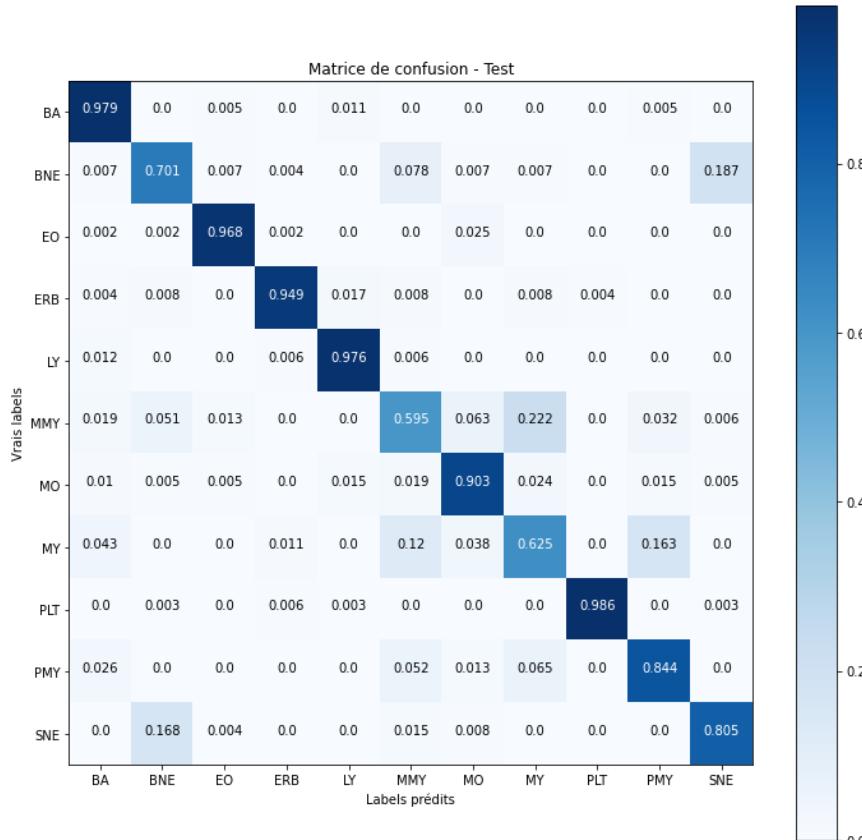


Figure 13 : Matrice de confusion pour le modèle B0 (jeu de test)

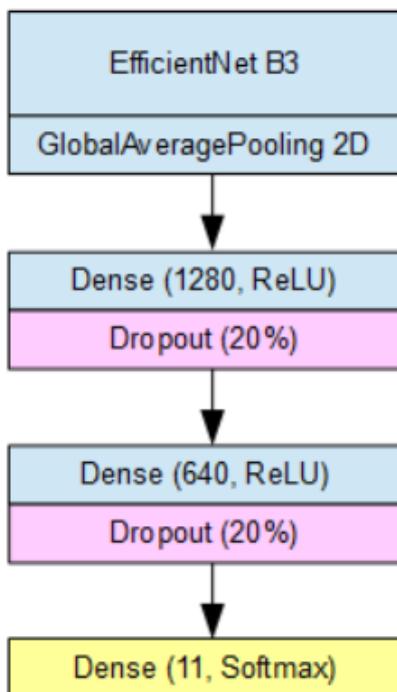
On retrouve essentiellement les mêmes problèmes qu'avec LeNet : confusion d'une part entre BNE et SNE¹⁸, et d'autre part confusion entre les groupes de cellules immatures PMY/MY/MMY. Notons que deux modèles différents (LeNet et EfficientNetB0) souffrent de ce même problème.

Structure et résultats du modèle B3 :

B0 conduit à des performances similaires à LeNet. **Le modèle B3 de la famille EfficientNet** est conçu pour travailler avec des images de meilleure résolution. Nous espérons ainsi dépasser la baseline représentée par LeNet.

Notre **modèle reposant sur “B3”** est constitué (schéma ci-dessous) :

- **du bloc d'extraction de features** (base_model) : il s'agit du modèle EfficientNetB3, privé de ses dernières couches (include_top = False) et entraîné sur la base de données ImageNet (weights = ‘imagenet’). Les poids importés sont gelés : ces couches ne seront pas entraînées ;
- **du bloc de classification** : il est constitué de deux **couches Dense** (units = **1280** et **640** neurones respectivement) activées par la fonction **ReLU**. Chacune de ces couches est suivie d'une couche de **Dropout** afin de réduire le surapprentissage potentiel. Enfin, la couche de sortie est une couche **Dense** (units = 11) activée par la fonction **Softmax**. Elle calcule les probabilités d'appartenance des images à chacune des 11 classes.



On choisit à nouveau **Adam comme optimiseur**, avec un **learning rate initial de 1e-3**. Ce dernier va varier au cours de l'entraînement grâce à un callback de type ReduceLROnPlateau.

¹⁸ ... et dans une moindre mesure avec les MMY, problème nouveau par rapport à LeNet. Les MMY constituent le stade de la vie cellulaire immédiatement antérieur à celui de BNE dans le cas d'un neutrophile, cette confusion a donc possiblement une raison biologique.

Assez curieusement, on obtient (Figure 14) une **performance en accuracy globale légèrement inférieure à celle obtenue avec le modèle B0** : 85,84% en validation, **85,22% sur le jeu de test**, pour 86,55% sur le jeu d'entraînement. Voir le rapport de classification ci-dessous pour avoir le détail pour chaque classe. Un **léger surapprentissage** apparaît, en dépit de la présence des couches de Dropout. Il reste inférieur à celui obtenu pour LeNet.

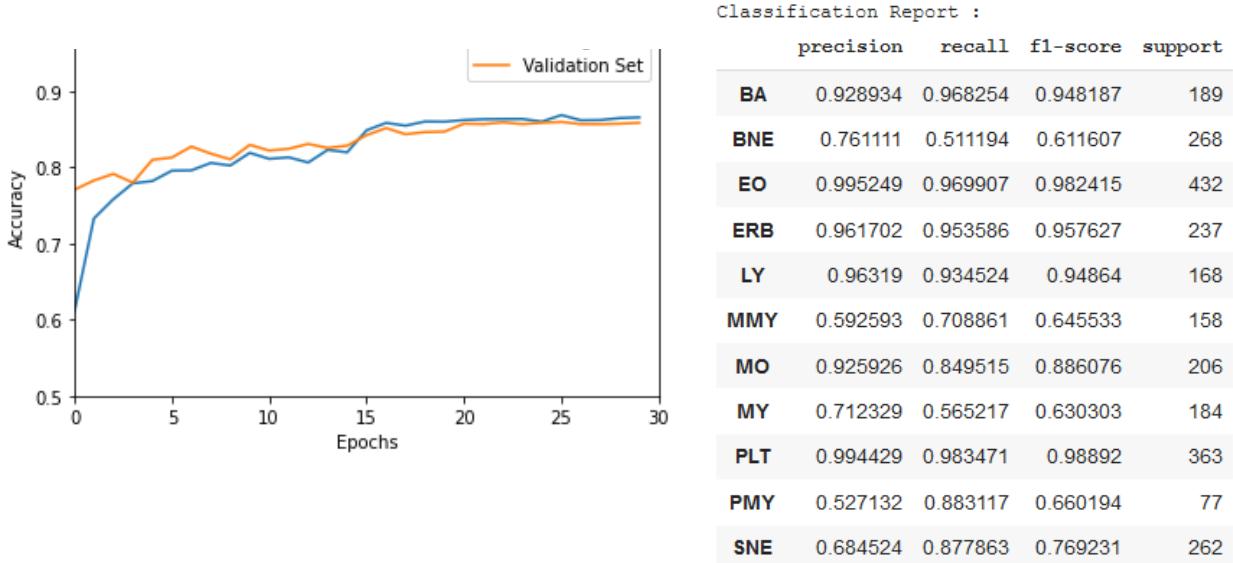


Figure 14 : Résultats obtenus avec le modèle B3 (transfer learning).

Ce modèle comporte 390 couches, dont 385 constituent le modèle EfficientNetB3 proprement dit (contre environ 240 pour le modèle EfficientNetB0) : les couches d'EfficientNetB3 les plus éloignées de l'input sont hautement spécialisées sur le dataset ImageNet, qui comporte plusieurs millions d'images réparties en 1000 classes (animaux, fleurs, véhicules etc...). Nous importons **les poids optimisés pour ce dataset, mais ils ne sont pas optimisés pour notre jeu de données** qui ne contient “que” 11 classes de cellules sanguines et ne sont donc peut-être pas adaptés au problème. La question s'est alors posée **d'entraîner la totalité d'EfficientNetB3** pour en tirer pleinement profit.

Modèle B3 - Fully trained

Cette fois-ci, on choisit ‘**weights = None**’ dans la fonction d'appel EfficientNetB3. Les paramètres de l'ensemble des couches du modèle seront optimisés spécifiquement pour le dataset Normal Barcelone.

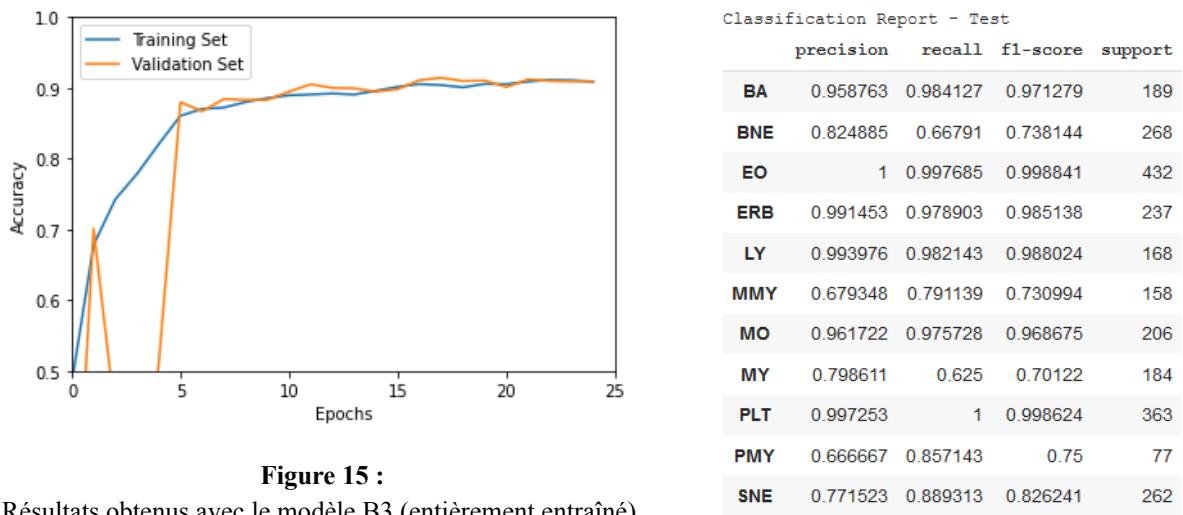


Figure 15 :

Résultats obtenus avec le modèle B3 (entièlement entraîné).

Le gain de performance est majeur : 90,92% en accuracy globale sur le jeu de validation, 90,90% sur le jeu d’entraînement (25 epochs) et 90,25% sur le jeu de test. **Le surapprentissage est inexistant.** On remarque (Figure 15, droite) que le modèle fait bien mieux que LeNet, et ce pour toutes les classes, en particulier celles qui étaient difficiles à classer (BNE, MMY, MO, MY, PMY, SNE). Plusieurs classes franchissent le seuil des 96%, mais la performance sur les neutrophiles reste très moyenne.

Xception

De la même manière que pour EfficientNet, on utilise Xception et les poids pré-calculés sur la base de données **ImageNet** pour du **transfer learning**.

Xception - Data

À des fins de **comparaison avec les modèles précédents** :

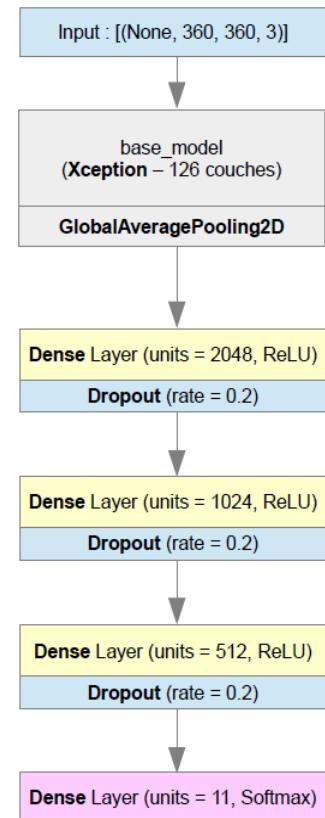
- les jeux de données utilisés sont les mêmes que pour LeNet et EfficientNet : Normal Barcelone, avec le même découpage en trois ensembles : train/valid/test;
- on utilise la même augmentation de données que pour LeNet et EfficientNet : rotation, symétrie axiale ;
- 11 classes sont modélisées.

On doit penser à utiliser la fonction de preprocessing spécifique à Xception (la valeur de chaque pixel de l'image en entrée du modèle est normalisée entre -1 et 1). On l'indique dans l'argument `preprocessing_function` de la méthode `flow_from_dataframe`.

Xception - Modèle

La structure du modèle est présentée ci-dessous (voir schéma ci-contre pour un schéma, et au bas de la page pour un extrait du code) :

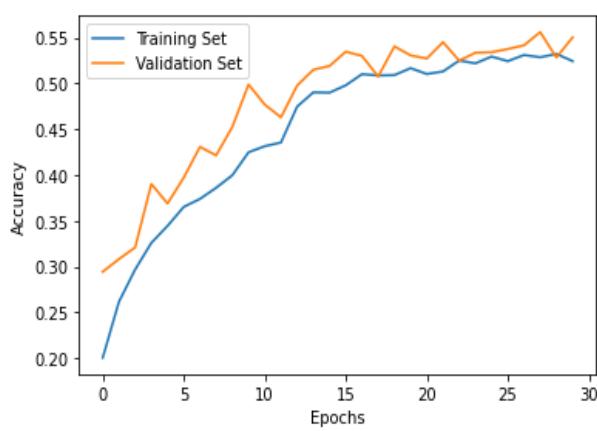
- le bloc “**base_model**” contenant l’ensemble des couches de **Xception**. On importe les poids calculés sur ImageNet puis on gèle leurs valeurs de manière à ce qu’ils ne soient pas entraînés;
- ce bloc est connecté à un **bloc de classification** : on empile trois **sous-blocs** constitués chacun d’une couche **Dense** et d’une couche de **Dropout**. On choisit **ReLU** comme fonction d’activation des couches Dense;
- la **dernière couche est une couche Dense**, dont la sortie est activée par la fonction **softmax**, permettant de retourner des probabilités d’appartenance à chacune des 11 classes modélisées.



On choisit comme optimiseur l'algorithme **Adam** avec un taux d'apprentissage (learning rate) initial de **1e-4**. Le learning rate est **ajusté durant l'entraînement** grâce à un callback de type ReduceLROnPlateau, lorsque la fonction de coût mesurée sur le jeu de validation (**val_loss**) ne diminue plus.

Xception - Résultats

Le calcul est fait sous GPU (Tesla V100, Google Colab). L'entraînement du modèle a duré **31 epochs** avant interruption par EarlyStopping faute de parvenir à réduire davantage val_loss. On affiche ci-dessous l'évolution de l'accuracy globale au cours de l'entraînement (accuracy mesurée sur le jeu de test en bleu ; accuracy mesurée sur le jeu de validation en orange). La **performance est médiocre** : environ 54% sur le jeu de validation, **57 à 58% sur le jeu de test**, mais le **surapprentissage est inexistant**.



Classification Report :

	precision	recall	f1-score	support
BA	0.533898	0.666667	0.592941	189
BNE	0.456693	0.432836	0.444444	268
EO	0.578406	0.520833	0.548112	432
ERB	0.821429	0.679325	0.743649	237
LY	0.662791	0.678571	0.670588	168
MMY	0.222222	0.113924	0.150628	158
MO	0.376471	0.621359	0.468864	206
MY	0.357143	0.108696	0.166667	184
PLT	0.97019	0.986226	0.978142	363
PMY	0.367742	0.74026	0.491379	77
SNE	0.506757	0.572519	0.537634	262

Figure 16 : Gauche : évolution de l'accuracy globale du modèle au cours de l'entraînement. Droite : rapport de classification détaillé, métriques mesurées sur le jeu de test.

Le rapport de classification (Figure 16) nous montre que les plaquettes sont bien modélisées. La matrice de confusion fait apparaître des confusions importantes entre plusieurs types de cellules : **BNE** et **SNE** (nous avons expliqué plus haut pourquoi cette confusion était compréhensible), **EO** et **MO** (celle-ci est difficilement pardonnable, ces deux types de cellules étant différenciables à l'œil nu). Les images de **MY** et **MMY** sont réparties de manière pratiquement homogène entre plusieurs classes, signe que ce modèle ne parvient pas à les gérer.



