IMA205 – Kaggle Challenge Report
White Blood Cell Classification

Sacha Ryszfeld
Télécom Paris

Avril 2026

Résumé

Ce rapport présente la démarche suivie pour le challenge de classification d'images de globules blancs en 13 classes, évalué au *macro*-F1. Le jeu de données est fortement déséquilibré (ratio 1 183× entre la classe la plus fréquente et la plus rare) et seuls les modèles pré-entraînés sur ImageNet sont autorisés. On part d'une approche classique à features handcrafted, puis on construit progressivement un pipeline profond : correction du déséquilibre par sur-échantillonnage, choix du backbone ConvNeXt-Tiny, classifieur final XGBoost sur embeddings, et validation par 5-fold stratifié. Le pipeline final obtient un *macro*-F1 moyen de $0{,}6935 \pm 0{,}0375$ en validation croisée, et la prédiction test est produite par moyenne des probabilités sur les cinq folds.

1 Introduction

Le challenge consiste à classer 28 901 images de globules blancs en 13 classes, évalué au *macro*-F1. Les labels ne sont fournis que sur le train, et seuls les modèles pré-entraînés sur ImageNet sont autorisés.

Dans ce rapport, on :

— présente une approche classique à base de features handcrafted comme point de comparaison,
— détaille la construction progressive d'un pipeline profond,
— discute la logique des choix successifs et les limites observées.

2 Description du dataset

Le dataset contient 28 901 images RGB de globules blancs, labellisées selon 13 catégories : `SNE, LY, MO, BL, EO, BA, MY, BNE, VLY, MMY, PMY, PC, PLY`. 9 634 images supplémentaires forment le test (labels non fournis). Le score Kaggle public est calculé sur 30 % du test et disclosé immédiatement ; les 70 % restants forment le score privé, révélé à la clôture du challenge, pour éviter l'overfit au leaderboard.

La distribution des classes en train est très *long-tail* (Fig. 1, Tab. 1). Quatre classes ont moins de 500 exemples et trois moins de 120. Le ratio entre la classe la plus fréquente (SNE, 13 015) et la plus rare (PLY, 11) est de 1 183. PLY pose un problème structurel : aucun modèle supervisé standard ne peut apprendre de frontière fiable sur si peu de données.

Table 1 – Effectifs par classe.

Classe	n	Classe	n	Classe	n
SNE	13 015	BL	2 012	BNE	391
LY	8 101	EO	861	VLY	366
MO	2 746	MY	441	MMY	360
		BA	415	PMY	114
				PC	68
				PLY	11

Le choix du *macro*-F1 comme métrique est central. Contrairement à l'accuracy, il donne le même poids à chaque classe : une classe rare mal prédite pèse autant qu'une classe fréquente. Un modèle prédisant uniquement SNE et LY en ignorant les autres classes atteindrait déjà $\sim 73\,\%$ d'accuracy tout en étant pratiquement inutilisable. Le traitement des classes minoritaires est donc le cœur du problème.

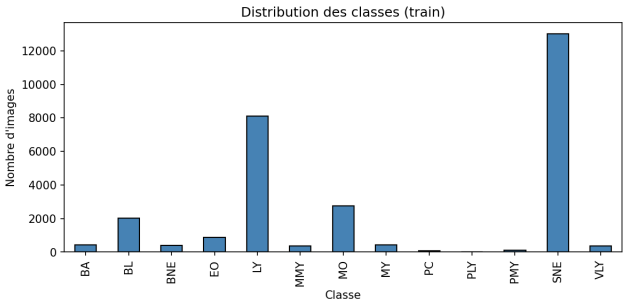


Figure 1 – Distribution des 13 classes dans le train.

Quatre questions se posent : quelle représentation des images choisir ? comment traiter le déséquilibre ? comment éviter l'overfitting avec des classes si rares ? et quel classifieur final utiliser ?

3 État de l'art

Plusieurs familles de techniques sont pertinentes ici. Les **architectures convolutives pré-entraînées sur ImageNet** forment la base du transfer learning pour les images de microscopie : ResNet [1] est le standard historique, SE-ResNet [2] ajoute un module d'attention par canal, et ConvNeXt [3] combine les conventions des Transformers avec les avantages des ConvNets.

Pour la classification *long-tail*, plusieurs méthodes existent : la pondération simple de la loss (inverse de fréquence), le sur-échantillonnage pondéré, la Balanced Softmax [5] qui corrige le biais de fréquence au niveau des logits, ou LDAM-DRW [6] qui impose des marges plus grandes aux classes rares. Kang et al. [7] ont montré qu'il peut être bénéfique de découpler l'apprentissage de la représentation et celui du classifieur final. Enfin, XGBoost [8] est un classifieur à base d'arbres à boosting

souvent utilisé comme tête sur des embeddings profonds.

La suite s'appuie sur le transfer learning via ConvNeXt-Tiny, le sur-échantillonnage pondéré, et XGBoost sur embeddings. D'autres techniques (Balanced Softmax, LDAM-DRW, Cosine Head) ont été testées mais écartées, comme discuté plus loin.

4 Approche classique : features handcrafted

La première approche est une baseline interprétable sans deep learning. L'objectif n'est pas de rivaliser avec l'approche profonde, mais de cadrer la difficulté du problème et de voir quelles caractéristiques sont intrinsèquement discriminantes.

