Projet PyCell

Léo Bariseel, Xavier Guilbert, Omar Benbrahim, Benyamin Ittah

2 novembre 2021

Ce rapport technique aborde la mise en place et l'optimisation de modèle de Deep Learning dans le cadre de la détection de leucémie. Il y'a dans l'ordre le contexte du projet, les objectifs posés, les jeux de données utilisés, les modèles abordés et leur optimisation, l'analyse des résultat obtenus, qui nous amène à la sélection du meilleur modèle répondant à notre problématique.

Table des matières

1 Introduction

1.1 Contexte général

Dans le cadre de ce projet, nous mettons en place un algorithme capable de déterminer la présence ou non de leucémie à partir de cliché de cellules. Parmis les 4 principaux types de leucémies existantes, nous en avons choisis 2, qui sont les types AML (Leucémie aiguë myéloblastique) et APL (Leucémie aiguë promyélocytaire).

Étant dans un contexte médical, l'erreur de prédiction est plus difficilement admise. Nous choisissons donc de minimiser les faux négatifs (patients malades détectés en tant que patients sains).

1.2 Point de vue technique

S'agissant de prédictions à partir d'images, et ayant une base de données conséquente à disposition, les modèles de Deep Learning se sont rapidement imposés dans le cadre du projet.

1.3 Enjeux du projet

Ce projet permet d'une part de gagner du temps retournant au medecin les clichés de cellules étant atteints de leucémie, il pose aussi les base pour le développement d'algorithme similaires de Deep Learning ainsi que leur optimisation pour la résolution d'un problème dans le même contexte, qui est la reconnaissance d'image.

2 Objectifs

2.1 Liste des objectifs

Ci-dessous sont les objectifs que nous devons atteindre, organisés par ordre chronologique :

- Etudes des données et du problème (documentation sur la leucémie...)
- Mise en relation avec des experts du domaine
- Traitement des données (Nettoyage et préparation des données...)
- Mise en application de différents modèles de Deep Learning
- Analyse et comparaison des performances
- Recherche d'intelligibilité et optimisation des modèles
- Choix du modèle optimal

2.2 Connaissances du groupe sur le domaine

Aucun des membres du groupe n'étant qualifié dans le domaine médical, nous avons alors contacté des personnes de ce milieu afin de mieux comprendre leur démarche pour aborder ce genre de problème.

2.3 Recherche de sujets similaires

Par exemple en radiologie, il existe des algorithmes d'aide au diagnostic mettant en avant les zones où une erreur est détecté suggérant au médecin de se concentrer dessus. Le fonctionnement de ce type de modèle nous pousse à mettre en place un mécanique d'interprétabilité du modèle.

3 Data

3.1 Jeux de données utilisés

Plusieurs jeux de données disponibles sur internet ont été explorés, chacun d'entre eux contenant une banque d'images de cellules.

Cependant, certains de ces jeux se sont révélés inexploitables pour notre projet, par exemple si la taille n'est pas assez importante ou bien si les images ne sont pas labélisés, nous ne pouvons alors pas distinguer une image de patient malade d'une image de patient sain.

Nous avons établis les conditions suivantes pour choisir nos jeux de données :
- Les images doivent provenir d'un organisme de confiance
- Les images doivent être labélisés
- Les images doivent être en quantitée suffisante.

Finalement, 2 jeux de données ont été utilisés :

Images de patients sains :
- Fourni par l'hôpital de Barcelone, Espage
- Disponible via https://data.mendeley.com/datasets/snkd93bnjr/1
- 17 000 images

Images de patients malades (AML/APL) :
- Suggéré par DataScientest
- Disponible via https://www.kaggle.com/eugeneshenderov/acute-promyelocytic-leukemia-apl
- 26 000 images

Note : Les colorations des cellules sur les images sont obtenues par coloration de May-Grünwald Giemsa qui colorie les cellules et en particulier leur noyaux de bleu à violet. Les hématies (globules rouges), les cellules les plus nombreuses et que l'on voit en arrière plan de nos images ne se colorient pas et restent dans une teinte beige-rose et sont donc facilement distinguables des cellules que nous étudions.

3.2 Images des jeux de données

Nous affichons ci-dessous quelques images de jeux de données des patients sains, prises au hasard.