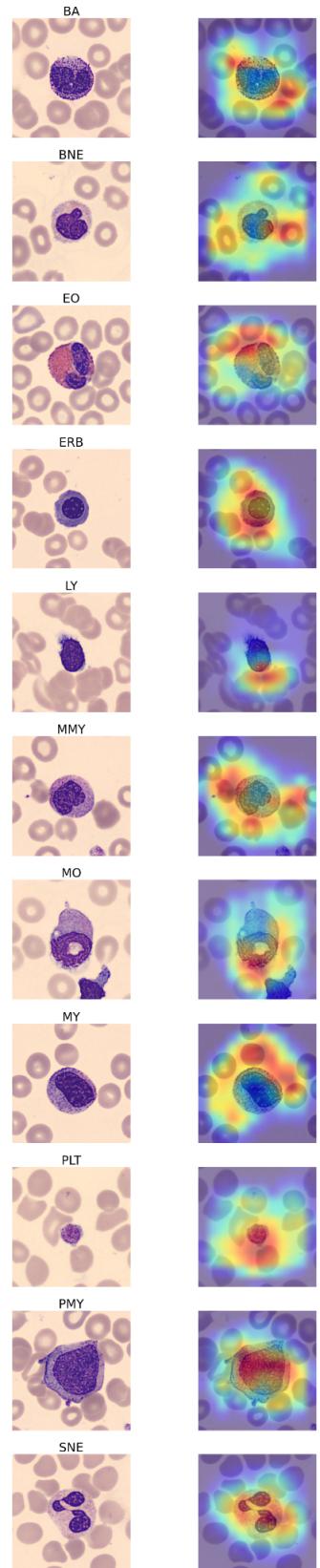
Une aussi piètre performance interroge : pourquoi un tel échec ?

Afin de mieux comprendre comment le modèle prend sa décision, il nous faut **rentrer plus en détail dans l'analyse de ce qu'il fait**. Lorsque nous identifions un objet, nous recherchons des caractéristiques de cet objet qui se raccrochent à l'idée que nous nous en faisons¹⁹. L'objectif ici est de savoir quelle idée se fait notre modèle de chacune des 11 classes de cellules : le CNN va extraire plusieurs caractéristiques de l'image (c'est le rôle du bloc Xception) : ces caractéristiques sont ensuite transmises au bloc de classification qui va calculer des probabilités d'appartenance à chaque classe, en fonction de ce qu'il aura appris à leur sujet durant l'entraînement.

La méthode **Grad-CAM permet de mettre en évidence les régions de l'image jugées pertinentes par le modèle**, et ce pour une classe donnée. Le résultat est une carte thermique qui colore en rouge/jaune ce qui est important pour le modèle : cette carte thermique sera différente selon que, pour une image donnée, on demande au modèle de rechercher les caractéristiques d'un EO ou d'un BNE. La **figure 14 ci-contre** représente une carte thermique typique de chaque type de cellule (image réelle à gauche, Grad-CAM à droite). **On peut voir que la région d'attention est rarement centrée sur la cellule d'intérêt** : le modèle s'intéresse souvent à ce qu'il y a dans son voisinage immédiat. Ainsi, **le modèle n'apprend pas des informations pertinentes** au sujet des images qu'on lui donne (comme la forme de la cellule, son noyau, sa couleur) : **le problème vient donc sans doute de l'extraction de features**, donc de Xception.

Cela ne signifie absolument pas que Xception est un mauvais modèle, mais plutôt que **nous l'utilisons mal : ce modèle possède plus de 120 couches**, qui ont été entraînées sur la base de données ImageNet. Plus les couches de convolutions sont profondes, plus elles recherchent des caractéristiques subtiles dans les images du jeu d'entraînement : elles se spécialisent. Or, **ImageNet n'est pas une base de données spécialisée dans les cellules sanguines** : on y trouve des images d'animaux, de plantes (1000 classes)... Ceci, couplé à la profondeur du modèle (beaucoup de couches sont spécialisées) ainsi qu'au fait que nous avons gelé ses paramètres (ils ne sont donc pas optimisés pour les cellules) peut expliquer la piètre performance présentée ici. Il faudra dégeler Xception et le réentraîner pour améliorer la capacité du modèle à "comprendre" nos images de cellules. Or, cela prend du temps et il faut beaucoup de données.

Une alternative consiste à procéder en deux étapes : d'abord entraîner seulement le bloc de classification, puis réentraîner le modèle en ne dégelant que les dernières couches de Xception, celles qui sont "au contact" du bloc de classification. C'est le principe du **fine tuning, que nous appliquerons à nos modèles les plus performants**. Dans la mesure où d'autres modèles (VGG, EfficientNet) donnaient de meilleurs résultats pour la même approche (pas de fine tuning, poids optimisés pour ImageNet et transfer learning), **nous avons rapidement abandonné la piste Xception**.



¹⁹ Par exemple, une pomme est un fruit globalement rond, d'une dizaine de centimètres de diamètre, avec une petite tige au sommet et une couleur pouvant aller du vert au rouge selon la variété.

VGG-19

VGG est la dernière famille de modèles que nous avons explorée. Nous allons détailler dans cette section un modèle de transfer learning de VGG19. Le cas de VGG16, connecté à un SVM, sera présenté plus loin. VGG est un modèle développé à l'Université d'Oxford au début des années 2010, connu pour avoir réalisé une précision élevée sur le dataset ImageNet dont nous avons déjà parlé. Nous allons exploiter ce modèle pour classer les images du dataset Barcelone Normal dans le cadre de l'apprentissage par transfert.

VGG-19 - Data

Par souci de comparaison avec les modèles présentés plus haut, nous allons travailler avec les mêmes jeux de données (Barcelone Normal) et avec la même augmentation de données (rotation, symétrie axiale). Pour utiliser VGG-19 avec Keras, il est nécessaire d'**appliquer un preprocessing spécifique** aux images, en utilisant la fonction **preprocessing_input de l'API de VGG19** : les images sont converties au format BGR et la distribution de luminosité est centrée à 0 pour chaque pixel.

Le déséquilibre du dataset, comme nous l'avons fait pour chacun des modèles présentés auparavant, est géré en utilisant l'argument “`class_weight`” de la fonction `fit` servant à entraîner le modèle : **on pénalise davantage les erreurs commises sur les classes de faible population**, afin d'inciter le modèle à fournir un effort supplémentaire vis à vis d'elles. Le poids attribué à chaque classe est calculé de la manière suivante :

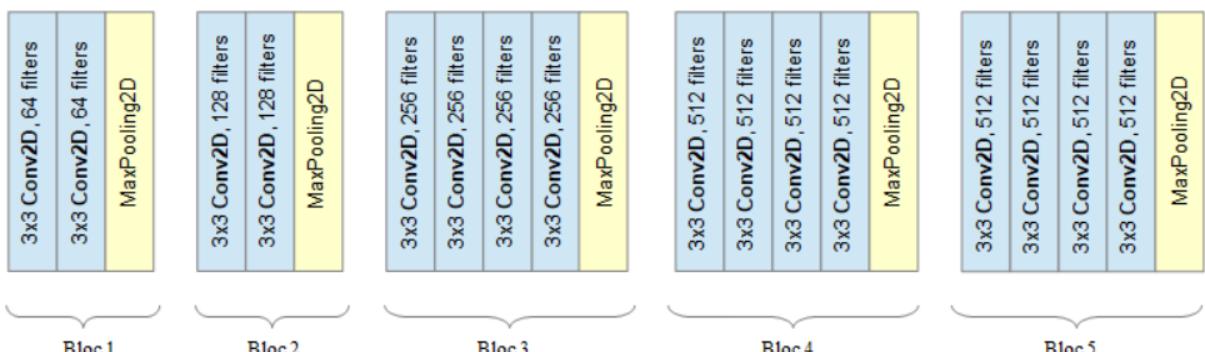
$$(1/\text{class_count}) * (\text{total_count}/2)$$

où **class_count** désigne le nombre d'images d'une classe et **total_count** le nombre total d'images.

VGG-19 - Modèle

Le modèle testé et présenté ici est construit en deux étages :

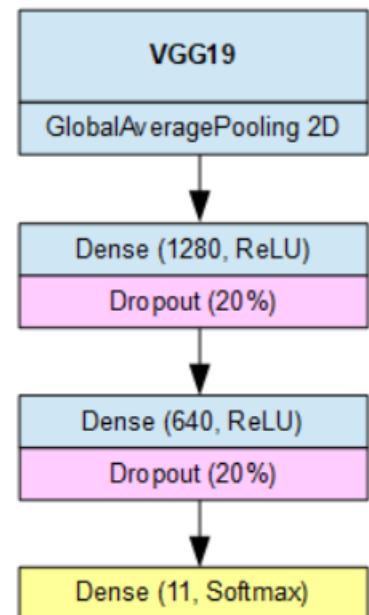
- un **étage d'extraction de features**, constitué de **VGG-19** auquel on a retiré les dernières couches (celles chargées de la classification); VGG-19 est organisé comme une succession de 5 blocs, constitués de couches de convolution. Le nombre de filtres utilisés augmente d'un bloc vers le



suivant, de 64 jusqu'à 512 :

- un **étage de classification**, constitué de trois couches **Dense** (units = 1280, 640 et 11). Les deux premières couches Denses sont activées avec la **fonction ReLU** et suivies d'une couche de **Dropout** pour réduire le surapprentissage éventuel du modèle. La dernière couche Dense est activée par la fonction **Softmax** dans le but de calculer et de retourner des probabilités d'appartenance à chaque classe.

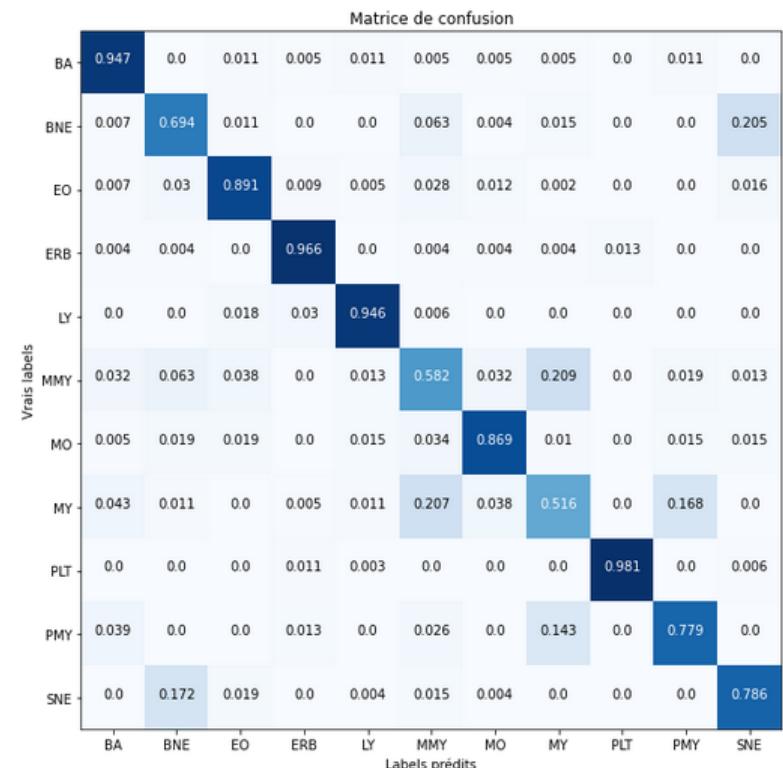
On choisit comme optimiseur l'algorithme **Adam** avec un taux d'apprentissage (learning rate) initial de **1e-3**. Le learning rate est **ajusté durant l'entraînement** grâce à un callback de type ReduceLROnPlateau, lorsque la fonction de coût mesurée sur le jeu de validation (**val_loss**) ne diminue plus.



VGG-19 - Résultats sur Normal Barcelone

L'entraînement a duré 26 epochs. L'accuracy globale obtenue sur le jeu d'entraînement est de 84,14% contre 82,49% sur le jeu de validation. Enfin, le modèle obtient **83,57% sur le jeu de test**. Le surapprentissage est faible.

Classification Report : avant Fine-Tuning				
	precision	recall	f1-score	support
BA	0.886139	0.94709	0.915601	189
BNE	0.712644	0.69403	0.703214	268
EO	0.943627	0.891204	0.916667	432
ERB	0.934694	0.966245	0.950207	237
LY	0.924419	0.946429	0.935294	168
MMY	0.525714	0.582278	0.552553	158
MO	0.895	0.868932	0.881773	206
MY	0.641892	0.516304	0.572289	184
PLT	0.991643	0.980716	0.98615	363
PMY	0.606061	0.779221	0.681818	77
SNE	0.749091	0.78626	0.767225	262



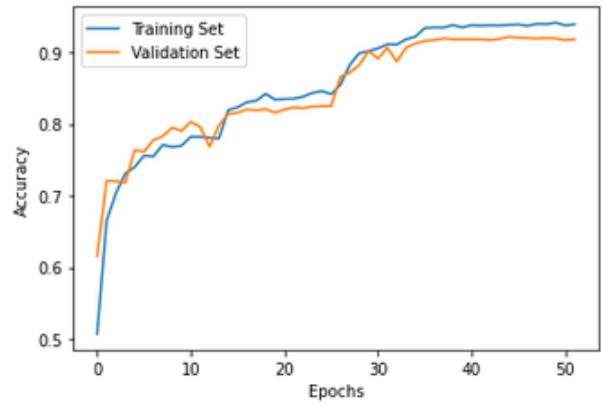
Plutôt que de nous attarder trop longtemps sur cette **performance très moyenne**, souvenons-nous que les modèles **LeNet** et **EfficientNetB3** étaient entièrement entraînés sur **Normal Barcelone** lorsque nous les avons comparés. Nous devons en faire de même avec VGG-19 si nous voulons que la comparaison ait plus de sens. Nous allons cependant adopter une approche légèrement différente : au lieu d'entraîner la totalité du modèle, nous allons procéder en deux temps pour faire du **fine-tuning**.

VGG-19 - Fine tuning pour améliorer les performances

Dans un premier temps, on importe les poids de VGG19 optimisés pour ImageNet, on gèle ce modèle et on remplace le bloc de classification par notre propre réseau de couches Denses, adapté au problème de classification de cellules sanguines. Seul le bloc de classification sera entraîné, en utilisant les features extraites par un VGG19 gelé. Cela a été présenté dans la section précédente.

Dans un second temps, **on dégèle une partie du bloc d'extraction de features pour pouvoir réentraîner les couches de convolution** qui sont au contact du bloc de classification, celles qui se spécialisent le plus par rapport au dataset d'entraînement. VGG19 est structuré en blocs : nous choisissons de **réentraîner le “block5”** de ce modèle. Celui-ci est constitué de **4 couches de convolution de type Conv2D** : ce sont ces 4 couches dont nous allons optimiser les paramètres pour Barcelone Normal.

Nous utilisons à nouveau **Adam** comme optimiseur, avec un **learning rate plus faible** que lors de la première étape ($1e-5$). L'entraînement prend 26 epochs (ci-dessus à droite, la courbe d'entraînement complète, soit 52 epochs). On observe un **gain important en accuracy globale** (train : 93.88%, valid : 91,79%, test : **91,27%**) mais également en termes de F1-score pour chaque classe (on surpassé les résultats obtenus pour LeNet, sauf en ce qui concerne les PMY ou le F1-Score diminue légèrement ; on notera que les PMY constituent la classe d'effectif le plus faible du dataset, les performances mesurées sur ce type d'images sont donc susceptibles de fluctuer plus fortement que pour les autres classes).



Matrice de confusion											
Vrais labels	BA	BNE	EO	ERB	LY	MMY	MO	MY	PLT	PMY	SNE
	185	0	0	0	1	0	1	2	0	0	0
BA	185	0	0	0	1	0	1	2	0	0	0
BNE	2	217	0	0	0	12	1	0	0	0	36
EO	0	0	431	1	0	0	0	0	0	0	0
ERB	2	1	0	232	1	1	0	0	0	0	0
LY	1	0	0	0	166	0	0	1	0	0	0
MMY	0	14	0	0	0	117	0	23	0	3	1
MO	0	2	1	0	3	6	192	0	0	2	0
MY	3	0	0	0	1	20	0	138	0	22	0
PLT	0	0	0	0	0	0	0	0	362	0	1
PMY	1	0	0	0	0	1	0	11	0	64	0
SNE	0	41	0	0	1	2	0	0	0	0	218

Classification Report : après fine-tuning				
	precision	recall	f1-score	support
BA	0.953608	0.978836	0.966057	189
BNE	0.789091	0.809701	0.799263	268
EO	0.997685	0.997685	0.997685	432
ERB	0.995708	0.978903	0.987234	237
LY	0.959538	0.988095	0.973607	168
MMY	0.735849	0.740506	0.73817	158
MO	0.989691	0.932039	0.96	206
MY	0.788571	0.75	0.768802	184
PLT	1	0.997245	0.998621	363
PMY	0.703297	0.831169	0.761905	77
SNE	0.851562	0.832061	0.841699	262

Figure 17 : Résultats obtenus sur le jeu de test avec FGG19 + fine tuning du block 5. LeNet est battu.

Grad-CAM et sélection du CNN :

Nous allons maintenant **comparer les résultats de Grad-CAM** pour avoir un critère supplémentaire permettant de choisir un ou plusieurs modèles parmi LeNet, EfficientNet et VGG19. **Xception est éliminé d'office** du fait de la mauvaise performance obtenue avec ce modèle.

LeNet :

Commençons par le modèle LeNet : on voit que pour certaines classes de cellules, comme BA, EO, LY, SNE et PNY, **LeNet s'intéresse essentiellement au contour des cellules d'intérêt, à la forme du noyau** (flagrant pour les BNE ou les SNE) et **à la structure de la chromatine à l'intérieur du noyau (PMY, LY, BA)**, autant de caractéristiques pertinentes biologiquement parlant pour identifier le type d'une cellule.