4.1 Segmentation

On segmente approximativement la cellule dans chaque image. La segmentation s'appuie sur la saturation HSV : les globules blancs colorés au May-Grünwald-Giemsa ont des noyaux très saturés et sombres, ce qui permet de les séparer du fond. On binarise la saturation par un seuillage d'Otsu, puis on applique des opérations morphologiques (*opening* et *closing* avec un élément structurant en disque de rayon 3, `remove_small_objects` avec `min_size`= 200) pour nettoyer le masque. On ne garde que la plus grande composante connexe. En cas d'échec, le masque par défaut couvre toute l'image.

4.2 Extraction de features

On extrait 35 features par image, réparties en quatre familles.

Morphologie (via `skimage.measure.regionprops`) : aire, périmètre, diamètre équivalent, excentricité, solidité, *extent*, aire remplie, circularité ($4\pi A/P^2$), *aspect ratio* de la bounding box, ratios d'aires, asymétries haut/bas et gauche/droite calculées par rapport au centroïde, irrégularité du contour ($P/2\sqrt{\pi A}$).

Couleur : moyennes et écarts-types sur les canaux R, G, B, H, S, V, calculés uniquement sur les pixels du masque.

Texture : descripteurs GLCM (distance 1, angle 0) – *contrast*, *homogeneity*, *energy*, *correlation* – plus l'entropie de Shannon et la variance d'intensité, restreintes au masque.

Taille : dimensions de l'image.

4.3 Modèles et protocole

Les features sont normalisées par `StandardScaler` et passées à quatre classifieurs : régression logistique, SVM à noyau RBF, Random Forest, MLP. Tout est encapsulé dans des `Pipeline` sklearn (scaler + modèle) pour éviter toute fuite de données lors de la validation croisée. Le tuning se fait par `GridSearchCV` avec un `StratifiedKFold(5)` scoré en *macro*-F1.

4.4 Résultats

Le Tab. 2 compare les quatre classifieurs dans leur configuration par défaut et après tuning par `GridSearchCV`. Le MLP tuné est le meilleur modèle avec un *macro*-F1 de validation de **0,4318**.

Table 2 – Résultats des classifieurs classiques sur la validation (split 80/20 stratifié). Les scores CV ne sont pas reportés en raison d'un problème de scoring lors de `cross_validate`.

Modèle	Val *macro*-F1
Logistic Regression	0,3175
SVM (RBF, défaut)	0,3023
Random Forest (n=200)	0,3620
MLP $(128, 64)$	0,4291
SVM tuned (C=0,1, γ=scale)	0,2181
RF tuned (max_depth= 10, n=100)	0,3103
MLP tuned (α=10^{-4}, hidden= $(64,)$)	**0,4318**

Deux observations contre-intuitives ressortent. D'abord, les versions tunées du SVM et du Random Forest sont *pires* que leurs versions par défaut. Pour le SVM, `GridSearchCV` converge vers $C = 0{,}1$, un réglage très régularisé qui écrase les classes minoritaires. Pour le Random Forest, `max_depth`= 10 limite trop la capacité du modèle. Cela illustre deux limites du tuning par validation croisée sur un dataset à classes très déséquilibrées, même lorsque le scoring est bien le *macro*-F1 (ce qui est notre cas). D'une part, le *macro*-F1 en CV est lui-même bruité à cause du faible nombre d'exemples rares par fold : un réglage qui gagne marginalement sur quelques classes rares peut être sélectionné au détriment d'un réglage plus stable. D'autre part, les régularisations retenues ($C = 0{,}1$ pour le SVM, `max_depth`= 10 pour le RF) améliorent peut-être le score CV moyen par un léger lissage des classes dominantes, mais coûtent trop sur les classes déjà limitantes (BNE, MY, MMY) dont la disparition totale dégrade fortement le *macro*-F1 val. Ensuite, le MLP est le seul modèle à capturer un minimum de signal sur les classes intermédiaires (EO, VLY, MY, MMY), probablement parce que ses couches non-linéaires peuvent combiner plusieurs features pour former des frontières plus complexes.

Table 3 – F1 par classe pour le meilleur modèle (MLP tuned).

Classe	Precision	Recall	F_1	Support
SNE	0,85	0,96	0,90	2603
LY	0,89	0,85	0,87	1621
BA	0,80	0,70	0,74	83
BL	0,72	0,75	0,73	403
MO	0,70	0,71	0,71	549
EO	0,68	0,45	0,54	172
VLY	0,57	0,27	0,37	73
PC	0,43	0,21	0,29	14
MY	0,29	0,21	0,24	88
MMY	0,33	0,10	0,15	72
PMY	0,08	0,04	0,06	23
BNE	0,13	0,01	0,02	78
PLY	0,00	0,00	0,00	2
macro avg	**0,50**	**0,40**	**0,43**	5781
weighted avg	0,80	0,82	0,80	5781

Le détail par classe (Tab. 3) est révélateur. Trois

groupes se dégagent clairement :

— les classes bien prédites (SNE 0,90, LY 0,87, BA 0,74, BL 0,73, MO 0,71) qui sont soit très fréquentes, soit morphologiquement distinctives,

— les classes intermédiaires (EO 0,54, VLY 0,37, PC 0,29, MY 0,24) où les features handcrafted capturent une partie du signal mais restent limitées,

— les classes quasi-impossibles (MMY 0,15, PMY 0,06, BNE 0,02, PLY 0,00) pour lesquelles le modèle échoue presque totalement.