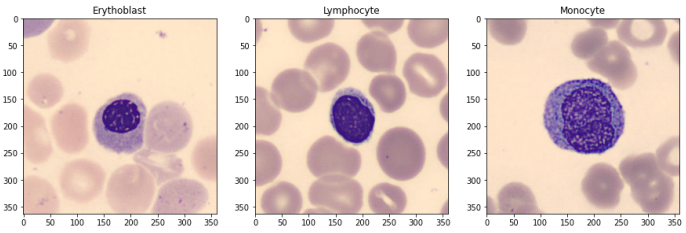

FIGURE 1 – Images de cellules saines.

Nous affichons ci-dessous quelques images du jeux de données des patients malades, prises au hasard.

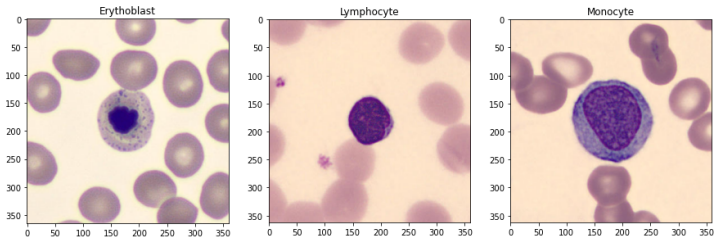

FIGURE 2 – Images de cellules malades.

Nous pouvons voir que les deux jeux de données possèdent les mêmes tons de couleurs, ce qui permet d'éviter des biais de classification.

Quelques images corrompues empêchant l'entrainement de notre modèle ont été supprimées. Ces images sont minoritaires, de l'ordre d'une dizaine sur les 45 000 de notre jeu de données. Ci-dessous nous affichons une image corrompue :

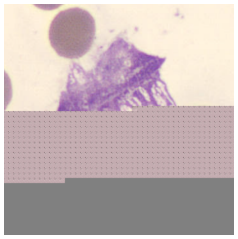

FIGURE 3 – Image corrompue.

3.3 Colorimétrie des jeux de données

Afin d'éviter des biais lors de l'entrainement de nos algorithmes, il faut s'assurer de l'homogénéité de nos images. Pour cela nous avons regardé d'un côté la distribution des couleurs, et d'un autre côté la netteté des images.

Pour la netteté, nous avons utilisé la somme des valeurs absolues des Laplacien : lorsqu'une image est nette, il y'a une grande variation entre chaque pixels, qui est traduit par une valeur élevée du Laplacien, qui peut être positifs ou négatifs. La somme de ces variations se traduit donc par la somme des valeurs absolues des Laplaciens.

Nous avons regroupés ces données dans les graphiques ci-dessous, sur la première ligne nous avons une analyse sur la distribution des couleurs, sur la deuxième ligne nous avons une analyse sur la nettetée des images.

La colonne de gauche représente les patients malades et la colonne de droite représente les patients sains. Notons que pour les patients malades, nous avons fais les calculs sur 8000 images par manque de mémoire, mais nous voyons tout de même une homogénéité des images qui est similaires à l'autre jeu de données.

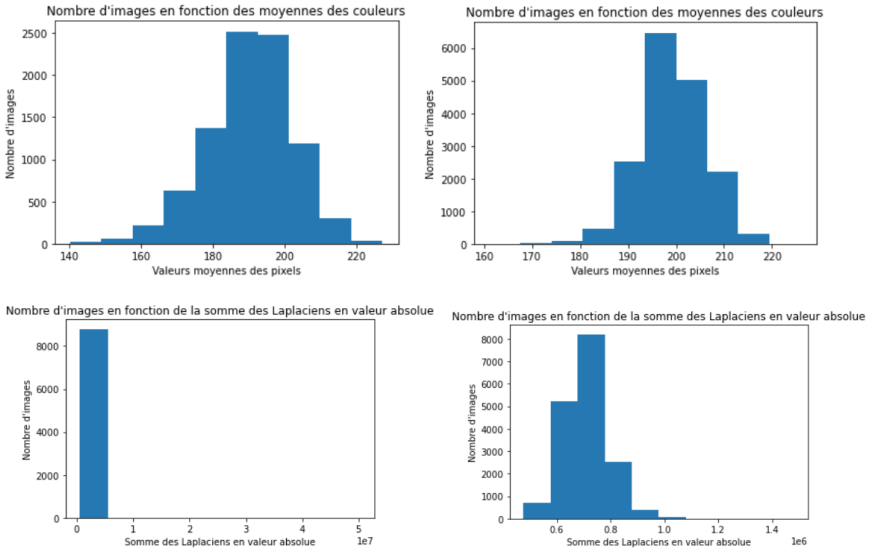

FIGURE 4 – Distributions de couleurs et nettetés des images des deux jeux de données.