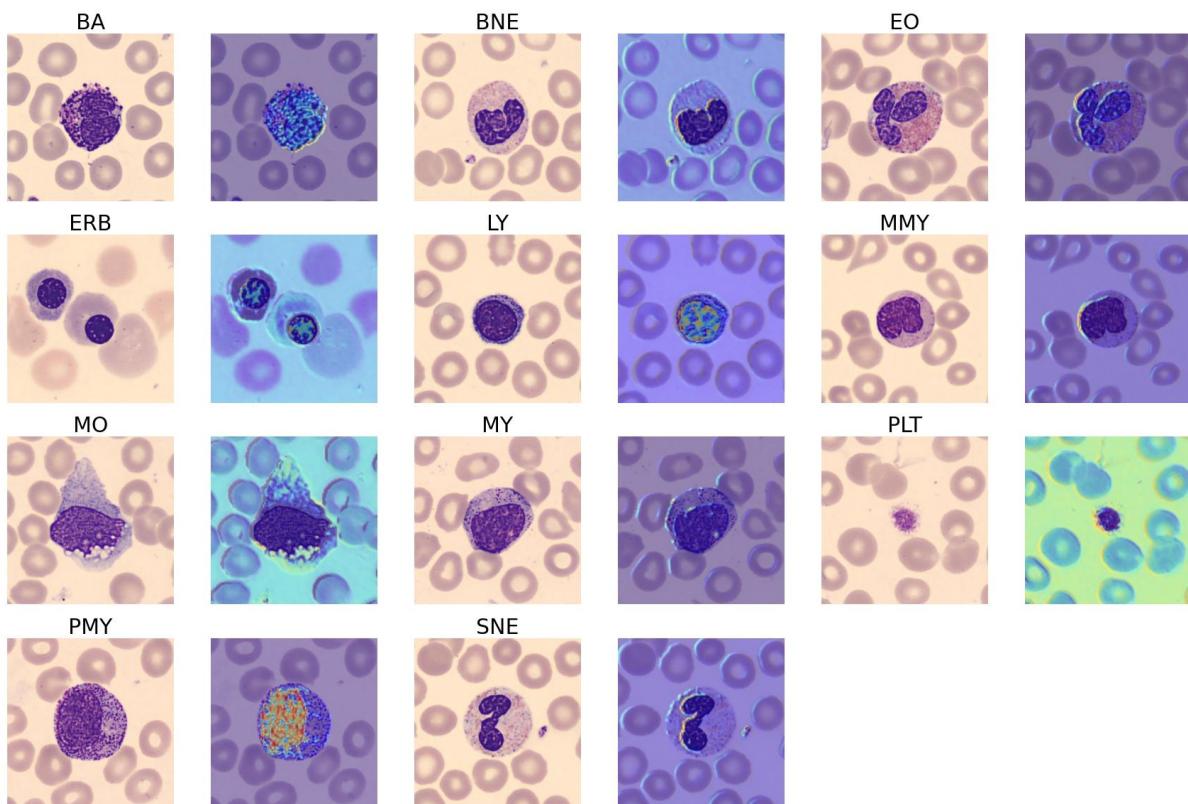


Figure 18 : Grad-CAM pour LeNet - un exemple pour chaque type de cellule

Néanmoins :

- pour PLT, ERB et MO, on voit que le modèle s'intéresse assez fortement à toute l'image en général, malgré un intérêt particulier pour le contour des cellules. LeNet est un réseau peu profond : il ne possède sans doute pas suffisamment de couches pour traquer des détails plus complexes, susceptibles de l'aider à distinguer le fond de l'objet d'intérêt lorsqu'ils se ressemblent (vaguement).
- pour les PLT en particulier, le modèle semble contrôler le contour de toutes les cellules présentes sur l'image, y compris, donc, les globules rouges. Cela s'explique peut-être par le fait que les PLT ont un diamètre comparable à celui des globules rouges;
- les EO : le modèle ne semble pas considérer les granulations roses caractéristiques de ces cellules.

LeNet est notre baseline, mais nous avons deux modèles (EfficientNetB3 et VGG19) capables de le détrôner pour ce qui est des métriques. Voyons maintenant ce qu'ils donnent du point de vue de Grad-CAM.

EfficientNetB0 :

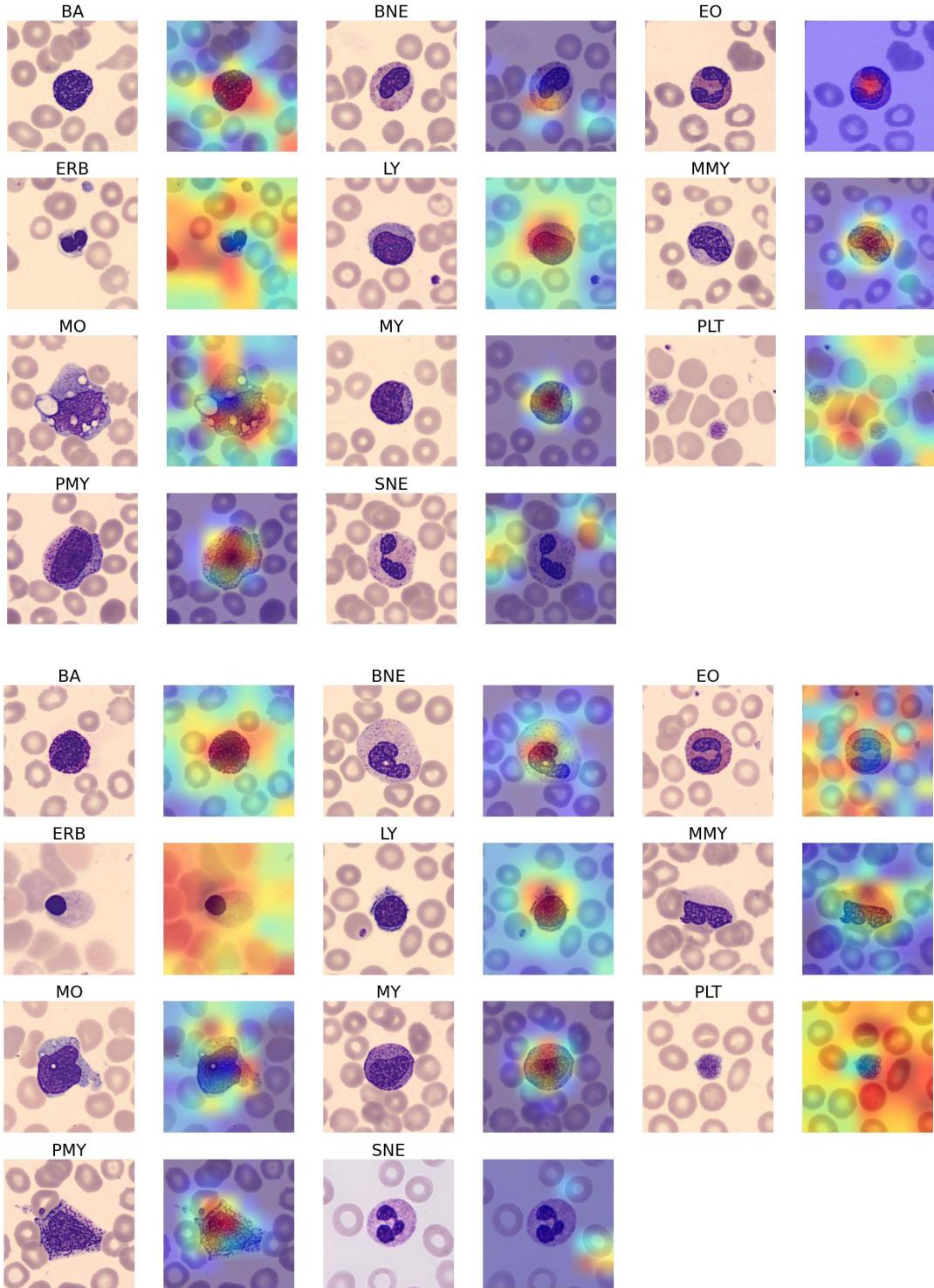


Figure 19 : Grad-CAM pour EfficientNetB0

Grad-CAM met en évidence certaines **irrégularités dans le comportement du modèle EfficientNetB0** : si pour BA, BNE, MY, MMY, PMY voire MO le modèle semble se concentrer essentiellement sur la cellule d'intérêt, au centre de l'image, ce qui est une bonne chose, les **cartes thermiques obtenues pour les ERB, les PLT et parfois EO sont plus difficiles à expliquer et à justifier**.

EfficientNetB3 :

B3 - Transfer Learning

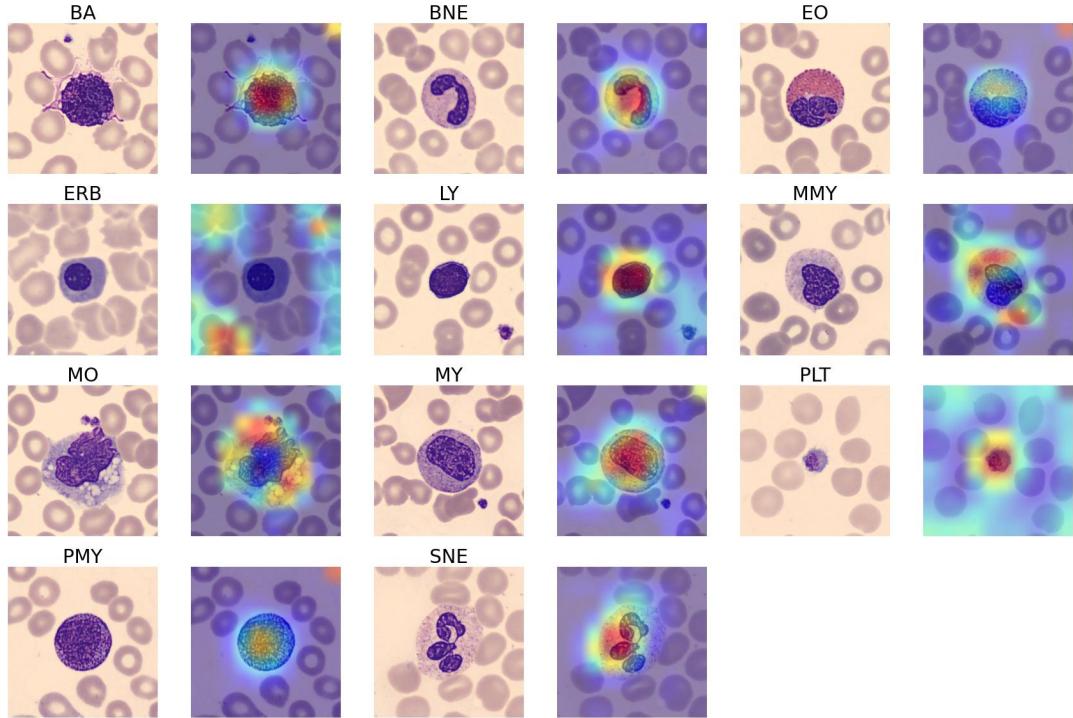


Figure 20 : Grad-CAM pour EfficientNetB3 avec transfer learning

B3 - Fully Trained

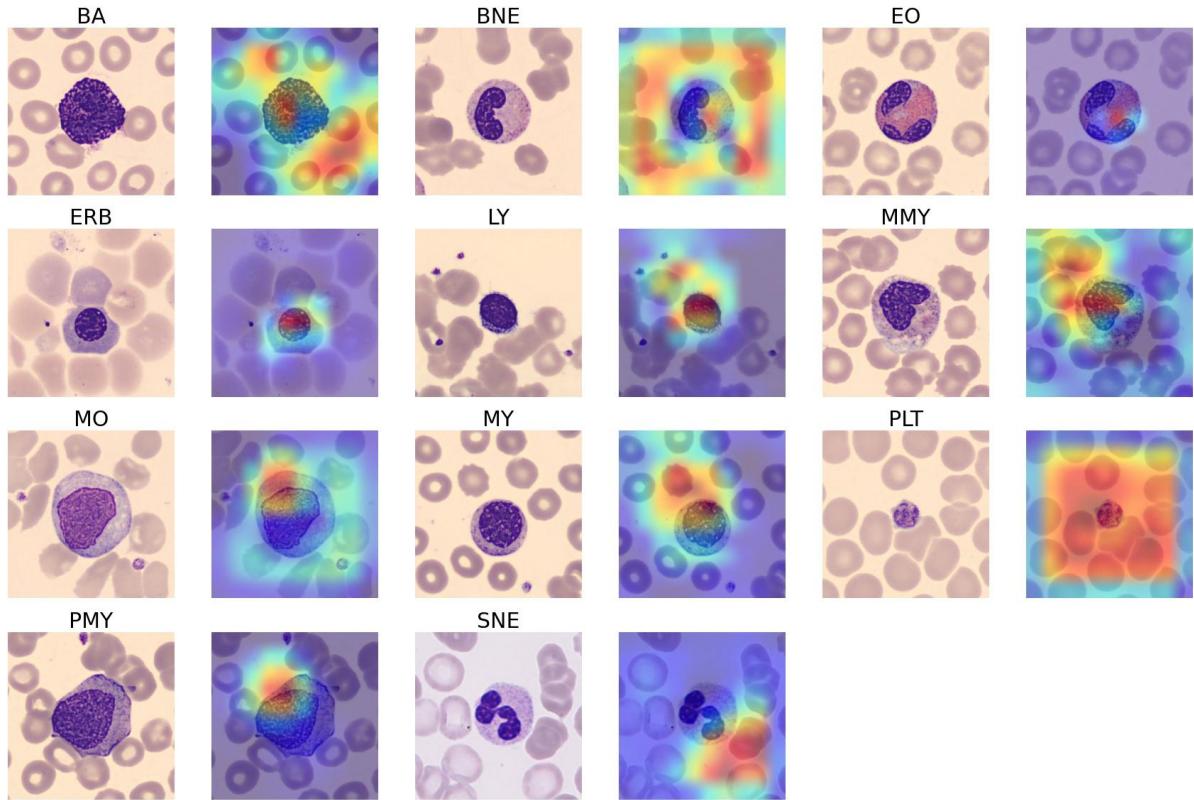


Figure 21 : Grad-CAM pour EfficientNetB3 “fully trained”

Cela peut paraître paradoxal : B3 “fully trained” obtient de meilleures métriques que B3 en transfer learning depuis ImageNet, mais **les résultats de Grad-CAM pour B3 “fully-trained” sont mauvais**: on voit que dans le cas des BNE, par exemple, le modèle s’intéresse à un carré centré autour de la cellule d’intérêt. Pire, pour les plaquettes, le modèle considère un patch carré occupant la quasi-totalité de l’image. **Cette version de B3 est donc à écarter.**

La performance du point de vue de Grad-CAM est nettement meilleure avec le modèle de transfer learning :

- on voit que le modèle cherche les granulations roses pour les éosinophiles
- qu’il s’intéresse au noyau de la cellule pour plusieurs classes.

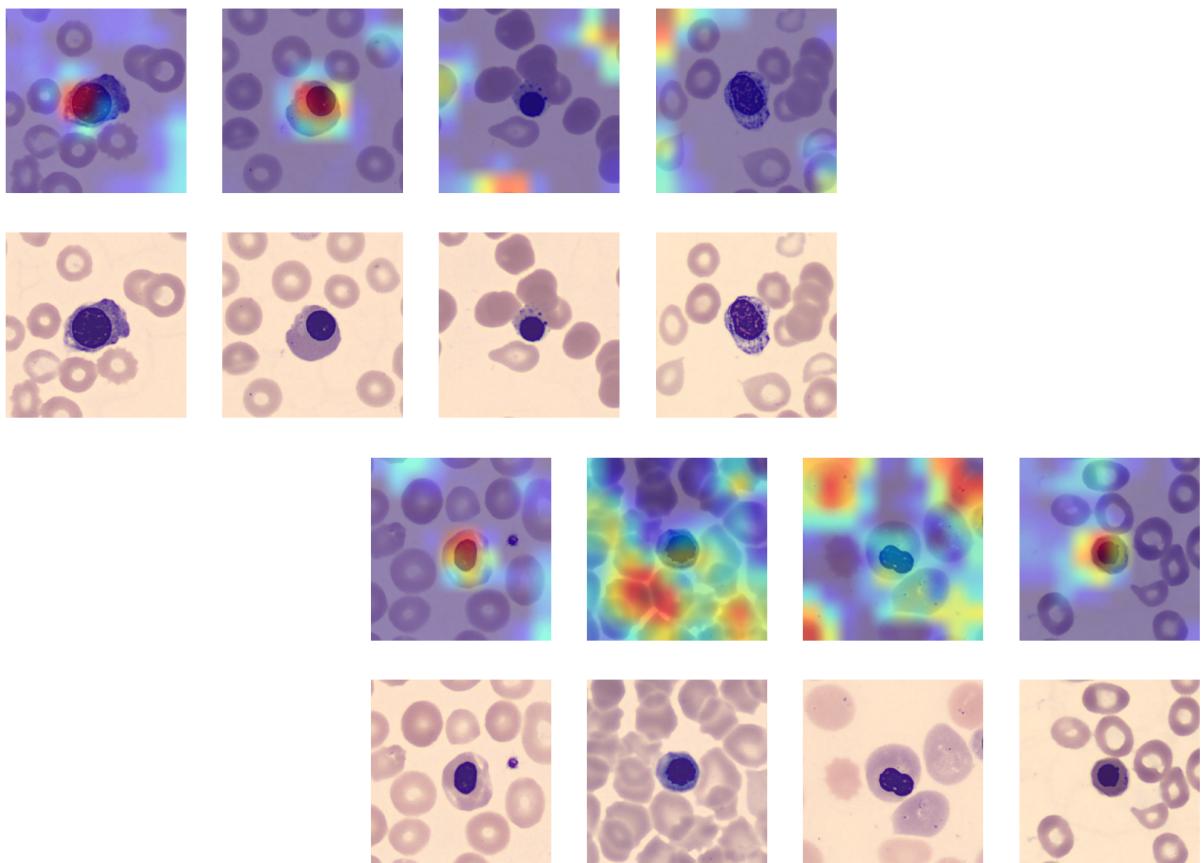
Dans le cas des plaquettes, le modèle s’intéresse un peu trop aux bords de l’image.