Le cas de BNE est particulièrement parlant : avec 391 exemples de train, la classe n'est pas rare au sens strict, mais son F1 de 0,02 montre que les features handcrafted ne capturent pas ce qui la distingue de SNE. La différence entre un polynucléaire segmenté et un polynucléaire en bande est une question de morphologie fine du noyau, que les descripteurs globaux (excentricité, solidité, périmètre) ne saisissent pas. Le constat est similaire pour la série des myélocytes (MY, PMY, MMY) : les stades de maturation se différencient par des détails internes de la cellule que les features handcrafted moyennent.

Le meilleur score classique (0,4318) reste donc bien inférieur à ce qu'on peut espérer d'une approche profonde. Les features manuelles sont suffisantes pour distinguer les classes visuellement tranchées mais échouent sur les classes fines, qui sont justement celles qui tirent le plus la moyenne du *macro*-F1 vers le bas. Cela justifie pleinement la bascule vers le transfer learning : une représentation apprise de bout en bout par un backbone pré-entraîné devrait capturer des features discriminantes que l'extraction manuelle manque.

5 Approche profonde

5.1 Pipeline général

Compte tenu de la contrainte ImageNet et de la taille modérée du dataset, le *fine-tuning* d'un backbone pré-entraîné est bien plus adapté qu'un entraînement *from scratch*. Le pipeline de base reste commun à toutes les expériences :

— images en 224×224, normalisation ImageNet,

— augmentations train : HFlip, VFlip, Rotation $\pm 15°$, ColorJitter modéré,

— optimiseur AdamW (lr $= 10^{-4}$), scheduler `ReduceLROnPlateau`,

— *early stopping* sur le *macro*-F1 de validation, `patience`= 4, max. 15 epochs,

— `CrossEntropyLoss` standard, tête linéaire précédée d'un *dropout*,

— batch size 32.

5.2 Baseline ResNet-18

La configuration de départ utilise ResNet-18 pré-entraîné [1]. Elle atteint un *macro*-F1 de validation de 0,6801. La matrice de confusion (Fig. 2) montre que, même avec un sur-échantillonnage pondéré, les classes rares restent difficilement apprises. Les classes majoritaires comme SNE et LY dominent les prédictions, tandis que les classes minoritaires (BNE, MMY, PMY, PC, PLY) présentent des rappels plus faibles. Le score global masque donc encore un déséquilibre structurel dans les performances du modèle.

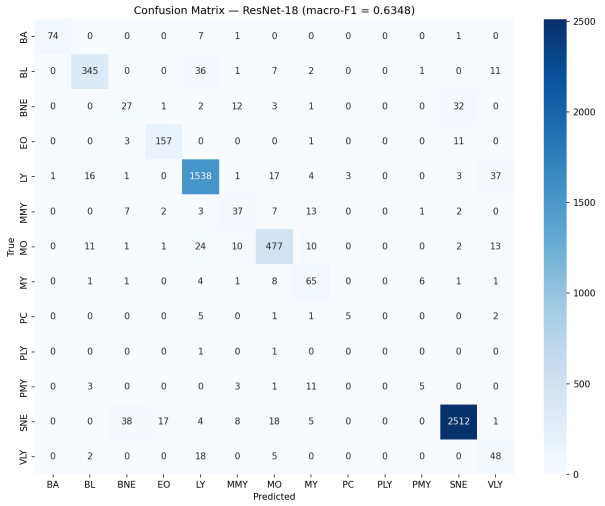


Figure 2 – Matrice de confusion de ResNet-18 avec sur-échantillonnage pondéré seul (macro-F1 = 0,6348). Les classes rares (BNE, MMY, PMY, PC, PLY) présentent des rappels plus faibles que les classes fréquentes.

6 Raffinements successifs

Chaque étape qui suit répond à un problème identifié à l'étape précédente. L'ordre n'est pas une chronologie exhaustive des essais mais la logique qui émerge avec le recul.

6.1 Traitement du déséquilibre

Deux leviers sont disponibles. La **loss pondérée** applique des poids $w_i \propto 1/n_i$ à la `CrossEntropyLoss` : elle augmente le coût des erreurs sur les classes rares, sans changer leur fréquence d'apparition. Le **sur-échantillonnage pondéré** (`WeightedRandomSampler`, `replacement=True`, $w_i = 1/n_i$) fait l'inverse : il ne touche pas au coût des erreurs mais force le modèle à voir les classes rares aussi souvent que les fréquentes. On peut combiner les deux, au risque de sur-corriger.

On isole l'effet de chaque levier par une ablation à backbone fixé, ResNet-18 + Dropout 0,3 + *early stopping* (Fig. 3, Tab. 4).

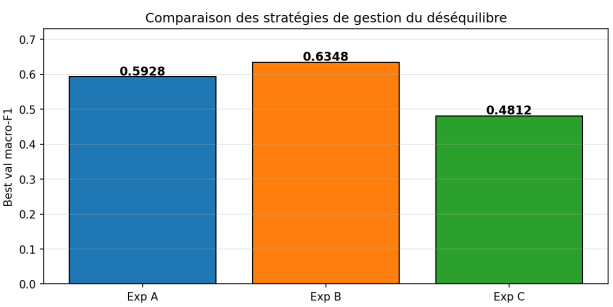


Figure 3 – Ablation des stratégies de rééquilibrage.

Table 4 – Résultats de l'ablation sur le déséquilibre.