3.4 Preprocessing des jeux de données

Afin d'avoir des données exploitables, nous avons du s'assurer que les images remplissent certains critères :

- Sélection d'images avec des colorimétries similaires
- Localisation et suppression des fichiers corrompus
- Suppression des images de groupe de dimensions différentes - via analyse de la taille du fichier

3.5 Variable cible

Notre objectif est de détecter la présence de leucémie, nous avons également inclus dans notre algorithme la détection du type de leucémie AML et APL.

Les variables cibles sont les suivantes :

- Patient sain
- AML
- APL

4 Projet

4.1 Définition du projet

4.1.1 Problème de Deep Learning : Classification

Nous souhaitons développer des modèles de classification supervisés sur la reconnaissance d'image. Les banques d'images labélisées constituent la base d'entraînement de nos modèles, qui permettent d'établir un diagnostique du patient.

4.1.2 Métrique de performance utilisée

Nous cherchons la bonne détection des patients malades. En effet un faux négatif peut avoir de graves répercussions en retardant la détection ou la guérison du patient. Un diagnostique positif est complété par un second examen, qui élimine les faux positifs.

Dans cette optique, nous choisissons la précision pour évaluer la performance de nos modèles dans un premier temps. Ensuite, nous utilisons le rappel pour réduire la proportion de faux négatifs si plusieurs modèles ont une précision équivalente.

4.2 Choix du modèle et optimisation

4.2.1 Modèles de Deep Learning

Nous avons commencé par regarder 3 différents modèles de Deep Learning : LeNet, CNN et un modèle de transfert learning avec une base VGG16.

Dans les graphiques ci-dessous, nous regroupons les précisions et fonctions de perte de ces différents modèles :

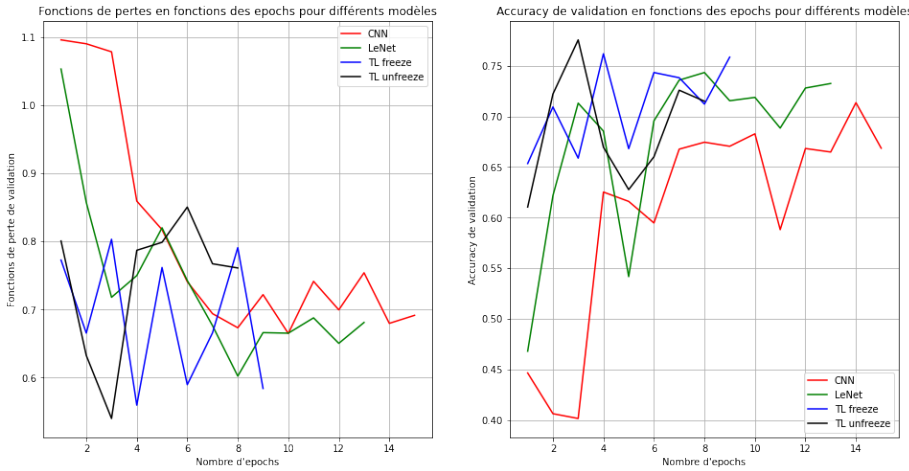

FIGURE 5 – Précisions et fonctions de perte de différents modèles de Deep learning.

Nous pouvons voir que le modèle CNN est derrière, ce que nous permet de le retirer de nos options. Les modèles LeNet et le modèle de transfert learning sont similaires en terme de précision, cependant on peut remarquer une différence importante de vitesse d'apprentissage des modèles. Cette première itération nous donne une précision de 75%.

4.2.2 Optimisation de fonctions d'activation

Nous avons optimisé le modèle LeNet en modifiant les optimizers ou bien les fonctions d'activation, ce qui nous permet de gagner un peu de précision et de monter à 80%. Nous avons regroupé dans le graphique suivant les précisions obtenues selon les différentes fonctions d'activation.