On remarquera enfin la présence d’un artefact sur la carte thermique dans l’un des coins de la quasi-totalité des images présentées pour B3 - transfer learning. Il s’agit d’un artefact curieux dont l’origine n’est pas comprise. Il semblerait que ce problème (ou un problème de même nature) ait été rencontré par d’autres utilisateurs d’EfficientNet, notamment les auteurs du modèle.

Sources :

- Figure 7 de <https://arxiv.org/pdf/1905.11946.pdf>,
- <https://github.com/kazuto1011/grad-cam-pytorch/issues/32>,
- <https://github.com/lukemelas/EfficientNet-PyTorch/issues/184> (voir les derniers posts de la page)

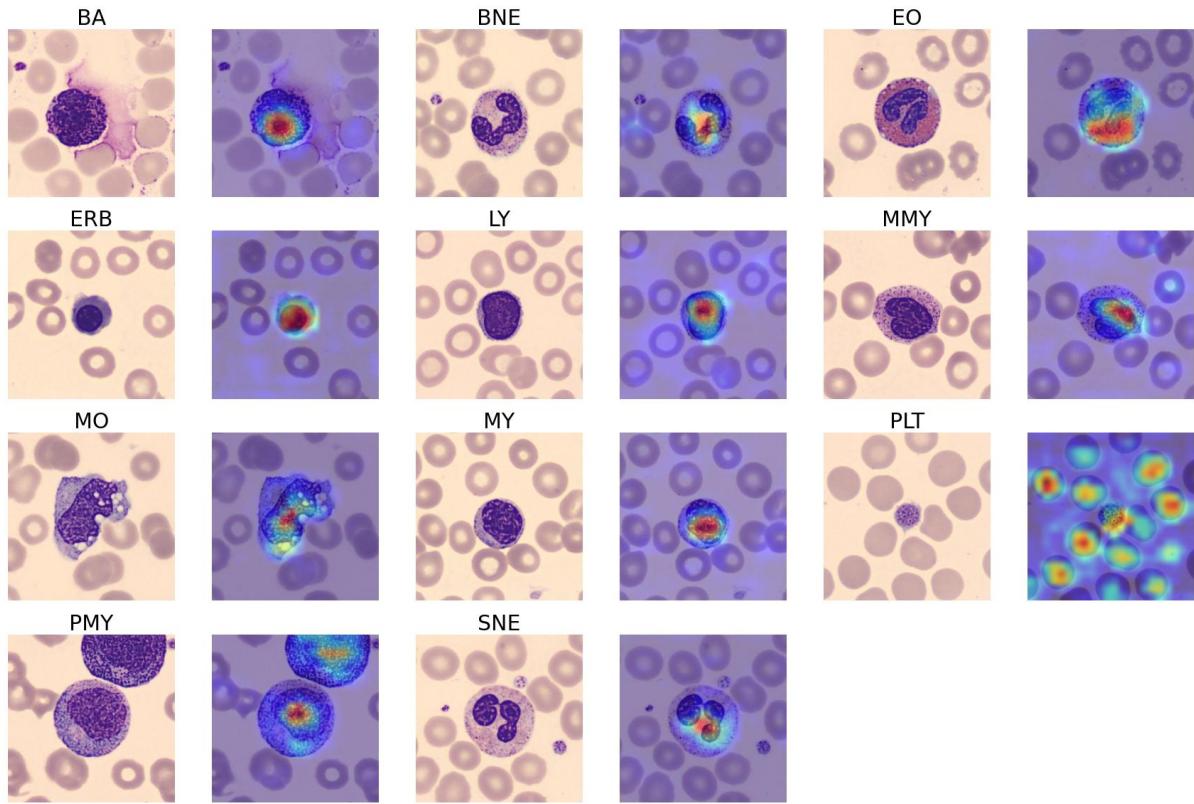
Pour les ERB, la carte thermique calculée par Grad-CAM laisse penser que le modèle semble rechercher des disques sombres, assez homogènes.. On représente ci-dessous le résultat de Grad-CAM pour 8 images d’ERB correctement classées par EfficientNetB3 + Transfer Learning :



Pour 4 images sur les 8 présentées, on voit que le modèle se focalise principalement sur la cellule d’intérêt (malgré plusieurs artefacts en bordure d’image). Les choix du modèle sont plus difficiles à expliquer pour les 4 autres images, le modèle sélectionnant tantôt des zones dépourvues de cellules et tantôt des globules rouges, isolés ou regroupés.

VGG19 :

Après le fine-tuning du block 5 : les images ci-dessous sont des images classées correctement.



Sauf pour les plaquettes, pour lesquelles le modèle a souvent tendance à repérer tout ou partie des globules rouges en plus de la cellule d'intérêt, Grad-CAM montre que **VGG-19 concentre son attention sur la cellule que nous souhaitons classer**. On remarque par ailleurs (image de PMY) que le modèle arrive à repérer de multiples cellules lorsqu'il y en a.

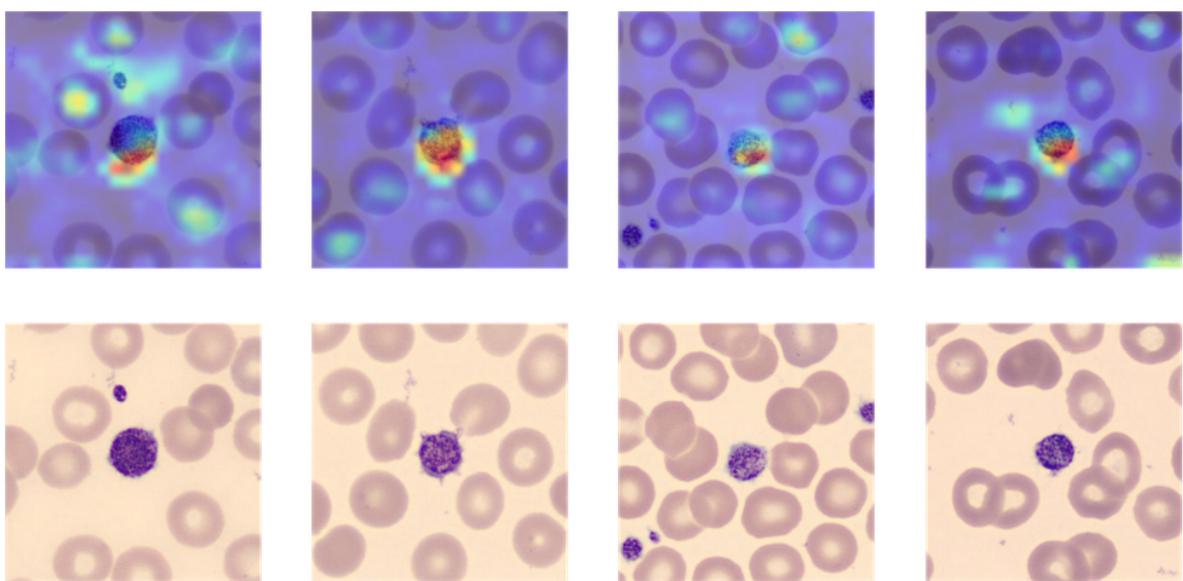


Figure 22 : images de plaquettes correctement classées - Grad-CAM montre que le modèle s'intéresse en priorité à ce qui se trouve au centre de l'image sur la plupart des images de cette classe.

Plusieurs points méritent notre attention :

- **il s'intéresse au cytoplasme et aux granules roses** sur les images d'éosinophiles : cela a du sens car ce sont les seules cellules du dataset à présenter cette particularité visible à l'œil. Cela plaide en faveur de VGG19

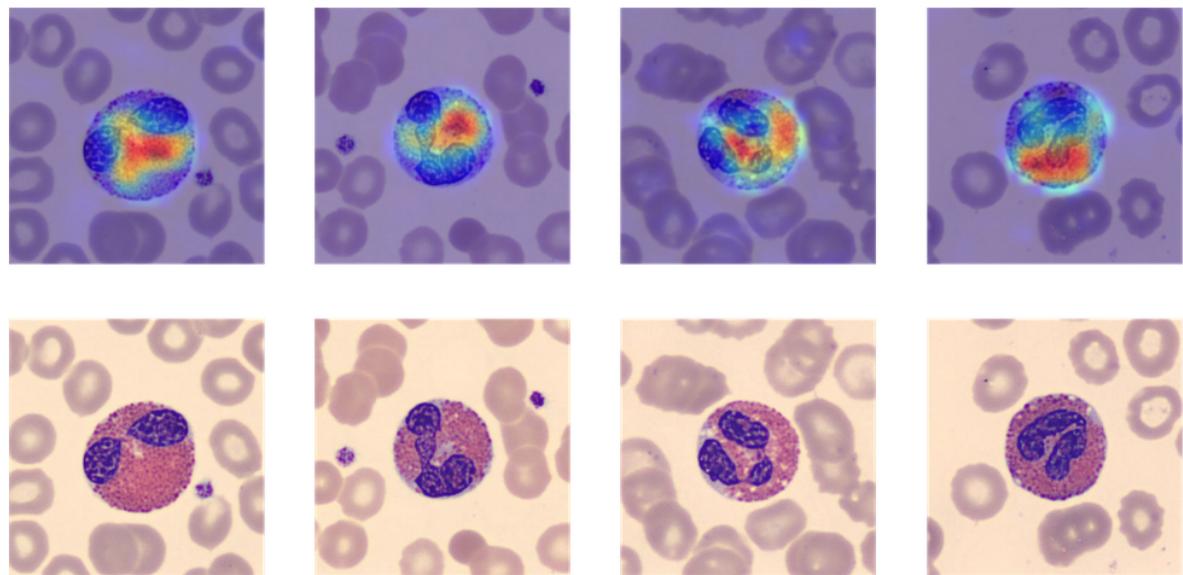


Figure 23 : images d'éosinophiles correctement classés

- pour les **BNE** et les **SNE**, il est **plus difficile d'interpréter la heatmap** : on dirait que le modèle s'intéresse à l'écart entre les lobes du noyau, mais ce n'est pas clair.

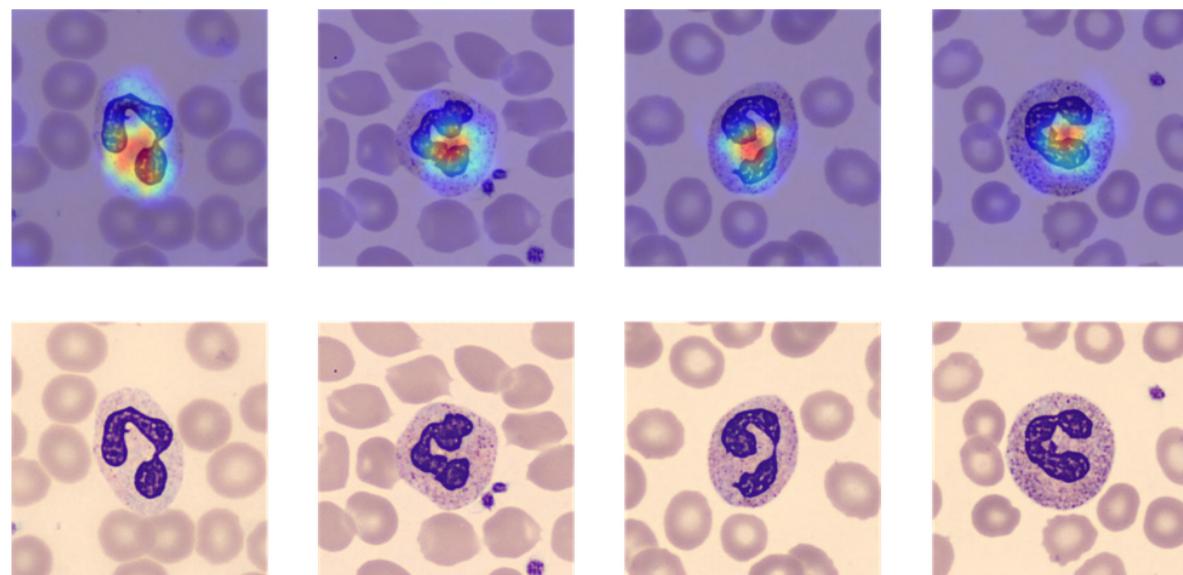


Figure 24 : images de SNE bien classées

- pour les **monocytes (MO)**, le modèle semble accorder de l'attention aux irrégularités dans la forme de la cellule : trous, déformations de la région cytoplasmique.

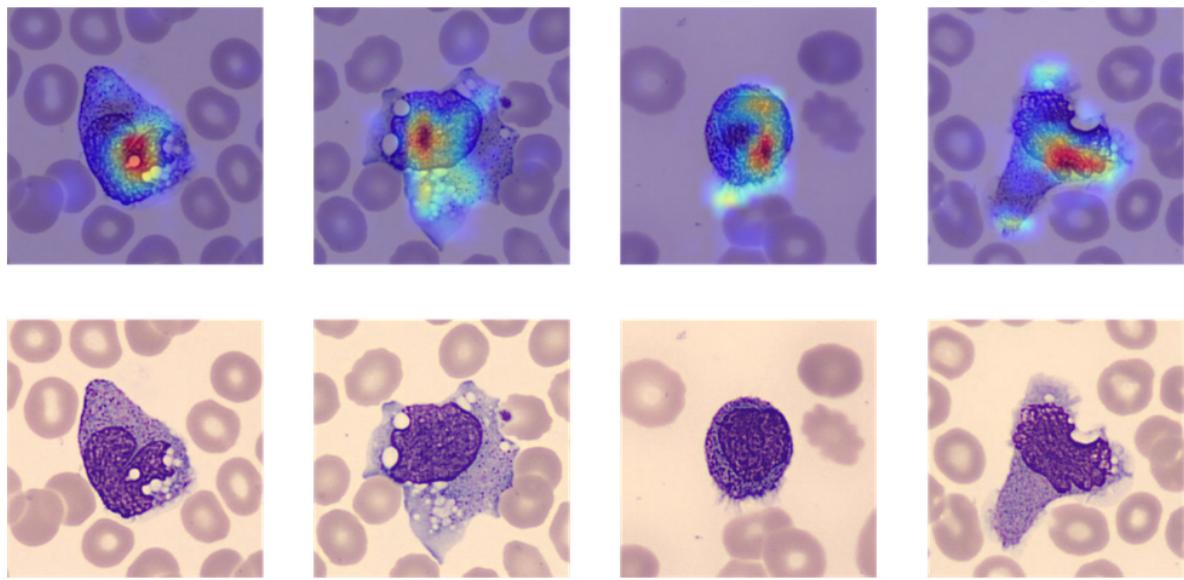


Figure 25 : images de MO bien classées

Nous retiendrons de cette analyse de nos modèles au moyen de Grad-CAM les éléments suivants :

- **LeNet semble analyser “correctement” certaines images** : le modèle semble s'intéresser au contour des cellules, de leur noyau, ainsi qu'à la répartition de la chromatine (filaments sombre dans le noyau) pour certaines classes en tout cas. Il s'agit de caractéristiques pertinentes du point de vue de la biologie pour classer les images de cellules. Un modèle plus profond, capable de percevoir des détails supplémentaires, pourrait produire une meilleure performance en termes de métriques.
- **Les modèles EfficientNet que nous avons essayés ne semblent pas fiables au vu des informations révélées par Grad-CAM** : le modèle fonde parfois ses prédictions sur des éléments qui n'ont pas de sens du point de vue la biologie (globules rouges, zones vides, variabilité importante des cartes thermiques pour une classe donnée).
- VGG19 utilisé en transfer learning avec fine-tuning montre, du point de vue de Grad-CAM, une **meilleure interprétabilité que ses rivaux** : le modèle se focalise sur des éléments de chaque cellule et très rarement sur des éléments extérieurs. **La performance de VGG19 dépasse nettement celle de LeNet une fois qu'il est ajusté (fine-tuning)**, c'est-à-dire lorsque les blocs de convolution “au contact” du bloc de classification sont ré-entraînés sur les images de Normal Barcelone.