Config.	Loss pond.	Sampler pond.	Val F_1
A	✓		0,5928
B		✓	**0,6348**
C	✓	✓	0,4812

Le sampler seul domine. La combinaison A+B est la pire des trois : en amplifiant à la fois le gradient et la fréquence sur les rares exemples disponibles des classes minoritaires, le modèle sur-apprend ces points (train loss $\sim 0{,}03$, val loss $\sim 1{,}5$ pour C). On retient donc le sampler seul avec une `CrossEntropyLoss` standard.

Un point reste anormal : la meilleure configuration B donne 0,6348, soit *moins* que la baseline brute sans aucun traitement du déséquilibre (0,6801). Un autre élément du pipeline est donc devenu nuisible entretemps. Plutôt que d'ajouter de nouvelles optimisations par-dessus, on prend le temps d'identifier le coupable par ablation à facteur unique.

6.2 Crop automatique de la cellule

L'intuition était qu'un *crop* automatique recentrant l'image sur la cellule devrait aider : moins de fond, meilleure exploitation des 224×224 pixels. On avait donc implémenté un *crop* par saturation HSV, seuillage d'Otsu et sélection de la plus grande composante connexe, avec un *fallback* sur l'image entière en cas d'échec. L'ablation à facteur unique (Fig. 4) est sans appel : avec *crop*, on obtient 0,6360, sans *crop* on remonte à 0,6804.

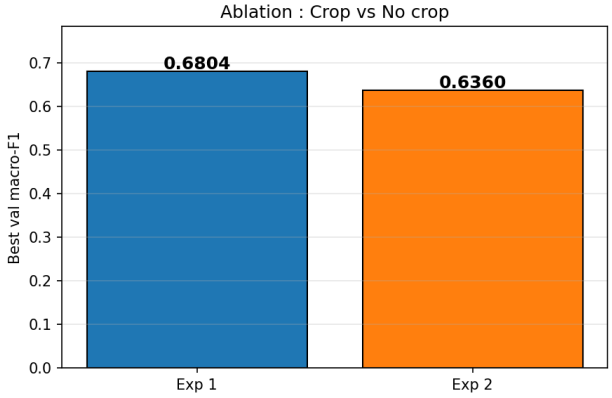


Figure 4 – Ablation *crop* HSV vs image brute.

Trois raisons peuvent expliquer cela. Un backbone pré-entraîné ImageNet utilise le contexte autour des objets, et un *crop* trop serré fait perdre des indices de coloration et de densité locale. Les rotations s'appliquent aussi moins bien sur des images très centrées, car les bordures noires introduites prennent une place relative plus importante. Enfin, les cas de *fallback* créent une incohérence d'échelle entre images cropées et non cropées.

Avec le *crop* désactivé, on retrouve le niveau de la baseline brute, mais cette fois avec un bon traitement du déséquilibre. Le pipeline est maintenant propre. Il reste néanmoins un écart train/val F1 important, de l'ordre de 0,29 : on s'attaque donc à l'overfitting.

6.3 Régularisation classique

Trois leviers sont testés : *weight decay*, *dropout* dans la tête, *progressive unfreezing*. Pour éviter les 12 combinaisons possibles, on procède en séquence : meilleur *weight decay* d'abord, puis *dropout* sur cette base, puis *freeze*.

Table 5 – Détail de l'ablation séquentielle.

Étape	Facteur	Val F_1	Gap tr/val
A	wd $= 10^{-4}$ (réf.)	0,6564	0,304
A	wd $= 5 \cdot 10^{-4}$	**0,6880**	0,292
A	wd $= 10^{-3}$	0,6652	0,277
B	dropout $= 0{,}3$	0,6616	0,308
B	dropout $= 0{,}5$	0,6325	0,270
C	no freeze	0,6836	0,267
C	3 ep. gelés	0,5556	0,275

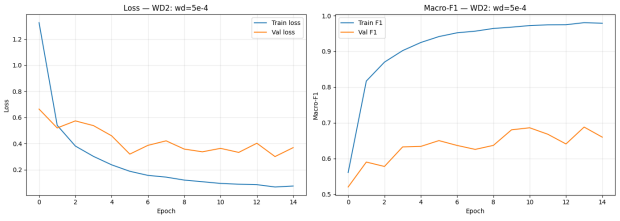


Figure 5 – Courbes d'entraînement pour la meilleure configuration (wd $= 5{\cdot}10^{-4}$). Le train F1 dépasse rapidement 0,97 tandis que le val F1 plafonne autour de 0,68.

Deux choses ressortent (Tab. 5, Fig. 5). Seul le *weight decay* apporte un gain mesurable : $5{\cdot}10^{-4}$ donne +0,008 sur la validation. Un *dropout* plus fort resserre le gap mais dégrade la validation, ce qui n'est pas l'objectif. Le *progressive unfreezing* consomme du budget d'epochs sans bénéfice net.

Le gain total reste modeste (+0,008). Surtout, même au meilleur réglage, le gap train/val reste à $\sim 0{,}29$. Si c'était purement de la mémorisation, une régularisation plus forte devrait continuer à le réduire. Or au-delà de $5{\cdot}10^{-4}$, on pénalise la validation sans gain. Cela indique que ce qui limite le modèle n'est plus la généralisation depuis le train mais la capacité intrinsèque du backbone à discriminer des classes morphologiquement proches (SNE vs BNE, MY vs PMY, LY vs VLY). La suite consiste donc à attaquer la qualité des features elles-mêmes.

6.4 Qualité de la représentation

Attention par canal (SE). Le premier essai insère un module *Squeeze-and-Excitation* [2] après chaque bloc résiduel de ResNet-18. Le module apprend un poids d'attention par canal via un *squeeze* (*global average pooling*), une *excitation* à deux FC avec bottleneck (*reduction*$= 16$), et une multiplication canal par canal. Le surcoût est de $\sim 0{,}4\,\%$ des paramètres.