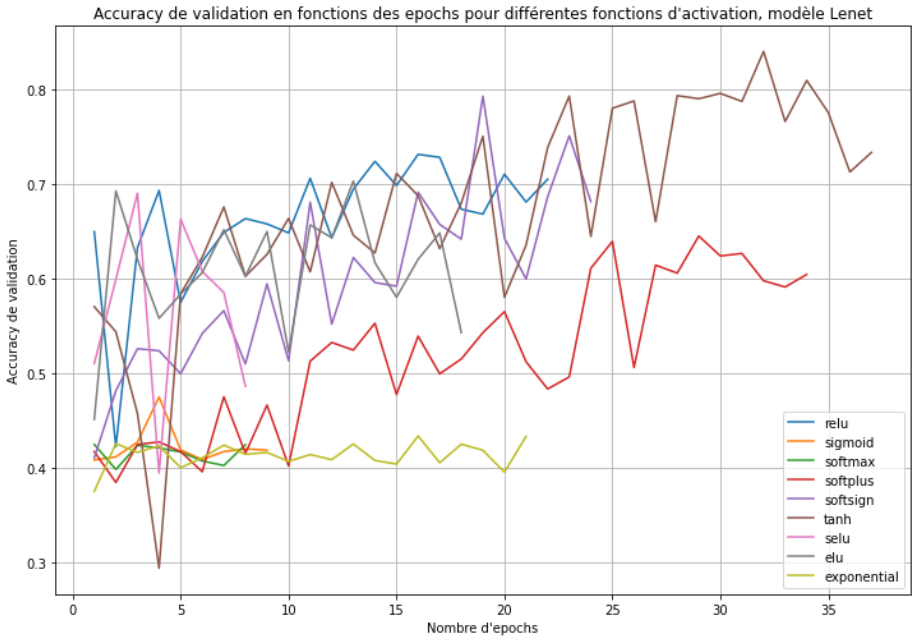

FIGURE 6 – Précisions du modèle LeNet selon différentes fonctions d'activation.

Les modèles de transfert learning offrent une bonne performance avec un temps d'apprentissage considérablement plus faible, nous choisissons cette option pour la suite de notre projet. En effet avec une base de donnée importante, les temps d'apprentissage peuvent s'étendre à plusieurs jours, ce qui rend cette caractéristique importante.

4.2.3 Modèles de transfert learning

Nous avons ensuite entraîné d'autres modèles de transfert learning en utilisant différentes bases disponibles depuis keras : EfficientNet, DenseNet, Inception, NasNet, MobileNet et ResNet. Étant donné le nombre important de couches cachées des modèles de transfert learning, nous n'avons pas pu optimiser les fonctions d'activations de celles-ci.

Nous avons regardé en premier lieu la précision des différents modèles pour avoir une idée grossière des algorithmes les plus performants, puis nous avons regardé les matrices de confusion pour choisir le meilleur modèle, en minimisant le nombre de patient malades classé en tant que non malade.

Nous regroupons dans le graphique ci-dessous les précisions des différents modèles étudiés, ayant une précision allant de 70% jusqu'à 95%. Les meilleurs modèles étant le ResNet101, l'EfficientNetB0 et l'EfficientNetB1 avec une précision similaire.

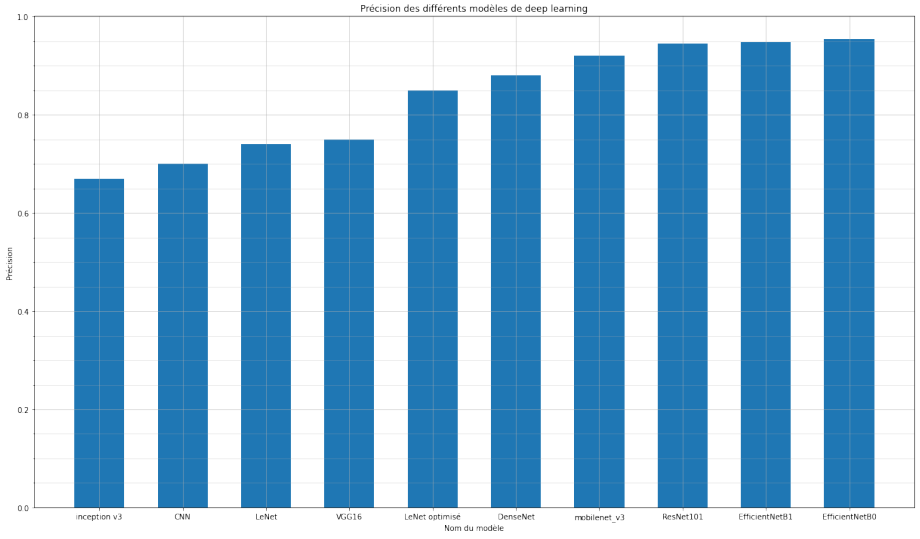

FIGURE 7 – Précisions de différents modèles de transfert learning.