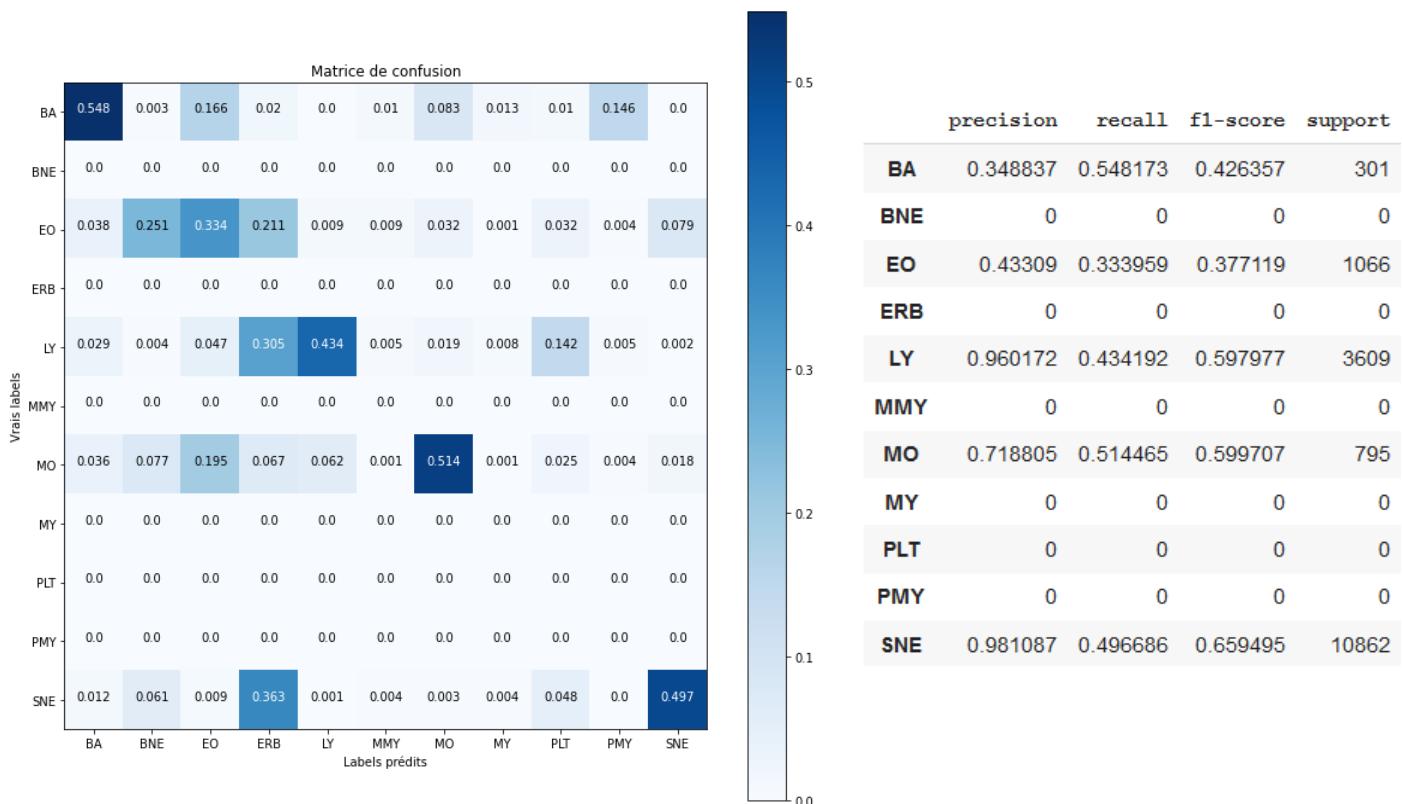
Nous privilierons donc par la suite les modèles de type VGG, laissant les autres options de côté.

VGG-19 - Test sur Raabin

Nous devons maintenant évaluer la capacité du modèle VGG19 à généraliser, c'est-à-dire à prédire correctement la classe d'images n'ayant pas été produites à l'hôpital clinique de Barcelone. Nous allons pour cela le **tester sur le dataset Raabin**. Ces images ont été produites en Iran, en utilisant un **protocole et un matériel différents de ceux utilisés pour constituer le dataset de Barcelone**.

Test sur Raabin : 16633 images regroupées dans cinq classes : **SNE** (10862, on a regroupé tous les neutrophiles sous ce label), **BA** (301 images), **MO** (795 images), **LY** (3609 images) et **EO** (1066 images).

L'accuracy globale du modèle plonge à 47% (contre 92,6% sur le jeu de test de Barcelone). La matrice de confusion et le rapport de classification nous apportent des informations supplémentaires :



- on pouvait s'attendre à ce que les images labellisées SNE soient mal classées (car elles contiennent potentiellement des BNE). Les "SNE" classées comme BNE sont minoritaires (6%), l'erreur principale commise sur les SNE étant que **le modèle en classe 36% comme ERB**;
- confusions importantes entre LY, PLT et ERB d'une part, et entre BA, EO et PMY d'autre part;
- **les éosinophiles (EO) sont extrêmement mal classées**, puisque le modèle les confond à hauteur de 21% avec des ERB et à hauteur de 25% avec des BNE. **Le F1-Score obtenu avec les EO sur le dataset de Barcelone plafonnait pourtant à 99%**. Ceci peut s'expliquer par le fait que **les images de Raabin n'ont pas été traitées avec les mêmes colorants qu'à Barcelone**.

Notre modèle est performant sur les données produites à Barcelone, mais mauvais sur celles produites en Iran : il **n'est donc pas capable de généraliser**, ce qui constitue une faiblesse majeure. Le dataset Barcelone Normal est en effet biaisé pour ce qui concerne la coloration des images et leur protocole d'acquisition (caméra utilisée, résolution des images). **Afin de résoudre ce problème, au moins partiellement, nous allons réentraîner VGG19 sur les datasets Normal Barcelone, Munich et Raabin.**

VGG-19 - Entraînement sur l'ensemble des datasets

On divise l'ensemble des données des datasets Barcelone, Raabin et Munich en trois jeux : train/valid/test comme auparavant. Voici la distribution des labels pour chacun de ces jeux :

	BA	BNE	EO	ERB	LY	MMY	MO	MY	PLT	PMY	SNE
Train	1269	1356	3741	1290	7036	830	3215	948	1864	538	8064
Valid	162	201	437	162	844	98	394	118	242	63	1048
Test	167	185	429	177	880	102	395	113	242	61	1018
Total	1598	1742	4607	1629	8760	1030	4004	1179	2348	662	10130

Certaines classes sont sur-représentées par rapport à d'autres : c'est notamment le cas des SNE et des LY. Nous allons gérer ce déséquilibre comme nous l'avons fait pour chacun des modèles présentés auparavant, en utilisant l'argument “class_weight” de la fonction fit servant à entraîner le modèle : on pénalise davantage les erreurs commises sur les classes de faible population, afin d'inciter le modèle à fournir un effort supplémentaire vis à vis d'elles. Le poids attribué à chaque classe est calculé de la manière suivante :

$$(1/\text{class_count}) * (\text{total_count}/2)$$

où **class_count** désigne le nombre d'images d'une classe et **total_count** le nombre total d'images.

Nous allons également **renforcer quelque peu l'augmentation de données**, en ajoutant le paramètre shear_range, qui produit une légère déformation de l'image (permettant de prendre en compte, au moins sur le papier, d'éventuelles différences produites par l'utilisation de systèmes optiques différents).

Le modèle VGG19 est le même que celui présenté plus haut : transfer learning à partir de VGG19 pour entraîner le bloc de classification puis fine tuning du “block 5”. Les performances obtenues en termes d'accuracy globale sont les suivantes : 0.9447 sur le jeu d'entraînement, 0.9419 sur le jeu de validation et **0.9387 sur le jeu de test**.

On constate finalement assez peu voire pas de surapprentissage : **le modèle est désormais capable de classer avec une bonne précision 11 classes de cellules provenant de trois sources indépendantes très différentes les unes des autres**.

Classification Report : avant Fine-Tuning				
	precision	recall	f1-score	support
BA	0.948276	0.988024	0.967742	167
BNE	0.660944	0.832432	0.736842	185
EO	0.995294	0.986014	0.990632	429
ERB	0.972222	0.988701	0.980392	177
LY	0.979522	0.978409	0.978965	880
MMY	0.76699	0.77451	0.770732	102
MO	0.941176	0.931646	0.936387	395
MY	0.763158	0.769912	0.76652	113
PLT	0.995851	0.991736	0.993789	242
PMY	0.774648	0.901639	0.833333	61
SNE	0.971816	0.914538	0.942308	1018

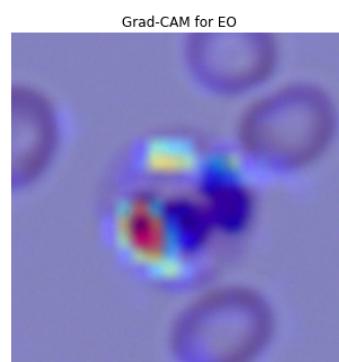
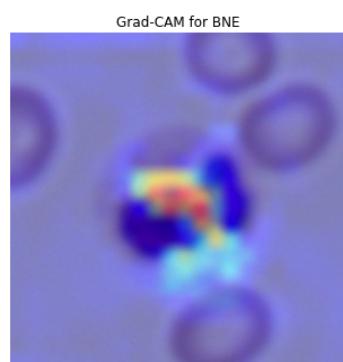
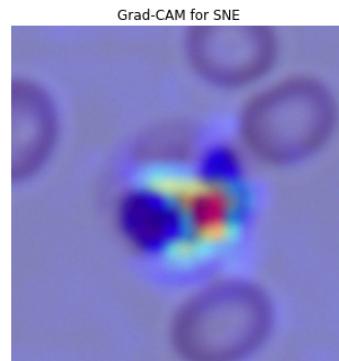
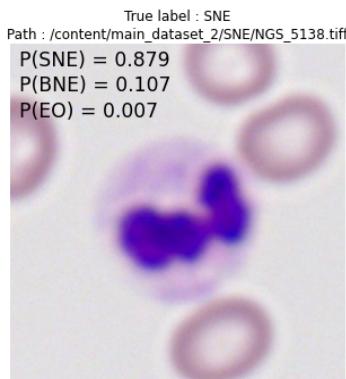
Comme avec les précédents modèles, **les principales erreurs sont les suivantes :**

- confusion entre SNE et BNE
- confusion entre PMY, MY et MMY

Nous avons souligné à plusieurs reprises que de telles confusions pouvaient s'expliquer à la lumière des connaissances en biologie. Un futur granulocyte neutrophile mature (SNE) passe par une succession de stades durant sa vie : promyélocyte (PMY), myélocyte (MY), métamyélocyte (MMY). De plus, il est difficile de différencier certaines cellules à l'œil nu, et c'est particulièrement vrai pour les stades les plus immatures (PMY, MY et MMY).

Afin de réduire le risque d'erreur, le modèle ne renverra pas un label unique pour chaque image, mais les probabilités d'appartenance aux 3 classes qu'il jugera les plus probables ainsi que la Grad-CAM calculée pour chacune des 3 classes prédites (cf. exemple ci-dessous).

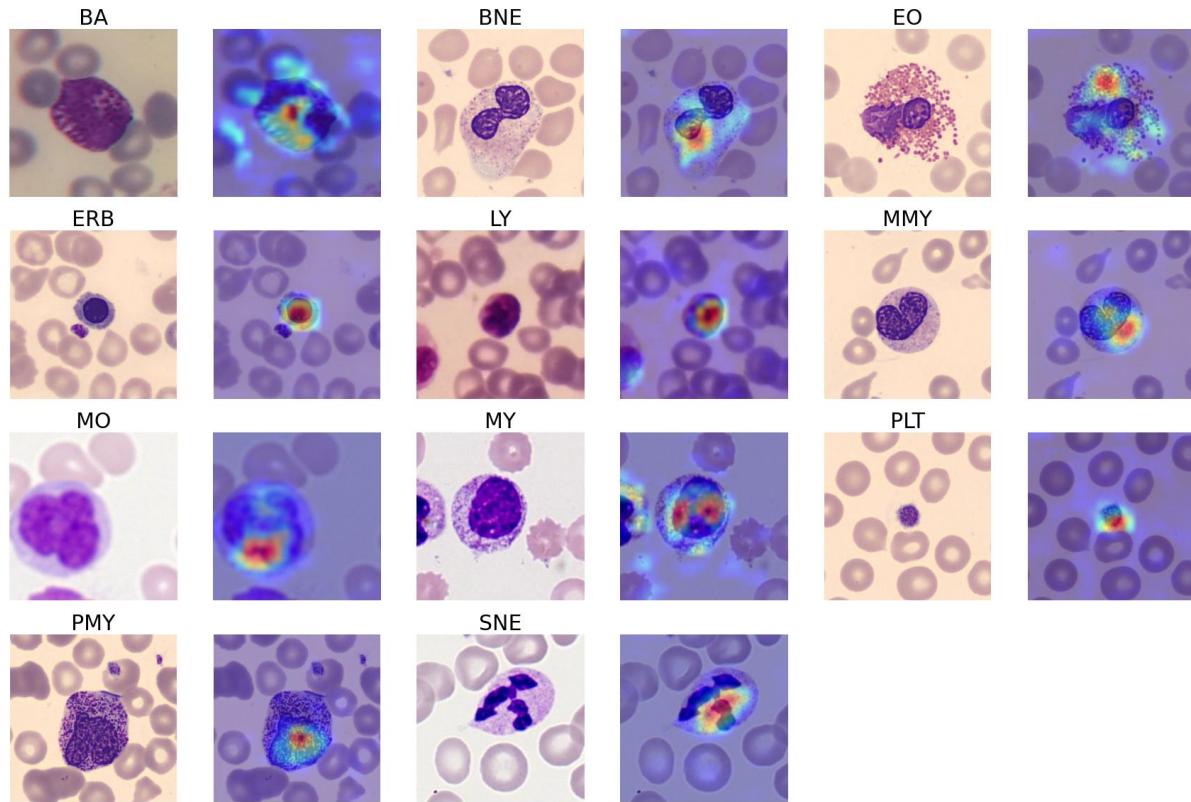
Matrice de confusion											
Vrais labels	BA	BNE	EO	ERB	LY	MMY	MO	MY	PLT	PMY	SNE
Labels prédictifs	BA	0.988	0.0	0.0	0.0	0.006	0.0	0.0	0.0	0.0	0.006
BA	0.988	0.0	0.0	0.0	0.006	0.0	0.0	0.0	0.0	0.0	0.006
BNE	0.005	0.832	0.0	0.0	0.005	0.038	0.016	0.0	0.0	0.0	0.103
EO	0.002	0.002	0.986	0.0	0.002	0.002	0.002	0.0	0.0	0.0	0.002
ERB	0.006	0.0	0.0	0.989	0.006	0.0	0.0	0.0	0.0	0.0	0.0
LY	0.0	0.0	0.0	0.003	0.978	0.0	0.014	0.002	0.001	0.0	0.001
MMY	0.0	0.01	0.0	0.0	0.01	0.775	0.0	0.196	0.0	0.01	0.0
MO	0.003	0.008	0.003	0.003	0.018	0.018	0.932	0.005	0.0	0.003	0.01
MY	0.018	0.0	0.0	0.009	0.0	0.062	0.018	0.77	0.0	0.124	0.0
PLT	0.0	0.004	0.0	0.0	0.0	0.0	0.0	0.0	0.992	0.0	0.004
PMY	0.016	0.0	0.0	0.0	0.0	0.016	0.016	0.049	0.0	0.902	0.0
SNE	0.002	0.072	0.001	0.0	0.006	0.001	0.004	0.0	0.0	0.0	0.915



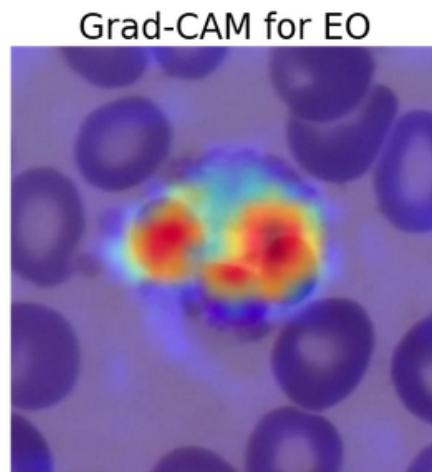
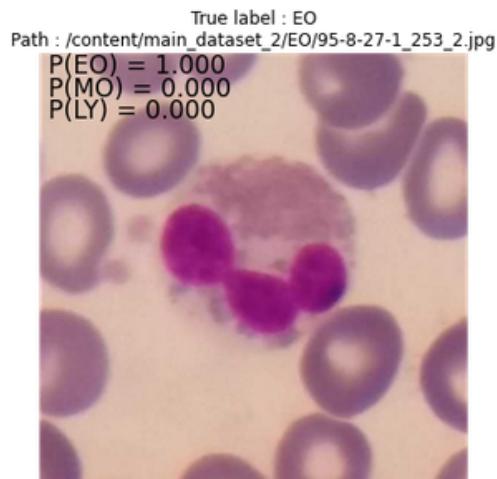
On peut voir dans cet exemple que le modèle estime à 88% que la cellule est un SNE (prédiction correcte). Grad-CAM permet de mettre en évidence la zone jugée importante pour le modèle, à savoir la jonction étroite reliant les deux lobes du noyau, élément caractéristique d'un SNE²⁰ arrivé à maturité.

²⁰ Ces cellules sont parfois qualifiées de “polynucléaires neutrophiles” en raison d'une erreur historique : le caractère plurilobé du noyau peut laisser penser que la cellule possède plusieurs noyaux.