Sur la validation locale, le gain est faible : 0,6498 contre 0,6449. Sur Kaggle en revanche, le score passe de 0,6528 à **0,6735**, soit +0,021. Le décalage entre gain local et gain Kaggle est le premier signal clair que la variance d'évaluation sur split unique est importante, et que de vrais gains peuvent être masqués par le bruit local. À ce stade, on n'en tire pas encore de conséquence méthodologique : l'écart pourrait être spécifique à SE. C'est le résultat de la tête *cosinus* (plus bas) qui rendra le problème impossible à ignorer.

Passage à ConvNeXt-Tiny. Le succès du SE montre qu'améliorer la qualité des features est un vrai levier. Plutôt que d'empiler des modules d'attention, on teste un backbone structurellement plus expressif : ConvNeXt-Tiny [3]. Cette architecture combine les principes des ConvNets (localité, invariance par translation) avec des conventions issues des Transformers (*patchify stem*, LayerNorm, GELU, grands noyaux). Avec $\sim$ 27,8M paramètres et des embeddings 768-d, sa capacité est nettement supérieure aux 11,2M / 512-d de (SE-)ResNet-18.

À pipeline identique, ConvNeXt-Tiny domine SE-ResNet-18 de +0,03 à +0,05 en *macro*-F1 selon la tête utilisée (Fig. 6). On le retient comme backbone.

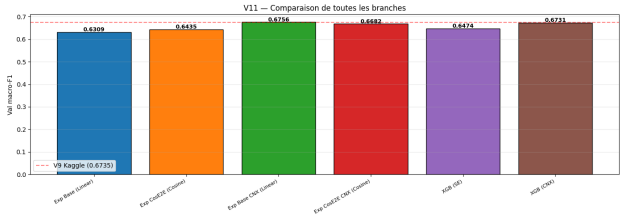


Figure 6 – Comparaison de six combinaisons backbone $\times$ tête. ConvNeXt-Tiny domine dans les trois classifieurs.

6.5 Choix de la tête de classification

Tête *cosinus* : un avertissement clair sur la validation. Une tête *cosinus* [4] normalise features et poids avant le produit scalaire, rendant la classification invariante à la norme des features. Dans une variante découplée (backbone entraîné séparément puis tête réentraînée), elle donne le meilleur *macro*-F1 de validation jamais observé localement : 0,6959. Mais la soumission correspondante régresse nettement sur Kaggle à 0,6290, soit $-0{,}045$ par rapport au meilleur score antérieur.

Deux problèmes expliquent cet écart. D'abord, le pipeline de soumission réintroduisait une tête linéaire et une `CrossEntropyLoss` standard pour un *fine-tuning* final sur *full train*, ce qui détruisait la géométrie angulaire apprise. Ensuite, le *best checkpoint* validation correspondait à un pic ponctuel, la métrique oscillant entre $\sim$ 0,65 et $\sim$ 0,70 d'une epoch à l'autre : le *best* sélectionné était en partie du bruit. Combiné à l'avertissement déjà observé sur SE, ce résultat rend intenable le fait de continuer à comparer des configurations sur un split unique.

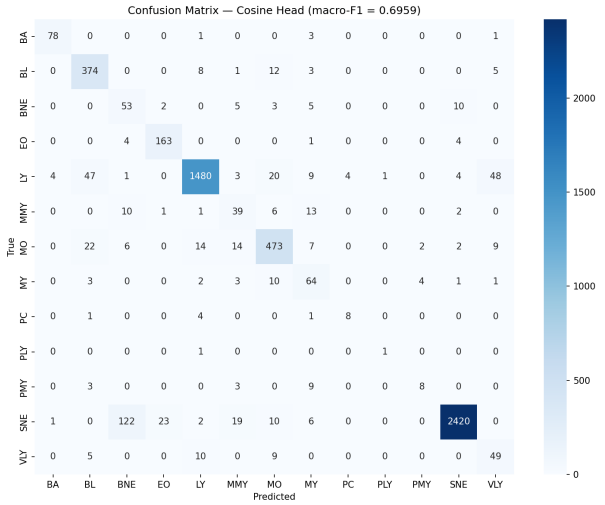


Figure 7 – Matrice de confusion avec la tête *cosinus* sur la validation locale. Les classes rares sont mieux prédites qu'avec la tête linéaire, mais le pipeline de soumission final dégrade cette géométrie.

XGBoost sur embeddings. L'échec de *cosinus* pose une question directe : existe-t-il une tête insensible à l'instabilité du *fine-tuning* final ? XGBoost [8] entraîné directement sur les embeddings 768-d du ConvNeXt-Tiny gelé est un candidat naturel : il n'est pas mis à jour conjointement avec le backbone, donc il est par construction à l'abri du problème rencontré par *cosinus*. Son résultat est de 0,6731, pratiquement identique à la tête linéaire profonde (0,6756). Ce constat, fait sur split unique, a deux implications. D'abord, il indique que les embeddings de ConvNeXt-Tiny sont déjà discriminants : le classifieur final n'est pas le goulet d'étranglement, le backbone fait l'essentiel du travail. Ensuite, il ouvre la voie à un classifieur externe qui, par construction, ne dépend pas de la stabilité du *fine-tuning* final (contrairement à *cosinus*) et reste rapide à réentraîner. La question de sa performance relative au backbone seul sera réexaminée en validation croisée section 8 : on y verra qu'en 5-fold, le backbone seul reste plus stable et marginalement plus performant que XGBoost sur ses embeddings.