Notons que nous n'avons pas pu essayer tous les modèles à cause du manque de mémoire, comme par exemple le modèle NasNet, le ResNet152 et les modèles EfficientNetB2 à EfficientNetB7.

Pour le jeu de validation, nous avons environ 7000 images, dont 2000 de chaque type de leucémie et 3000 de patients sains.

Trois modèles sortaient du lot en regardant leurs précisions :
- EfficientNetB0 : 95,38% de précision.
- EfficientNetB1 : 94,88% de précision.
- ResNet101 : 94,47% de précision.

Pour différencier ces modèles, nous regroupons les informations de ces 3 modèles sous la forme de matrices de confusion : dans l'ordre nous avons le modèle EfficientNetB0, EfficientNetB1 et le modèle ResNet101.

Classe réelle \ prédiction	AML	APL	Patient sain
AML	2014	107	3
APL	197	1786	5
Patient sain	3	2	2757

FIGURE 8 – Matrice de confusion du modèle EfficientNetB0.

Classe réelle \ prédiction	AML	APL	Patient sain
AML	1975	148	1
APL	190	1798	0
Patient sain	8	5	2749

FIGURE 9 – Matrice de confusion du modèle EfficientNetB1.

Classe réelle \ prédiction	AML	APL	Patient sain
AML	1967	156	1
APL	214	1773	1
Patient sain	4	4	2754

FIGURE 10 – Matrice de confusion du modèle ResNet101.

4.2.4 Choix du modèle

En s'intéressant au nombre de patient malades classés en tant que patients malades, les différents modèles nous donnent les prédictions suivantes :

- EfficientNetB0 : 8 patients malades classés en tant que patients sains
- EfficientNetB1 : 1 patients malades classés en tant que patients sains
- ResNet101 : 2 patients malades classés en tant que patients sains

Comme nous voulons minimiser le nombre de faux négatifs, nous choisissons le modèle EfficientNetB1, bien qu'il n'ait pas la meilleur précision, il permet une meilleur détection de patients malades.

Nous illustrons les prédictions retournées sur une image au hasard de notre banque de données de l'ensemble de validation :

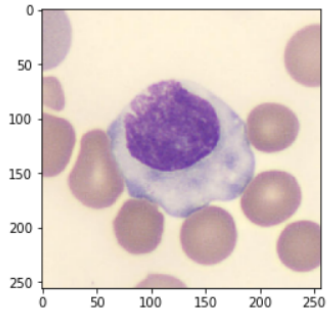

FIGURE 11 – Image d'une cellule de l'ensemble de validation.

Cette cellule est de la classe AML d'un patient malade. Les prédictions retournées par notre modèle sont les suivantes :

- 0.99472% de chance que ce soit une cellule AML
- 0.00525% de chance que ce soit une cellule APL
- 0.0000225% de chance que ce soit une cellule saine

Comme nous pouvons le voir, notre modèle fais bien la distinction dans la classification de patients malades et sains.

Nous avons également entrainés un modèle sur un type de cellule spécifique, ce qui nous a permis d'augmenter un peu la précision et de monter à 99%.

Nous nous sommes également intéressés aux images que notre modèle a mal classé entre patients sains et patients malades. Sur la figure suivante nous regroupons des erreurs de prédictions pour les trois classes : AML, APL et patients sains.

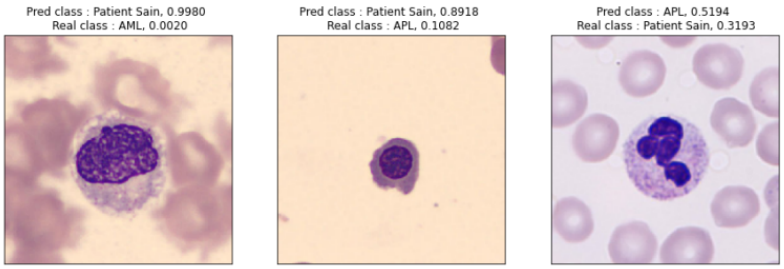

FIGURE 12 – Image de cellules mal classés de l'ensemble de validation.

Nous pouvons voir que sur les deux premières images, l'algorithme se trompe totalement de classe, mais il peut y'avoir aussi le cas où les prédictions sont assez proches comme dans la dernière image. Cela suggère une limite de choix de classe en prenant la probabilité la plus élevée.

4.3 Analyse des biais

Dans cette section, nous effectuons des modifications sur une image de classe AML afin de visualiser l'impact sur les probabilités associées.