La figure ci-dessous représente Grad-CAM pour un exemple de chaque type de cellule. On remarque que les images de cellules proviennent des trois datasets : Munich (fond clair, MO, MY et SNE), Raabin (faible luminosité, BA et LY), Barcelone (BNE, EO, ERB, MMY, PLT, PMY). L'attention du modèle est focalisée sur les cellules d'intérêt (malgré quelques difficultés pour le BA de Raabin) :

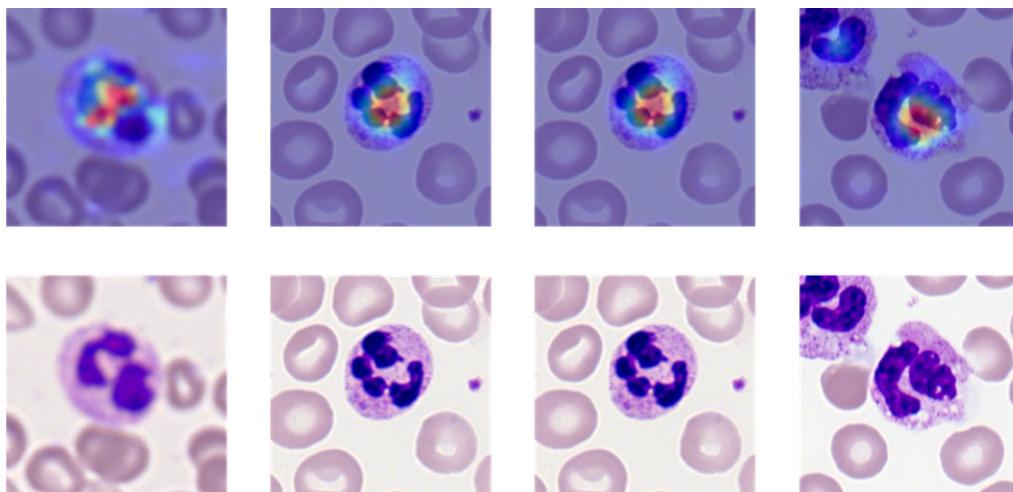


- **le modèle s'intéresse à la couleur typique des granulations des EO;** l'image ci-dessous est cependant source de questionnement :

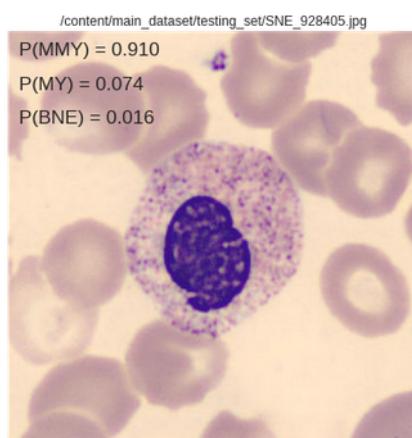
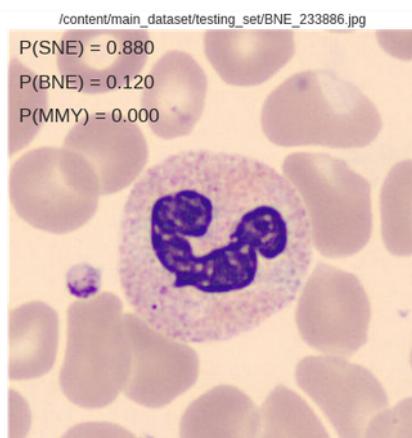
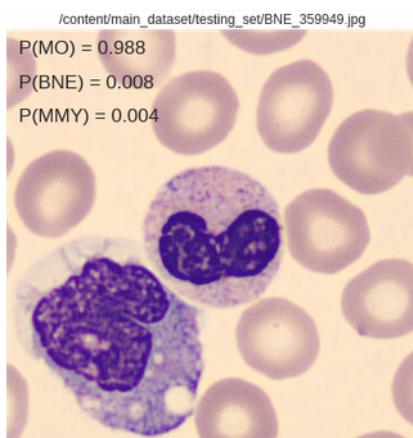


l'image est classée correctement (EO) mais on remarque avec Grad-CAM que le modèle semble fonder sa prédiction sur le noyau... qui sur cette image en particulier a la couleur des granulations roses qu'on peut observer sur les images d'EO provenant des datasets Barcelone et Munich. Dans le cas des EO, **le modèle serait donc sensible à une certaine couleur, mais ne tient peut-être pas suffisamment compte de sa localisation à l'intérieur de la cellule;**

- le modèle a beaucoup moins tendance à regarder les globules rouges lorsqu'il traite le cas des PLT : l'entraînement à partir d'images utilisant des luminosités et des colorations différentes a peut-être été bénéfique pour l'aider à “oublier” les globules rouges;
- pour ERB, LY, PMY, MY et BA, le modèle s'intéresse à des proportions diverses du noyau mais la carte thermique est assez difficile à interpréter. Il existe encore des cas où la carte thermique est aberrante;
- on voit que le modèle détecte plutôt la jonction entre les lobes du noyau de SNE (image-ci dessous) : précisons que **les SNE étant la classe de loin la mieux représentée du dataset**, ce résultat est encourageant quant à notre choix de travailler avec des modèles de type VGG.



Enfin, la figure ci-dessous représente **quatre images qui ont été mal classées par le modèle** : le label réel est indiqué dans le titre de chaque figure :



- **en haut à gauche** : l'image va nécessairement poser problème, car deux cellules blanches différentes y sont représentées. Au centre, un BNE. En bas à gauche un monocyte. Le code prédit un monocyte à 98,8%, ces cellules étant généralement les plus grosses du dataset;
- **en haut à droite** : la cellule est labellisée comme étant un BNE, mais on peut remarquer que le noyau est en train de former des lobes au niveau de ses extrémités. Le modèle prédit un SNE à 88% et un BNE à 12%, ce qui n'est pas aberrant;
- **en bas à gauche** : la cellule est labellisée comme étant un BNE. Il semble que l'extrémité gauche du noyau soit en train de former un lobe, mais c'est assez léger. Le modèle hésite fortement : 57% en faveur de SNE, 43% en faveur de BNE;
- **en bas à droite** : la cellule est considérée comme SNE alors que le noyau ne présente pas de lobes, ni une forme en "C" (nous sommes peut-être en présence d'une erreur de label) ; le modèle tranche en faveur du précurseur, le MMY, puis du MY, lui même précurseur du MMY.

Ces exemples, pris parmi de nombreux autres, montrent que même si le code peut se tromper sur le premier label, **la “bonne” réponse se trouve généralement dans les 3 classes les plus probables**. Seule l'image à en bas à droite semble mal analysée, mais nous sommes visiblement en présence d'un outlier (forme du noyau curieuse pour un SNE) ou d'une erreur de labelling.

Insistons enfin sur un point important : les problèmes de biais liés à l'origine des images ne sont pas totalement éliminés. Toutes les images de PLT proviennent du dataset Barcelone. Le dataset Raabin n'a que cinq classes de cellules et nous n'avons pas utilisé leurs neutrophiles (ne sachant pas si la distinction BNE/SNE a été faite dans les données ou si ces deux classes sont mélangées). Il faudrait compléter les datasets avec des données provenant d'autres institutions et pays et essayer d'équilibrer les classes en échantillonnage celles qui sont en surnombre. La relative facilité avec laquelle le modèle s'est adapté à l'ajout des images de Raabin et de Munich est cependant de bon augure.

VGG-16 & SVM

Cette section comprend les sous sections suivantes : Tableaux des données, Objectifs, Modèle, Algorithme de masquage, quelques exemples de résultats et Conclusion. On commence par montrer les tableaux des données qui viennent de trois bases de données différentes. Ensuite, on va fixer nos objectifs; les classifications des cellules sanguines saines en 8 classes et en 6 classes. Dans le cas de 6 classes, on a 5 sous-classes et 1 classe principale. Ensuite, on parle de notre modèle qui est un VGG16 partiellement entraîné en série avec un SVM. L'entraînement comporte 3 étapes. Les étapes sont montrées dans un schéma. Finalement, on donne l'algorithme de masquage (ou filtrage ou segmentation), on montre quelques résultats avant de finir avec la conclusion.

Tableaux des données

Les bases de données de Mendeley, de Raabin et de AML ont été présentées avant dans le texte. Les tableaux suivants sont des parties de ces bases pour lesquelles le filtre a bien fonctionné. Évidemment, pour faire une comparaison correcte entre les deux cas : le cas des données filtrées et le cas des données non filtrées, on a pris aussi un tableau correspondant, contenant exactement les mêmes fichiers mais non filtrés. Ces fichiers sont dans des tableaux avec le suffixe “Correspond”.

Données Mendeley: Data_Mendely_Segment5, Data_Mendely_Segment5_Correspond

dossier	Nombre de fichiers	dossier	Nombre de fichiers
basophil	1178	lymphocyte	1197
eosinophil	3006	monocyte	1357
erythroblast	1491	neutrophil	3260
ig	2511	platelet	2348

Données Raabin: Data_Raabin_Segment, Data_Raabin_Segment_Correspond

dossier	Nombre de fichiers	dossier	Nombre de fichiers
Basophil	158	Lymphocyte	1867
Eosinophil	361	Monocyte	265
		Neutrophil	3045

dossier	Nombre de fichiers	dossier	Nombre de fichiers
BAS	71	MON	848
EBO	74	MYB	24
EOS	365	NGB	95
LYT	2535	NGS	1986
MMZ	13	PMB	10
		PMO	0

Objectifs

Nos objectifs sont :

- Classification en 8 classes principales, étudier l'effet de masquage
- Classification en 6 classes (1 principale +5 sous classes), étudier l'effet de masquage

Classification en 8 classes principales, étudier l'effet de masquage

§ Tableaux des données :

On considère tous les tableaux de données. Les données non filtrées sont dans les tableaux qui portent dans le préfix « Correspond ». Il y a 28065 fichiers d'images jpg et tiff.

§ Classes :

8 classes des cellules sanguines saines prise en compte sont :

"neutrophil", "eosinophil", "ig", "platelet", "erythroblast", "monocyte", "basophil", "lymphocyte".

Les données de tableaux de AML nécessitent un traitement pour être classées dans ces 8 classes. On a donc remplacé dans ce tableau :

"BAS", "EBO", "EOS", "LYT", "MMZ", "MON", "MYB", "NGB", "NGS", "PMB", "PMO"

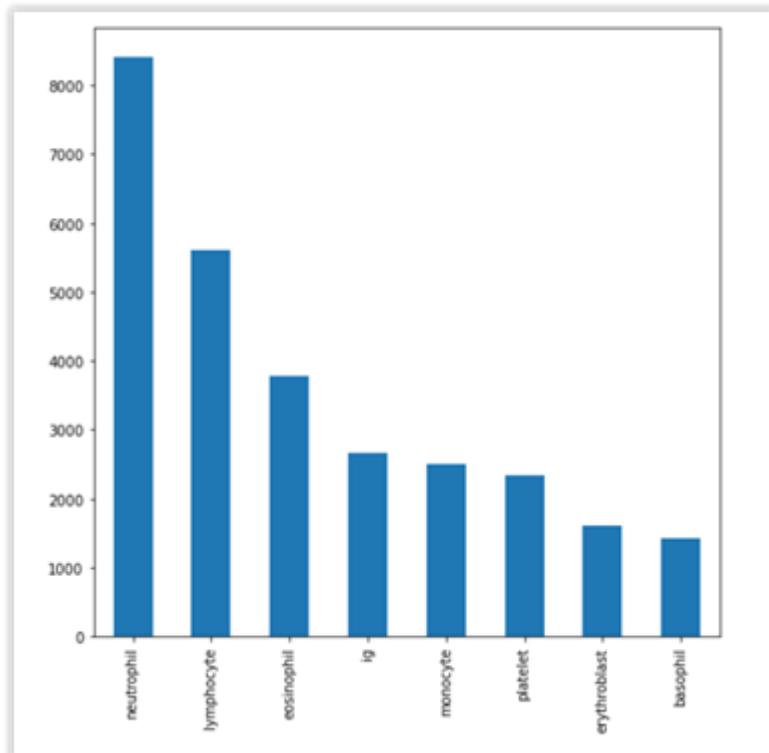
par

"basophil", "erythroblast", "eosinophil", "lymphocyte", "IG_MMY", "monocyte", "IG_MY", "neutrophil", "neutrophil", "IG_PMY", "IG_PMY"

et ensuite :

“IG_IG”, “IG_MMY”, “IG_MY”, “IG_PMY” sont mis dans une classe nommée : “ig”.

La répartition des fichiers est montrée dans la figure suivante :



§ Données de training et de test :

Les données (28065 fichiers) sont partagées avec `test_size=0.2`. Nous obtenons 22452 fichiers pour les données de training et 5613 fichiers pour les données de test . En plus, pour pouvoir faire une comparaison précise entre deux cas les données filtrées et non filtrées, on utilise le même random seed dans les deux cas.

§ Modèle :

Le modèle est montré dans la suite. Il s'agit de VGG16+SVM. L'entraînement est fait en 3 étapes. Les étapes sont montrées dans le schéma. Sur ce schéma, dans chaque étape, seulement les blocs non colorés sont entraînés, et les blocs hachurés gardent les mêmes valeurs que l'étape précédente.

§ Résultats :

Sans masque :

	precision	recall	f1-score	support
neutrophil	0.98	0.99	0.99	1702
eosinophil	0.99	0.99	0.99	734
ig	0.96	0.95	0.96	489
platelet	0.99	1.00	0.99	509
erythroblast	0.98	0.97	0.97	291
monocyte	0.96	0.93	0.94	514
basophil	0.97	0.99	0.98	267
lymphocyte	0.97	0.98	0.98	1107
accuracy			0.98	5613
macro avg	0.98	0.97	0.97	5613
weighted avg	0.98	0.98	0.98	5613

Avec masque :

	precision	recall	f1-score	support
neutrophil	0.98	0.98	0.98	1702
eosinophil	0.98	0.97	0.98	734
ig	0.90	0.94	0.92	489
platelet	0.99	0.99	0.99	509
erythroblast	0.98	0.96	0.97	291
monocyte	0.93	0.92	0.93	514
basophil	0.96	0.97	0.97	267
lymphocyte	0.97	0.96	0.96	1107
accuracy			0.97	5613
macro avg	0.96	0.96	0.96	5613
weighted avg	0.97	0.97	0.97	5613

On constate que l'usage du masque ne change pas considérablement les résultats.

§ Fichiers :

-MTL_Segment_8classes_Correspond.ipynb : classification avec les données non filtrées data_Total_Segment_Correspond1_T.pkl

-MTL_Segment_8classes.ipynb : classification avec les données filtrées data_Total_Segment1_T.pkl

Classification en 6 classes (1 principale +5 sous classes), étudier l'effet de masquage

§ Tableaux des données:

On considère seulement les données de Mendely, composé de 16348 fichiers.

§ Classes :

L'objectif est de reconnaître des sous classes « IG_IG », « IG_MY » , « IG_MMY » , « IG_PMY », de classe « ig » et des sous classes

« BNE_NEUTROPHIL », « BNE_BNE », « BNE_SNE » de classe NEUTROPHIL.

Puisque les nombres de données dans les classes « IG_IG » et « BNE_NEUTROPHIL » ne sont pas suffisants, on les néglige. Ainsi, on réduit le nombre de données à 16204.

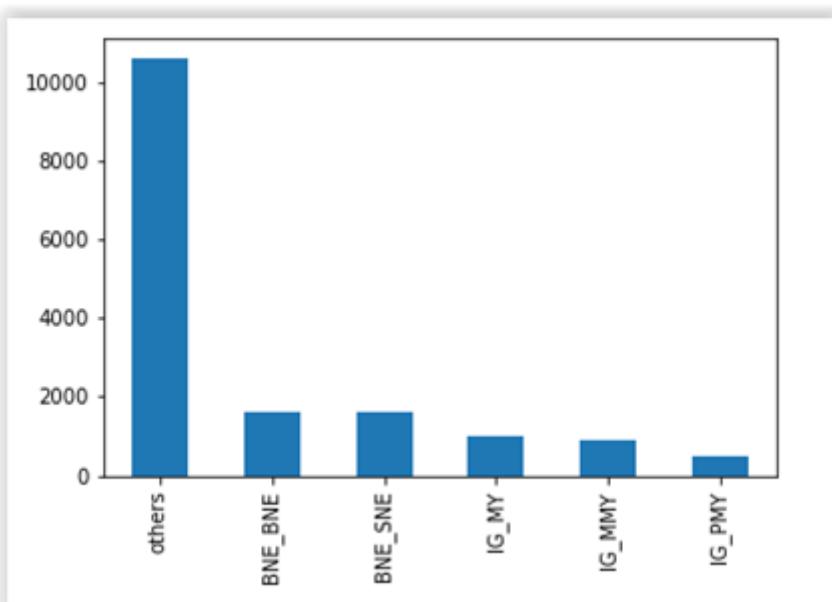
De plus, on met les autres classes : “eosinophil”, “platelet”, “erythroblast”, “monocyte”, “basophil” et “lymphocyte”

dans une classe nommée : « Others ». Ce choix est basé sur l'analyse suivante: on peut avoir deux modèles VGG16+SVM en série ! Le premier modèle fait une classification en 8 classes, comme on l'a expliqué dans la section précédente. Le deuxième modèle, il n'est actif que si le résultat du premier modèle était « neutrophil » ou « ig ». Dans ce cas, le deuxième modèle reprend la donnée pour trouver la sous classe de « neutrophil » ou de « ig ». Évidemment, puisque le premier modèle peut se tromper et indiquer « neutrophil » ou « ig » tandis que c'était d'autres classes, on prend la classe « Others » dans le deuxième modèle. Donc, ici l'objectif est de trouver des paramètres de ce deuxième modèle.

Finalement, on aura les 6 classes suivantes :

« IG_MY » , « IG_MMY » , « IG_PMY » , « BNE_BNE » , « BNE_SNE » et “Others” .

La répartition des données est montrée dans la figure suivante:



§ Données de training et de test :

On partage les données en training et test avec `test_size=0.2`. Ainsi, on obtient 12963 fichiers pour training et 3241 fichiers pour le test. On utilise le même numéro de random seed dans les deux cas ; le cas des données avec le filtre et le cas des données sans filtre.

§ Modèle :

Le modèle est montré dans la suite. Il s'agit de VGG16+SVM, comme dans le cas des 8 classes.