6.6 Validation par 5-fold stratifié

Un split 80/20 unique ne suffit plus à départager des configurations dont les différences attendues sont du même ordre que la variance d'échantillonnage des classes rares. On passe à un `StratifiedKFold(5)` avec seed 42, et on rapporte la moyenne et l'écart-type du *macro*-F1 sur les folds. La prédiction test devient un ensemble : on moyenne les probabilités XGBoost sur les cinq folds avant d'appliquer l'`argmax`.

Chaque fold est un job autonome (fine-tuning du backbone, extraction des embeddings, entraînement XGBoost, probabilités test). On parallélise sur deux GPU via `subprocess` avec `CUDA_VISIBLE_DEVICES`, un fold par processus, avec une file d'attente simple. Le temps total passe de $\sim$ 230 min (mesuré en séquentiel sur un GPU) à 132 min, soit un *speedup* de 1,74$\times$.

7 Pipeline final

Table 6 – Hyperparamètres du pipeline final.

Composant	Configuration
Backbone	ConvNeXt-Tiny, ImageNet, 27,8M param.
Embeddings	768-d après `LayerNorm`
Tête (FT)	Dropout(0,3) + Linear(768, 13)
Optimiseur	AdamW, lr $= 10^{-4}$, wd $= 5 \cdot 10^{-4}$
Loss	`CrossEntropyLoss` standard
Sampler	`WeightedRandomSampler`, $w_i = 1/n_i$
Scheduler	`ReduceLROnPlateau`
Early stopping	Val F_1, `patience`= 4, max. 15 ep.
Augm. train	HFlip, VFlip, Rot($\pm 15°$), Jitter(0,1)
Image size	224×224
Batch size	32
Classif. final	XGBoost, max_depth= 6, lr= 0,1, subsample = 0,8, colsample= 0,8, reg_α = 0,1, reg_λ = 1,0, ES= 30
Validation	`StratifiedKFold`(5), seed 42
Inférence	Moy. probas XGBoost sur 5 folds, argmax
Parallél.	1 fold = 1 process = 1 GPU, 2 GPU

Le pipeline final (Tab. 6) combine les conclusions précédentes. Le **découplage représentation / classifieur** est rendu possible par le fait que XGBoost sur embeddings gelés atteint en split unique un score comparable à la tête linéaire profonde : la qualité des embeddings est suffisante pour qu'un classifieur externe les exploite. On retient XGBoost pour deux raisons pratiques : il est insensible aux aléas du *fine-tuning* final qui ont plombé *cosinus*, et il est rapide à réentraîner par fold. Le **5-fold stratifié** vient des deux avertissements successifs (SE, *cosinus*) qui ont montré qu'un split unique donne une image trompeuse de la vraie performance. L'**ensemble des folds** est la conséquence naturelle : une fois qu'on a cinq modèles indépendants entraînés sur des splits différents, moyenner leurs probabilités réduit la variance de sortie sans coût supplémentaire.

8 Résultats

8.1 Performance par fold

Table 7 – Résultats du 5-fold stratifié.

Fold	GPU	Backbone F_1	XGB F_1	trees	t (min)
1	0	0,7052	0,6918	44	67,6
2	1	0,7284	0,6619	43	54,7
3	1	0,7225	0,6685	50	36,2
4	0	0,7289	**0,7566**	49	40,2
5	1	0,7435	0,6886	49	39,9
Moy.		**0,7257 ± 0,014**	**0,6935 ± 0,038**		

Les résultats par fold sont en Tab. 7. Deux constats. Premièrement, le backbone seul est à la fois plus performant en moyenne *et* plus stable ($\sigma = 0{,}014$) que XGBoost sur ses propres embeddings ($\sigma = 0{,}038$). XGBoost ne peut s'appuyer que sur les quelques exemples qui tombent en train pour une classe rare donnée, alors que la tête neuronale bénéficie du sur-échantillonnage lors de l'entraînement. Deuxièmement, l'écart inter-fold sur XGB est important (0,6619 à 0,7566) : une classe comme PC (~ 14 exemples en val par fold) peut voir son rappel bouger de plusieurs dizaines de points selon le tirage.

8.2 Analyse par classe

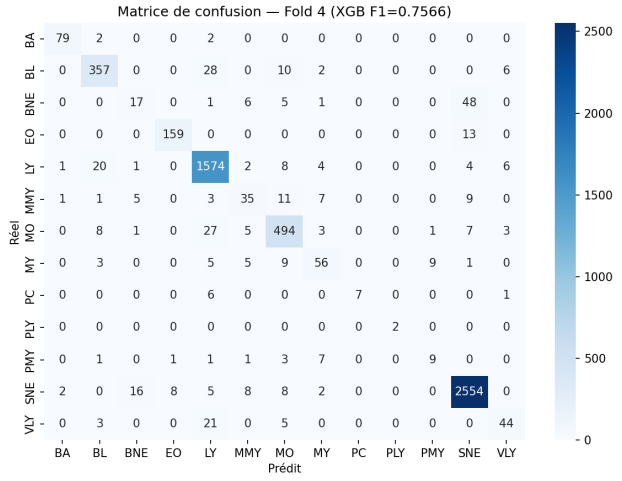


Figure 8 – Matrice de confusion sur la validation du meilleur fold (Fold 4, XGBoost $F_1 = 0{,}7566$).