Les modifications apportés sont les suivantes :

- Cacher la cellule malade
- Ajouter des formes aléatoires
- Isoler des cellules et changer le fond
- Isoler la cellule malade et appliquer un filtre
- Isoler la cellule malade

Ces modifications sont illustrées sur la figure suivante, la dernière image correspond à l'image originale, sans modification. Nous affichons également la classe prédite ainsi que la classe de la cellule.

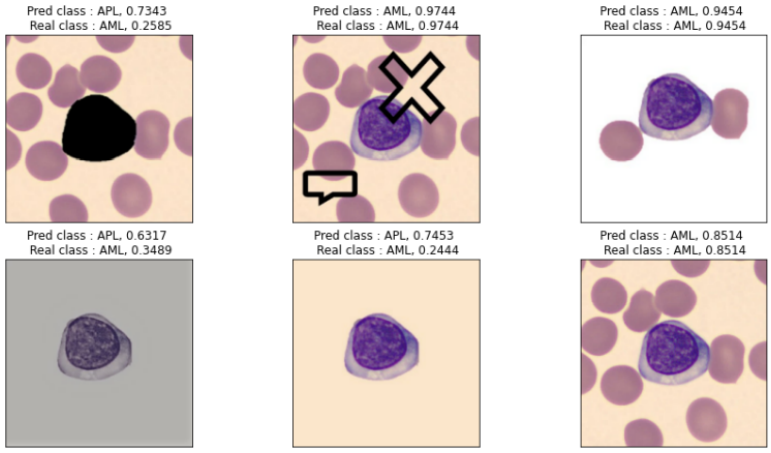

FIGURE 13 – Images augmentées d'une cellule de classe AML avec prédictions associées

On remarque que en appliquant des augmentations, les classes prédites sont satisfaisantes en donnant toujours des cellules malades. Toutefois il peut se tromper en donnant le mauvais type de leucémie après avoir appliqué un transformation.

Nous avons développé un algorithme de Deep Learning basé sur la reconnaissance d'image afin de détecter la leucémie chez des patients. Une précision de 95% a été atteinte, mais notre modèle a tout de même du mal à différencier les différents types de leucémie. Avec environ 10% d'erreur entre les classes AML et APL, mais les patients sont tout de même classés en tant que malades .

5 Conclusion

Nous avons sélectionné le modèle EfficientNetB1 pour sa bonne précision et son rappel, qui est le modèle classant le moins de patients malades en tant que non malades. Notre modèle possède toutefois des difficultés à différencier les différents types de leucémie.

6 Axes d'amélioration

Afin d'améliorer notre modèle, nous suggérons l'implémentation de visualisation des zones que regarde l'algorithme pour la classification. Pour cela, plusieurs méthodes sont disponibles :

6.1 Grad-CAM et Grad-CAM++

Le Grad-CAM crée une carte thermique à partir d'une carte d'activation pondérée par le gradient. Cette carte permet de voir quelle partie de notre image est retenue par le modèle pour faire son choix.

Grad-CAM ++ est similaire à Grad-CAM mais utilise des dérivées secondes au lieu du gradient, cette méthode en meilleure et est plus adaptée aux images de groupe.

6.2 LIME

A partir d'une image segmentée en plusieurs plus petites comme un puzzle, LIME modifie les données d'une pièce avant d'utiliser le modèle. De la même manière que pour le Grad-CAM, on peut ainsi facilement visualiser par une heatmap les zones les plus déterminantes pour le modèle.

7 Bibliographie

Information générale sur la leucémie :
https://fr.wikipedia.org/wiki/Leuc%C3%A9mie

Exemple de projets similaire : Radiologie des poumons
https://towardsdatascience.com/bias-in-your-datasets-covid-19-case-study-d065aa698b74

Détection de cancer de la peau par un algorithme CNN :
https://www.kaggle.com/fanconic/cnn-for-skin-cancer-detection

Intelligibilité des modèles avec LIME :
https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b

Intelligibilité des modèles avec Grad-CAM :
https://keras.io/examples/vision/grad_cam/
https://datascientest.com/grad-cam

Intelligibilité des modèles avec Grad-CAM++ :
https://www.researchgate.net/publication/320727679_Grad-CAM_Generalized_
Gradient-based_Visual_Explanations_for_Deep_Convolutional_Networks

Détection d'objet :
https://public.roboflow.com/object-detection/bccd