§ Résultats :

Sans masque:

	precision	recall	f1-score	support
<code>Others</code>	0.98	0.99	0.99	2114
<code>BNE_BNE</code>	0.75	0.81	0.78	323
<code>BNE_SNE</code>	0.84	0.78	0.81	335
<code>IG_MY</code>	0.73	0.76	0.74	205
<code>IG_MMY</code>	0.72	0.72	0.72	166
<code>IG_PMY</code>	0.81	0.64	0.72	98
<code>accuracy</code>			0.91	3241
<code>macro avg</code>	0.81	0.78	0.79	3241
<code>weighted avg</code>	0.91	0.91	0.91	3241

Avec masque:

	precision	recall	f1-score	support
Others	0.99	0.99	0.99	2114
BNE_BNE	0.76	0.81	0.79	323
BNE_SNE	0.85	0.80	0.83	335
IG_MY	0.73	0.75	0.74	205
IG_MMY	0.75	0.70	0.72	166
IG_PMY	0.80	0.68	0.74	98
accuracy			0.92	3241
macro avg	0.81	0.79	0.80	3241
weighted avg	0.92	0.92	0.91	3241

On constate, de nouveau, que l'usage du masque ne change pas considérablement les résultats. Les résultats ne sont pas aussi bons que pour la 8 classes.

§ Fichiers :

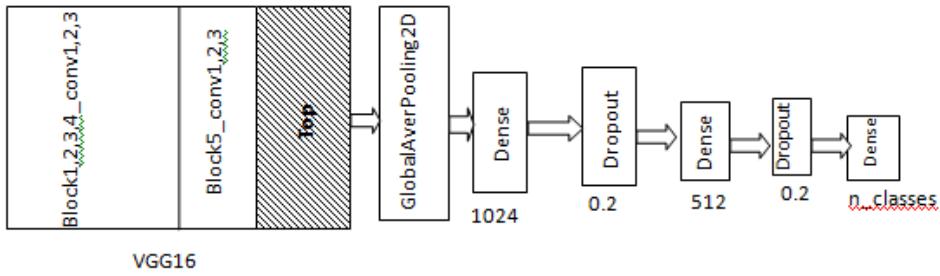
-MTL_Segment5.ipynb : classification avec les données filtrées :
data_1_13_classes_Segment5.pkl

-MTL_Segment5_Correspond.ipynb : classification avec les données non filtrées :
data_1_13_classes_Segment5_Correspond.pkl'

Modèle

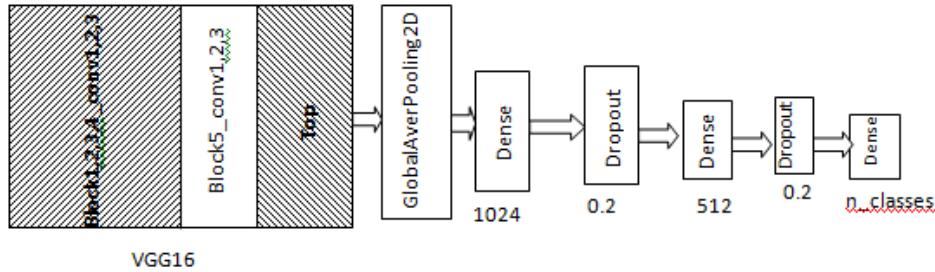
Comme nous l'avons déjà indiqué, c'est un modèle avec VGG16 partiellement entraîné en série avec un modèle Support Vector Machine. L'entraînement est fait en trois étapes comme la figure suivante le montre. Dans chaque étape, les blocs hachurés ne seront pas entraînés. C'est une méthode de Transfer Learning. Le paramètre n_classes est fixé à 8 pour notre premier objectif et fixé à 6 pour le deuxième objectif. L'optimisation est faite par Adam avec learning rate de 1e-4. Les données ont été augmentées pour faire face à leur répartition non équilibrée.

Etape 1 : epochs=10



VGG16

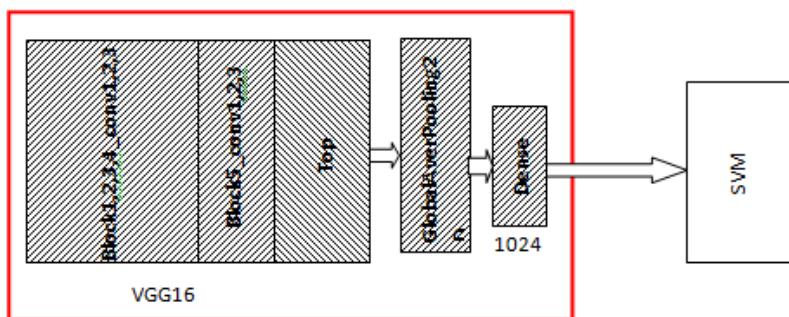
Etape 2 : epochs=30



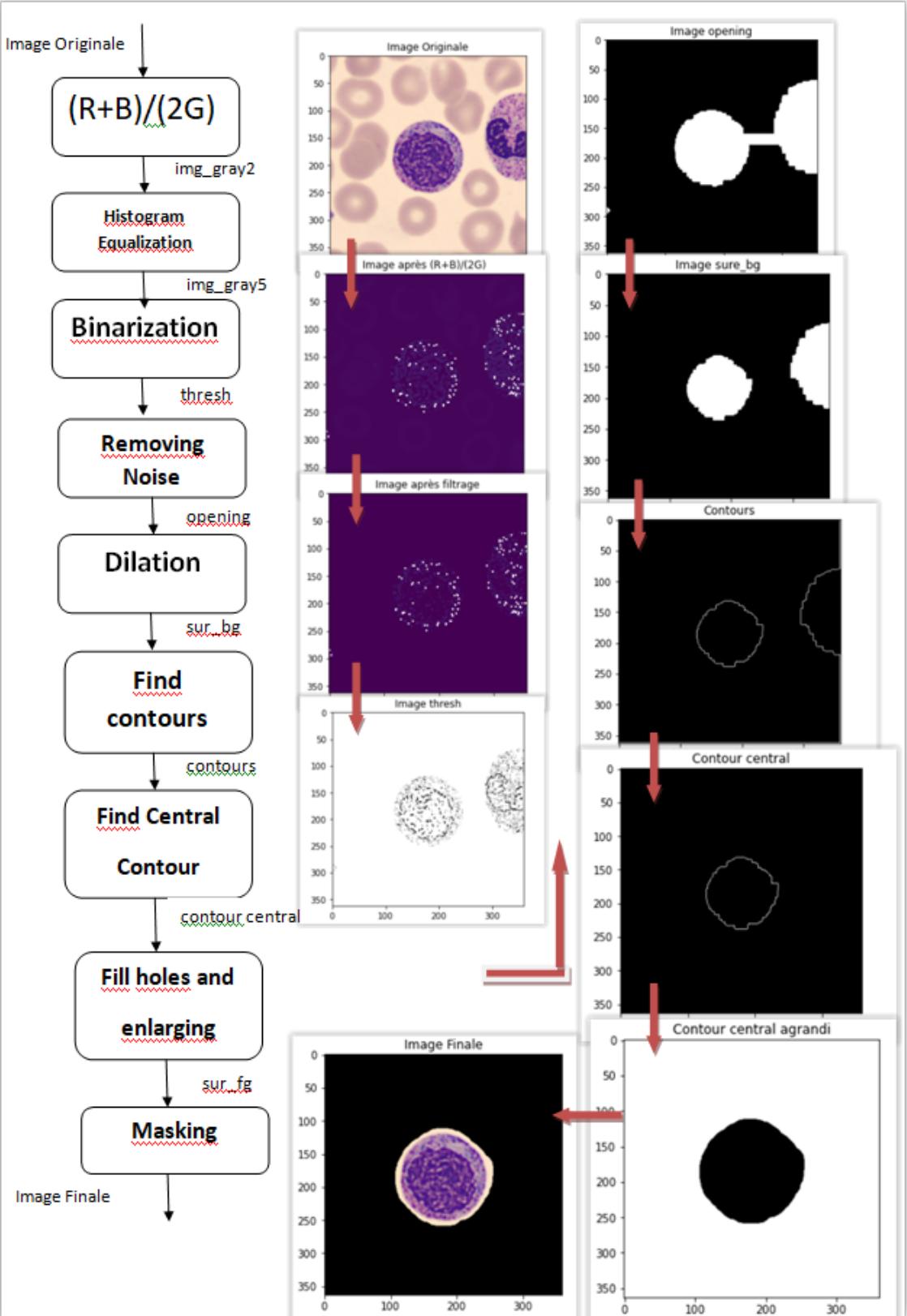
VGG16

Etape 3 : GridSearchCV

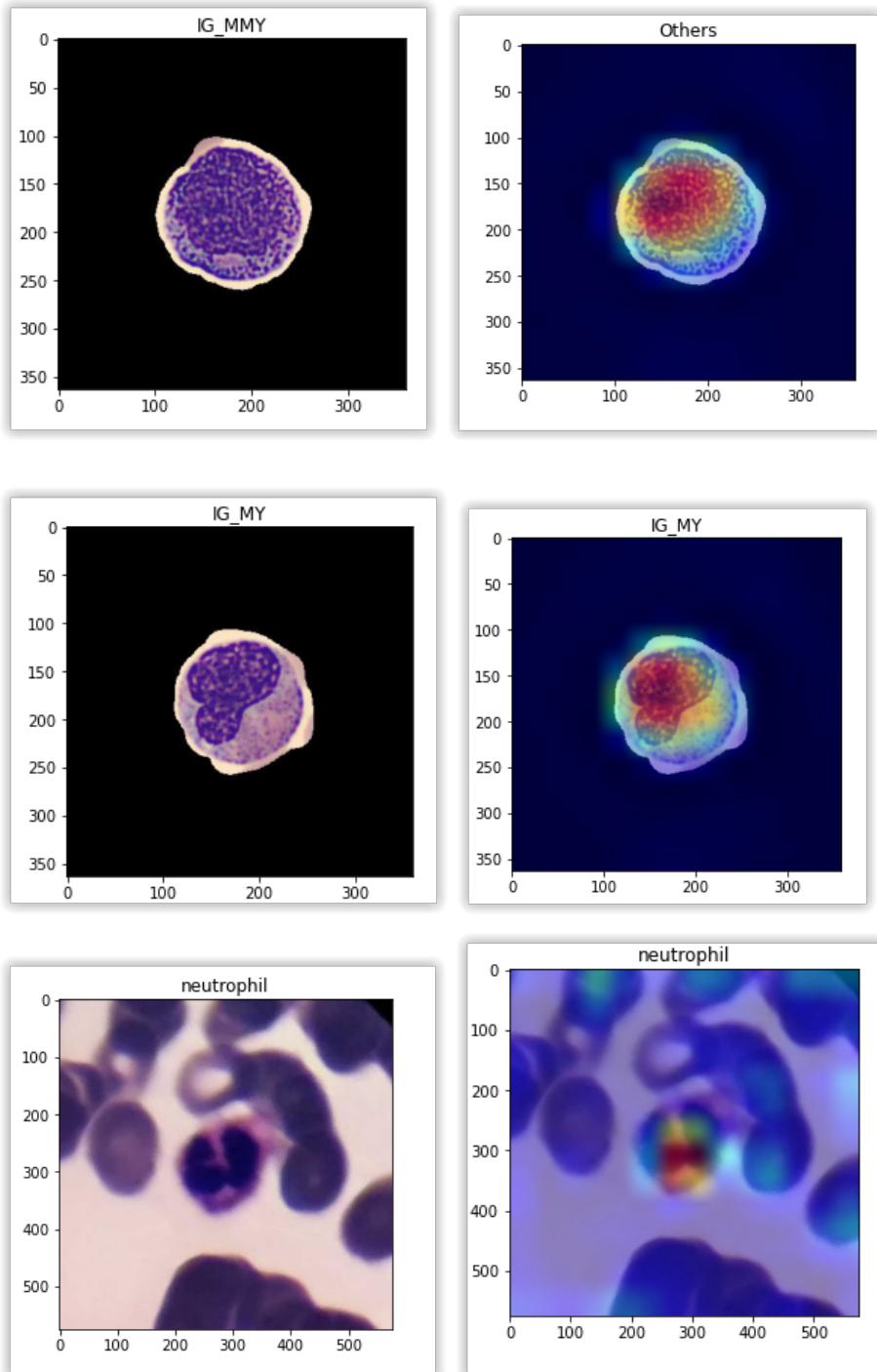
Intermediate model

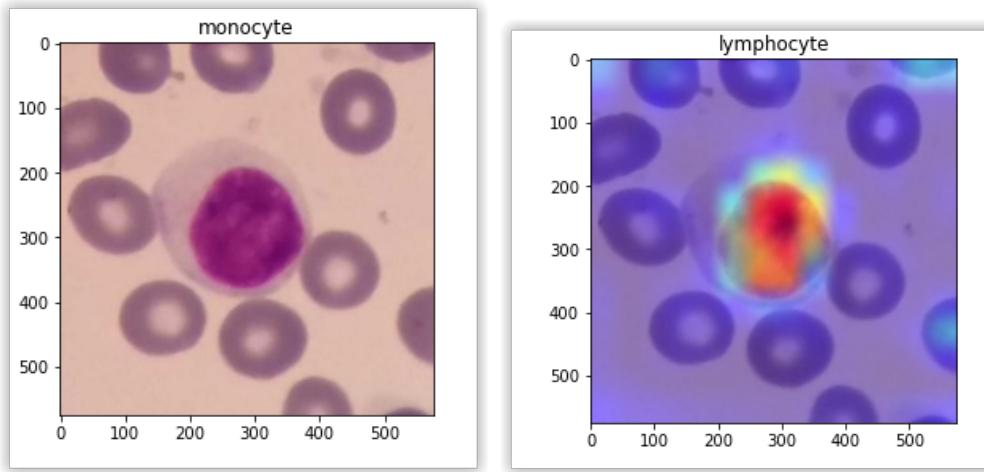


Algorithme de masquage (ou filtrage ou la segmentation)



Quelques exemples (avec gradcam)





Conclusion

D'après ces résultats, le masquage n'a pas forcément amélioré les résultats : ni dans le cas de 8 classes, ni dans 6 classes. Par contre, le choix du modèle pour le cas de 8 classes était bon. Pour le cas de 6 classes, on pourrait tester d'autres modèles, par exemple VGG19, toujours en mode Transfer Learning.

Vision Transformer

Nous avons aussi essayé un modèle de vision transformer (ViT)²¹. Ce type modèle est apparu comme prometteur du fait de la couche d'attention qui **permet au modèle de se concentrer sur un zone spécifique** de l'image (Fig. 6 et Fig. 7).



Figure 6: La carte d'attention des ViT. (source Keras implementation of ViT (Vision Transformer) - GitHub)

De plus, ces modèles requièrent moins de temps de calcul. Comme les couches d'attention utilisent une opération quadratique (pairwise distance), appliquer cette couche sur une image de 300x300 se révèle coûteux. Une solution est de découper l'image en patches (16x16 ou 32x32, flatten) connectés par un embedding positionnel. De plus, les processus d'attention (self attention) sont dits ‘multi-head’ et sont exécutés en parallèle. A l'inverse des transformateurs utilisés sur des données textuelles, les ViT n'ont pas décodeur mais une tête MLP (multilayer perceptron). Nous avons utilisé une librairie disponible sur fond Keras; **ViT-Keras**²². Les ViT sont recommandés uniquement en transfert learning, puisqu'ils n'ont pas l'équivariance translationnelle des CNNs²³.

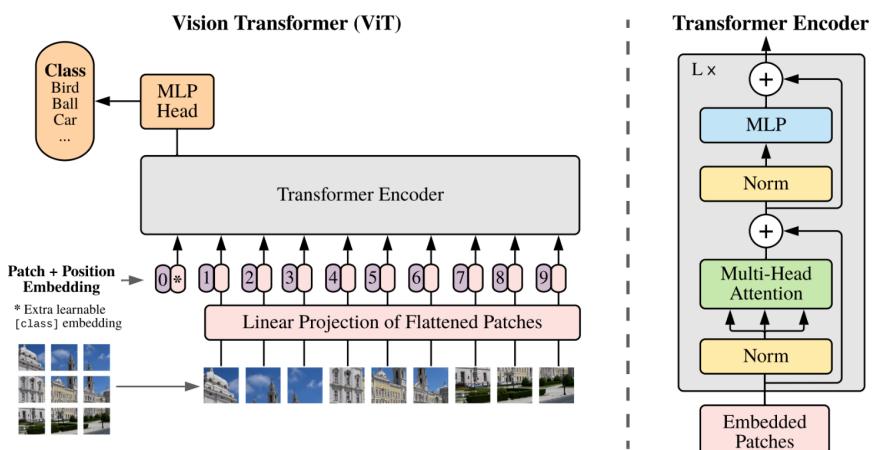


Figure 7: Vision Transformer: architecture. (source An Image is Worth 16x16 Words: Transformers)

²¹ "[2010.11929] An Image is Worth 16x16 Words: Transformers - arXiv." <https://arxiv.org/abs/2010.11929>. Accessed 27 Sep. 2021.

²² "Keras implementation of ViT (Vision Transformer) - GitHub." [https://github.com/faustumorales/vit-keras](https://github.com/faustomorales/vit-keras). Accessed 27 Sep. 2021.

²³ "How to train your ViT? Data, Augmentation, and Regularization in" 18 Jun. 2021, <https://arxiv.org/pdf/2106.10270.pdf>. Accessed 28 Sep. 2021.

Pour nos images, nous avons utilisé une résolution de 352 x 352 (patch 32, ViT-b32) ou 224 x 224 (patch de 16x16, ViT-b16). La librairie ViT-Keras offre la possibilité de fine-tuning avec les code suivant:

```
image_size = 224
model = vit.vit_b32(
    image_size=image_size,
    pretrained=True,
    include_top=False,
    pretrained_top=False)
```

En utilisant ce code avec une couche Dense avec une activation softmax nous avons obtenu une accuracy de 86-88%. Cependant, en inspectant le nombre de paramètres et les couches du ViT nous avons constaté que le modèle était entièrement ré-entraîné sur nos données. Nous avons donc essayé de bloquer les couches du ViT mais l'accuracy à diminué à 60%. En regardant le code source sur GitHub il est difficile d'identifier comment améliorer le code ou quelles sont les raisons de cette baisse.