Table 8 – F1 par classe sur le Fold 4 du pipeline final (ConvNeXt-Tiny + XGBoost).

Classe	Precision	Recall	F_1	Support
SNE	0,97	0,98	0,97	2603
LY	0,94	0,97	0,96	1620
MO	0,89	0,90	0,90	549
BL	0,90	0,89	0,89	403
EO	0,95	0,92	0,94	172
BA	0,95	0,95	0,95	83
MY	0,68	0,64	0,66	88
BNE	0,42	0,22	0,29	78
VLY	0,73	0,60	0,66	73
MMY	0,56	0,49	0,52	72
PMY	0,47	0,39	0,43	23
PC	1,00	0,50	0,67	14
PLY	1,00	1,00	1,00	2
macro avg	**0,81**	**0,73**	**0,76**	5 780
weighted avg	0,93	0,93	0,93	5 780

Sur le meilleur fold (Fig. 8, Tab. 8), le comportement est typique d'un modèle *long-tail* avec une hiérarchie claire entre trois groupes. Les classes majoritaires (SNE 0,97, LY 0,96, MO 0,90, BL 0,89) atteignent des F1 très élevés, comparables à ce qu'on obtient sur des datasets équilibrés. Les classes intermédiaires (EO, BA autour de 0,94) bénéficient aussi pleinement de la représentation apprise. Les classes rares présentent des performances hétérogènes, avec des F1 allant de 0,29 (BNE) à 0,67 (PC). Cette variabilité reflète à la fois le déséquilibre des données et la séparabilité intrinsèque des classes : c'est là que la marge de progrès se concentre. Les progrès les plus spectaculaires par rapport à l'approche classique concernent précisément ces classes intermédiaires et rares qui étaient presque inaccessibles aux features handcrafted : BNE passe de 0,02 à 0,29, MY de 0,24 à 0,66, MMY de 0,15 à 0,52, PMY de 0,06 à 0,43, VLY de 0,37 à 0,66.

Le cas de PLY mérite d'être commenté. Sur ce fold particulier, les 2 exemples tombés en validation sont correctement prédits ($F_1 = 1{,}00$), mais ce résultat est fragile : avec seulement 2 observations, une unique erreur ferait chuter le F1 à 0. Les autres folds montrent d'ailleurs une performance bien plus variable sur PLY,

et l'ensemble final (moyenne des probabilités sur les 5 folds) ne prédit jamais PLY sur le test. La classe reste donc structurellement non apprenable ; le $F_1 = 1{,}00$ de ce fold est un artefact de la très petite taille du support, pas un vrai succès.

Les confusions principales ne sont pas aléatoires : elles suivent la morphologie. La matrice de confusion (Fig. 8) met en évidence trois blocs de confusion récurrents : SNE ↔ BNE (polynucléaires segmentés et en bande), la série des myélocytes (MY, PMY, MMY) à différents stades de maturation, et LY ↔ VLY (lymphocytes normaux et variants). Les erreurs portent donc sur des cas réellement ambigus visuellement, pas sur des confusions triviales entre classes aux morphologies très différentes.

8.3 Distribution des prédictions test

L'ensemble des cinq folds produit 9 634 prédictions test. La distribution reflète celle du train : SNE 4 426, LY 2 788, MO 922, BL 676, EO 273, MY 142, BA 133, MMY 101, VLY 72, BNE 59, PMY 33, PC 9, PLY 0. Avec seulement 11 exemples de PLY dans le train au total, aucun fold ne reçoit assez de signal pour apprendre une frontière fiable, et la classe n'est jamais prédite.

8.4 Synthèse des principales soumissions

La Tab. 9 synthétise les principales configurations testées, de l'approche classique jusqu'au pipeline final. Les deux points marquants sont (i) la divergence local/Kaggle pour SE et *cosinus*, qui a motivé le 5-fold, et (ii) le fait que XGBoost sur embeddings atteint pratiquement le même score que la tête linéaire profonde, ce qui a motivé le découplage. Le gain de l'approche profonde sur la meilleure configuration classique est de +0,26 en *macro*-F1.

9 Discussion

Variance d'évaluation. L'écart-type inter-fold de 0,038 sur XGBoost implique qu'un gain de 0,04 observé sur un seul fold, a fortiori sur un split unique, n'est pas statistiquement significatif. Cette observation rétrospective éclaire deux épisodes du projet : le léger gain local du SE (masqué par le bruit mais visible sur Kaggle), et surtout le *best* local de *cosinus* qui s'est révélé illusoire sur Kaggle. Les deux confirment que sur un dataset aux classes très rares, évaluer sur un split unique peut récompenser des configurations adaptées au tirage plus qu'au problème lui-même.

Limites structurelles des classes rares. PLY, avec 11 exemples, n'est pas réellement apprenable en supervision standard. Le sur-échantillonnage ne crée pas d'information absente : avec `replacement=True`, on fait simplement revoir les mêmes deux ou trois exemples plusieurs fois par epoch, ce qui favorise le sur-apprentissage sur ces points. Les augmentations limitent le risque mais ne génèrent pas de variabilité intrinsèque nouvelle. Le même raisonnement s'applique à PC, PMY et BNE. Ces classes bénéficieraient principalement d'un effort d'annotation supplémentaire plutôt que d'une amélioration algorithmique.