N'ayant pas encore reçu le training sur PyTorch et du fait de la nouveauté du modèle (essentiellement Kaggle notebooks), nous n'avons pas poursuivi plus en profondeur le tuning du modèle à ce jour. Une piste que nous souhaitons encore explorer est l'augmentation plus intense des données (plus de transformations) comme suggéré dans la référence sur le training des ViT.

De plus, nous avons écarté ce modèle par manque de temps mais une exploration avec des modèles hybrides CNN-ViT ou des architectures plus récentes en utilisant PyTorch pourrait mettre ce type de modèle en avant.

Description des travaux réalisés

Répartition de l'effort sur la durée et dans l'équipe

Le diagramme de Gantt en annexe offre une description détaillée de la répartition des tâches et de l'implication des membres de l'équipe.

Bibliographie

Modèles:

"[2010.11929] An Image is Worth 16x16 Words: Transformers - arXiv." <https://arxiv.org/abs/2010.11929>. Accessed 27 Sep. 2021.

"Keras implementation of ViT (Vision Transformer) - GitHub." <https://github.com/faustomorales/vit-keras>. Accessed 27 Sep. 2021.

"How to train your ViT? Data, Augmentation, and Regularization in" 18 Jun. 2021, <https://arxiv.org/pdf/2106.10270>. Accessed 28 Sep. 2021.

"EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" (ICML 2019) <https://arxiv.org/abs/1905.11946>. Last revised 11 Sep 2020.

"Xception: Deep Learning with Depthwise Separable Convolutions" (CVPR 2017) <https://arxiv.org/abs/1610.02357>. Last revised 4 Apr 2017.

Classification de cellules sanguines :

"White blood cells identification system based on convolutional deep" <https://www.sciencedirect.com/science/article/pii/S016926071730411X>. Accessed 26 Sep. 2021.

"Deep learning approach to peripheral leukocyte recognition - PLOS." 25 Jun. 2019, <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0218808>. Accessed 26 Sep. 2021.

"Fine-grained leukocyte classification with deep residual learning for" <https://www.sciencedirect.com/science/article/pii/S0169260718300567>. Accessed 26 Sep. 2021.

"White blood cells detection and classification based on regional" <https://www.sciencedirect.com/science/article/pii/S0306987719310680>. Accessed 26 Sep. 2021.

"Detection of subtype blood cells using deep learning - ScienceDirect." <https://www.sciencedirect.com/science/article/pii/S1389041718303760>. Accessed 26 Sep. 2021.

"White blood cell differential count of maturation stages in ... - PLOS." 11 Dec. 2017, <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189259>. Accessed 26 Sep. 2021.

"Recognition of peripheral blood cell images using convolutional" <https://www.sciencedirect.com/science/article/pii/S0169260719303578>. Accessed 26 Sep. 2021.

"Human-level recognition of blast cells in acute myeloid leukaemia." 12 Nov. 2019, <https://www.nature.com/articles/s42256-019-0101-9>. Accessed 26 Sep. 2021.

"Generalizability in White Blood Cells' Classification Problem | bioRxiv." 28 Jul. 2021,
<https://www.biorxiv.org/content/10.1101/2021.05.12.443717v3>. Accessed 26 Sep. 2021.

"Colour deconvolution: stain unmixing in histological imaging - PubMed." 16 Jun. 2021,
<https://pubmed.ncbi.nlm.nih.gov/32997742/>. Accessed 26 Sep. 2021.

Datasets:

"A dataset of microscopic peripheral blood cell images for ... - NCBI." 8 Apr. 2020,
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7182702/>.

"Fast and robust segmentation of white blood cell images by self"
<https://www.sciencedirect.com/science/article/pii/S0968432817303037>. Accessed 18 Sep. 2021.

"Raabin-WBC: a large free access dataset of white blood ... - bioRxiv." 29 May. 2021,
<https://www.biorxiv.org/content/10.1101/2021.05.02.442287v4>. Accessed 26 Sep. 2021.

"A Single-cell Morphological Dataset of Leukocytes from AML" 17 Feb. 2021,
<https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=61080958>. Accessed 26 Sep. 2021.

"Deep Learning for Distinguishing Morphological Features of Acute" 5 Nov. 2020,
<https://ashpublications.org/blood/article/136/Supplement%201/10/472209/Deep-Learning-for-Distinguishing-Morphological>.

Difficultés rencontrées lors du projet

Le principal verrou scientifique rencontré lors du projet reste la qualité et la disponibilité des **données** :

- **les datasets sont déséquilibrés** : certains types de cellules comme les éosinophiles (EO), les neutrophiles (surtout SNE) sont sur-représentés par rapport à d'autres catégories plus difficiles à trouver²⁴ et que le modèle a du mal à classer (PMY, MMY, MY, en faible nombre et difficiles à reconnaître à l'œil nu, pour un observateur peu expérimenté ; il existe de plus une controverse scientifique sur la pertinence de distinguer ou non ces catégories). **Ce déséquilibre s'explique cependant par des raisons biologiques** : certains types de cellules, comme par exemple les SNE, sont présentes en quantités plus importantes que d'autres dans le sang. La présence accrue de cellules immatures (comme les BNE ou les PMY/MMY/MY) peut être considérée comme révélatrice d'infections ou d'inflammations, ce qui explique la faible quantité de ces cellules chez des patients sains ;
- **le problème des labels** : nous avons des doutes quant aux labels de certaines images (notamment lorsqu'il s'agit des SNE des BNE, qui sont en fait la même cellule mais observée à deux instants différents de sa vie). Ces doutes sont nourris par les résultats de nos modèles, mais aussi par ce que nous avons pu lire dans la littérature scientifique. On citera notamment la Table n°4 de l'article de recherche présentant le dataset Raabin, où les auteurs montrent les résultats de l'identification des images par deux experts indépendants ne s'étant pas consultés²⁵ : à titre d'exemple, près de 30% des images classées comme BNE par un expert ont été classées comme SNE (le terme exact est Neutrophil dans l'article) par l'autre. Plusieurs milliers d'images ont été classées comme "Not labelled" ou "Not recognized", attestant de la difficulté de l'exercice pour un observateur humain, même chevronné ;
- **finalement, assez peu de sources de données différentes**, ce qui pose de gros **problèmes en termes de généralisation**. Beaucoup de travaux dans le domaine de la reconnaissance des cellules sanguines par du deep-learning mettent en avant des modèles obtenant des performances excellentes, mais utilisent des données provenant d'une seule origine. Lorsque nous avons testé la première version du modèle VGG19, entraîné sur le dataset Barcelone Normal, nous obtenions une accuracy de plus de 95% pour la plupart des classes de cellules (notamment 99,8% sur les EO). Lorsque ce même modèle a été testé sur les données du dataset Raabin, l'accuracy globale du modèle a littéralement plongé (passage de 92,6% à 47,4%, en tombant à 37,7% pour les seuls EO de Raabin).

Par souci d'éviter de construire un modèle trop spécialisé, nous avons cherché à compléter notre dataset. Cela nous a conduit à affronter des difficultés d'ordre prévisionnel :

²⁴ On peut citer les PMY, MMY, MY qui ne sont pas présents dans tous les datasets, ainsi que les BNE, eux aussi en faible nombre et parfois regroupés avec les SNE sous l'appellation générale "Neutrophile".

²⁵ "Raabin-WBC: a large free access dataset of white blood ... - bioRxiv." 29 May. 2021, <https://www.biorxiv.org/content/10.1101/2021.05.02.442287v4>. Accessed 29 Sep. 2021.

- Le dataset AML_APL_John-Hopkins a été exploré mais n'a pas été utilisé à cause d'un **mauvais labelling**.
- **D'autres datasets ne proposaient que des images ayant subi une procédure de segmentation :** l'image avait été découpée autour des cellules, pour en éliminer le fond contenant les globules rouges. Il nous a paru peu pertinent de mélanger ces images avec celles dont nous disposions et pour lesquelles les globules rouges étaient présents. De plus et comme nous l'avons souligné dans l'introduction de ce document, bien que les systèmes de reconnaissance existants des globules blancs utilisent la segmentation, **ce type de méthode est difficile à généraliser** : variabilité des colorations utilisées pour traiter les échantillons de sang, variabilité des systèmes d'acquisition. L'utilisation d'un modèle reposant sur des CNN permet potentiellement de contourner ce problème, pourvu que les données disponibles soient suffisamment diverses.
- Le **téléchargement des datasets** d'une part, puis leur **transfert sur Google Drive** d'autre part, a également posé plusieurs problèmes : des problèmes de temps, les durées de téléchargement ayant été parfois très longues (jusqu'à plusieurs jours pour les transferts sur Drive), des problèmes de connexion internet, des problèmes de capacité de stockage sur Drive.

Notre volonté d'explorer des sentiers parfois non balisés par la formation nous a conduit à prendre du temps pour acquérir un certain nombre de compétences, via la documentation des bibliothèques utilisées et grâce à des cours et tutoriaux trouvés sur internet. On citera notamment :

- **la familiarisation avec Google Colab**, pour le transfert des données depuis le Drive et l'utilisation du calcul sur GPU. Il aurait bien d'avoir une machine virtuelle à partager pour le cloud computing;
- **la découverte et l'implémentation de CAM / Grad-CAM** pour analyser le comportement du modèle, élément absent de la formation;
- l'entraînement du **ViT** a été plus difficile qu'envisagé au premier abord et a demandé beaucoup de recherches bibliographiques. De plus, le manque de connaissances sur les **transformateurs et PyTorch** a ralenti le développement de ce modèle. Avec le recul, nous pensons qu'il aurait été **pratique d'avoir eu accès au module 152 sur les CNN** avec PyTorch, en plus des 151 et 155 qui nous ont été proposés.

Bilan & Suite du projet

En utilisant une approche combinatoire avec des sources variées de données, nous avons pu obtenir un modèle capable de classifier plus de 11 classes de leucocytes circulants ou immatures. Ce type de modèle pourrait être utilisé comme base pour un fine-tuning et une classification d'autres types cellulaires ou de cellules leucémiques.

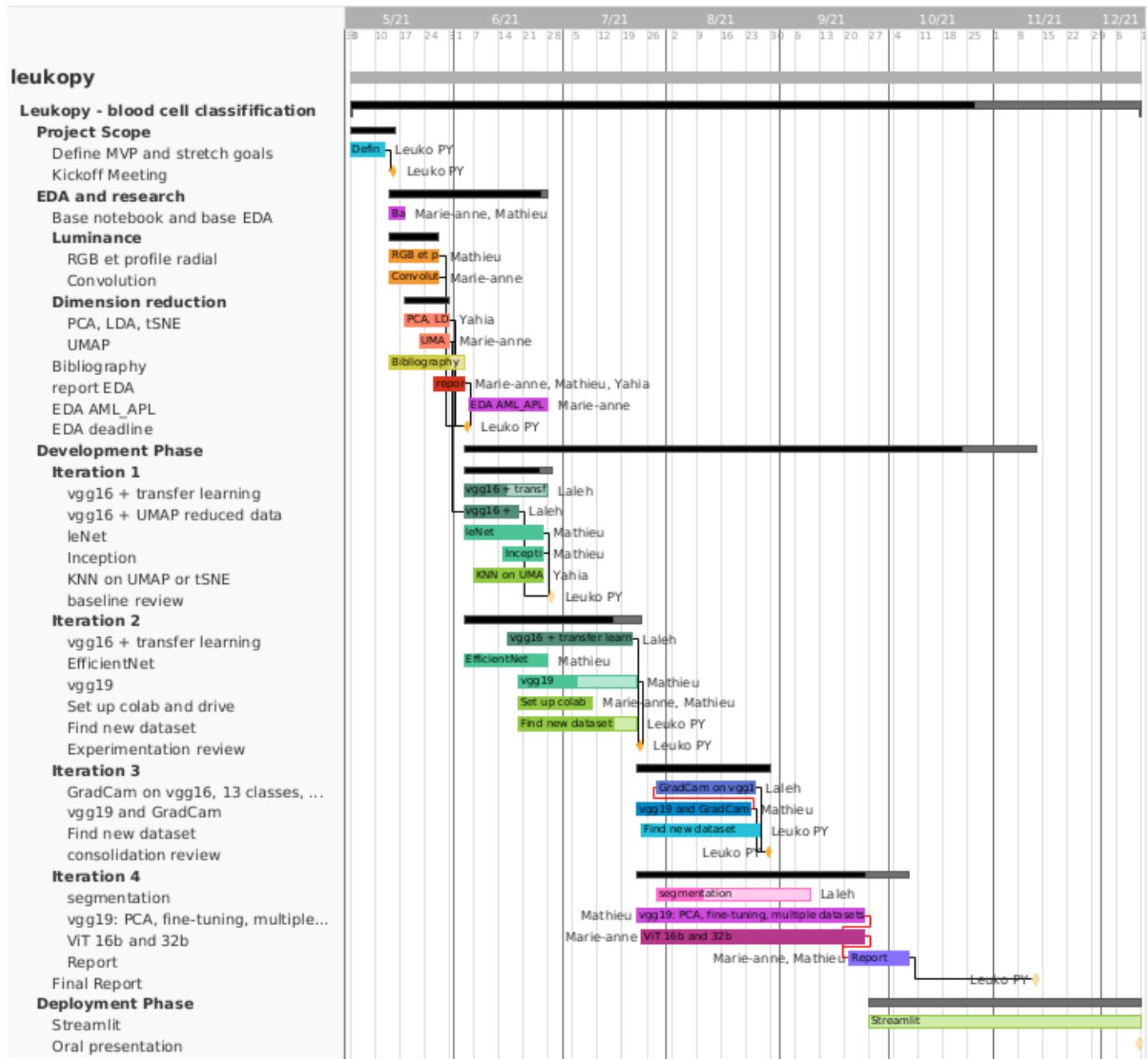
Notre objectif principal de classer les cellules saines circulantes a été atteint et nous avons pu développer une analyse poussée de l'interprétabilité de nos modèles. Nous sommes confiants quant à notre capacité à construire un outil utile et accessible pour les experts pathologistes, leur permettant d'identifier les cellules sur des frottis sanguins plus efficacement et avec plus de confiance.

De nombreuses améliorations sont cependant encore possibles et souhaitables. Voici plusieurs pistes qu'il reste à explorer ou à approfondir :

- **Diversifier davantage la provenance des images du dataset**, dans l'optique de construire un modèle généraliste, susceptible de fournir des prédictions correctes dans des hôpitaux et laboratoires différents. **Développer l'augmentation de données** peut aussi contribuer à cet objectif.
- L'utilisation de **GANs** serait un moyen de générer de nouvelles images, notamment pour les classes de cellules saines en faible effectif.
- **S'assurer de la qualité des labels des images utilisées**, par exemple à travers un partenariat avec des spécialistes qui évaluerait indépendamment chaque image, dans le but de ne conserver que celles faisant consensus.
- Avec davantage de temps et de moyens, il serait possible d'abandonner le transfer-learning et de construire un modèle depuis le départ. On pourrait avoir recours à des méthodes d'ensemble à partir des features extraites par le réseau neuronal pour améliorer les performances.

Annexes

Diagramme de Gantt



Description des fichiers de code

notebooks

```
    └── leukopy_lib.py: helper functions
    └── cleaning
        ├── generate_dataframes.ipynb: notebook pour generer les datasets de training, validation et test.
        ├── Importer_PBC_AML_APL.ipynb: nettoyage et EDA rapide du dataset AML_APL (exclu)
        └── Importer_PBC_normal.ipynb: nettoyage et EDA rapide du dataset de barcelone
    └── EDA
        ├── First_EDA_Barcelona.ipynb: premières explorations de dataset de barcelone
        ├── Dimensionality reduction (PCA, umap...).ipynb:
        ├── EDA_deconv_hist.ipynb; deconvoloution
        ├── EDA_mean_color.ipynb: exploration des niveaux de RGB et profil radial
        ├── EDA_mean_gray.ipynb: exploration des niveaux de gris
        ├── EDA_mean_gray_UMAP.ipynb: to clean?
        ├── EDA_PCA_LDA_TSNE_V3-3.ipynb: to clean?
        ├── EDA_PCA_LDA_TSNE_V3.ipynb: to clean?
        ├── EDA_PCA_V1.ipynb: to clean?
        └── EDA_PCA_V2.ipynb: to clean?
    └── model
        ├── Dimension_red_KNN.ipynb: test de dimension de reduction et KNN en classifieur
        └── efficientNet
            ├── EfficientNet_subclasses.ipynb
            ├── EfficientNet_subclasses_test_tf_dataset.ipynb
            ├── EffNetB0_model.ipynb
            └── EffNetB3_GPU.ipynb
        └── lenet
            └── leuko_lenet_gc.ipynb
        └── vgg16
            ├── Models_151_Transfer_Learning_class_rep.ipynb
            ├── Models_151_Transfer_Learning_LR.ipynb
            ├── Models2_151_Transfer_Learning_LR.ipynb
            ├── Model_Transfer_Learning_13classes.ipynb
            ├── Model_Transfer_Learning_9classes.ipynb
            ├── Model_Transfer_Learning_9classes_Suit2.ipynb
            └── Model_Transfer_Learning_9classes_Suite.ipynb
        └── vgg19
            ├── VGG19_TL_11.ipynb
            └── vgg19_tl_11.py
    └── ViT
        ├── Vit_b16.ipynb
        └── ViT_b32.ipynb
```