Pourquoi s'arrêter à cette configuration. Deux raisons complémentaires. D'un côté, la variance inter-fold de 0,038 rend difficile la validation de nouveaux gains sans un protocole plus coûteux (plus de folds, plusieurs seeds par fold). De l'autre, la stabilité du backbone ($\sigma = 0{,}014$) indique que la marge de progrès ne se situe plus côté architecture : les pistes restantes relèvent de l'annotation, de l'augmentation ou du tuning XGBoost dédié, c'est-à-dire de leviers qui sortent du cadre actuel.

Pistes d'amélioration. Plusieurs pistes restent cohérentes avec les observations. (i) Tuning des hyperparamètres XGBoost par Optuna ou équivalent : la configuration actuelle n'a pas été optimisée. (ii) Augmentations plus riches en amont du backbone (RandAugment, MixUp modéré), en écartant les opérations destructrices pour la morphologie cellulaire. (iii) Ensemble entre backbones (ConvNeXt-Tiny + SE-ResNet-18 + EfficientNet-B0) : les erreurs étant partiellement décorrélées, l'ensemble devrait lisser les confusions. (iv) Contrastive supervisé (SupCon) pour mieux structurer l'espace latent. (v) Test-Time Augmentation à l'inférence.

Métrique d'évaluation

La métrique officielle est le *macro*-F1, qui pondère chaque classe de manière égale. Ce choix est adapté dans un cadre d'aide au diagnostic, où une erreur sur une classe rare peut être aussi critique qu'une erreur sur une classe fréquente. L'accuracy aurait été trompeuse : un modèle prédisant uniquement SNE et LY atteindrait déjà $\sim 73\,\%$ d'accuracy tout en étant pratiquement inutilisable.

Limitations

Plusieurs limitations sont à reconnaître. D'abord, le dataset reste modeste pour une classification à 13 classes, surtout pour les classes rares dont certaines ne comptent que quelques dizaines d'exemples. Ensuite, la contrainte ImageNet empêche d'utiliser des backbones spécifiquement pré-entraînés sur des images médicales, qui pourraient offrir une meilleure représentation de départ. Enfin, l'évaluation reste sensible au tirage des folds sur les classes très rares : même avec un 5-fold stratifié, l'écart-type reste de l'ordre de 0,04, ce qui limite la détection de gains fins.

10 Conclusion

On a construit progressivement un pipeline de classification pour un problème *long-tail* à 13 classes, sous

Table 9 – Synthèse des principales configurations testées.

Méthode	Val F_1	Kaggle
MLP tuned (handcrafted features)	0,4318	–
ResNet-18 baseline	0,6801	–
ResNet-18 + régularisation (wd $= 5 \cdot 10^{-4}$)	0,6880	0,6528
SE-ResNet-18	0,6498	**0,6735**
Cosine Head (pipeline FT linéaire)	0,6959	0,6290
ConvNeXt-Tiny + Linear	0,6756	–
ConvNeXt-Tiny + XGBoost (single split)	0,6731	–
ConvNeXt-Tiny + XGBoost, 5-fold ensemble	$\mathbf{0{,}6935 \pm 0{,}038}$	0.6813

contrainte de pré-entraînement ImageNet. La progression est nette : de 0,4318 avec la meilleure approche classique (MLP sur features handcrafted) à $0{,}6935 \pm 0{,}0375$ en validation croisée avec le pipeline profond final, soit un gain de +0,26 en *macro*-F1. Le backbone seul atteint $0{,}7257 \pm 0{,}0138$, ce qui montre que la représentation apprise est largement supérieure aux features extraites manuellement.

Les expériences font ressortir plusieurs points concrets :

- le sur-échantillonnage pondéré suffit pour traiter le déséquilibre, la loss pondérée ajoute peu et leur combinaison sur-corrige ;
- un preprocessing intuitivement utile (*crop* HSV automatique) peut en réalité dégrader les performances, d'où l'importance d'ablations à facteur unique ;
- la régularisation classique plafonne rapidement : ce qui fait vraiment bouger la performance ensuite, c'est le choix du backbone ;
- les embeddings d'un backbone correctement entraîné sont suffisamment discriminants pour qu'un classifieur externe (XGBoost) atteigne un score comparable sur split unique, même si en validation croisée la tête neuronale entraînée *end-to-end* reste plus stable et légèrement plus performante ;
- sur un dataset aux classes très rares, la validation sur un seul split récompense trop facilement des configurations adaptées au tirage : le K-fold stratifié devient une nécessité, pas un raffinement.

La marge de progrès restante ne se situe plus du côté de l'architecture ($\sigma = 0{,}014$ sur le backbone) mais du côté du classifieur final, des augmentations et de l'annotation des classes rares.

Références

[1] K. He, X. Zhang, S. Ren, J. Sun. *Deep Residual Learning for Image Recognition*. CVPR, 2016.

[2] J. Hu, L. Shen, G. Sun. *Squeeze-and-Excitation Networks*. CVPR, 2018.

[3] Z. Liu *et al. A ConvNet for the 2020s*. CVPR, 2022.

[4] S. Gidaris, N. Komodakis. *Dynamic Few-Shot Visual Learning without Forgetting*. CVPR, 2018.

[5] J. Ren *et al. Balanced Meta-Softmax for Long-Tailed Visual Recognition*. NeurIPS, 2020.

[6] K. Cao *et al. Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss*. NeurIPS, 2019.

[7] B. Kang *et al. Decoupling Representation and Classifier for Long-Tailed Recognition*. ICLR, 2020.

[8] T. Chen, C. Guestrin. *XGBoost : A Scalable Tree Boosting System*. KDD, 2016.