## Plan baseline

1. **Baseline principale = classifieur CNN (MobileNetV2/ResNet18)**  
   → c’est la référence “downstream” la plus parlante. Elle exercera ton DataGenerator, expose vite le sur-apprentissage et sert d’ancre pour comparer **l’apport des GAN** plus tard.
2. **Baseline “classique” complémentaire = PyRadiomics + (LogReg / Linear SVM)**  
   → très peu de code, très robuste, et **interprétable**. Elle te donne un repère non-neuronal crédible et améliore ta note méthodo (features hand-crafted standardisées).
3. **SSM/LSSM = module “morpho/explicabilité” (option générateur historique)**  
   → excellent pour montrer la **variabilité de forme** (modes PCA), faire de la **détection d’anomalies** (erreur de reconstruction) et préparer un **baseline de génération orienté forme**. Mais ce n’est **pas** le meilleur point de départ si ton objectif immédiat est d’obtenir des **scores de classification** + un pipeline stable.

## Pourquoi ce choix

* **Impact/effort** : le **CNN** te donne rapidement une métrique **downstream** forte (accuracy/macro-F1) et “stress-teste” tes prétraitements (stain aug, artefacts).
* **Risque méthodo** : **PyRadiomics** limite le risque de “black box only”, apporte des **features explicables** (GLCM/GLRLM/GLSZM, etc.) et un résultat souvent surprenant d’efficacité.
* **Pertinence GAN** : **SSM/LSSM** produit des formes (peu de texture) → utile pour un **baseline génération** (tu peux mesurer LPIPS/KID/FID mais attends-toi à des scores modestes côté réalisme) et pour des **figures pédagogiques** (modes de variation). C’est parfait comme **encadré explicatif** et comme “générateur historique” à opposer à **StyleGAN/PixCell**.

## Plan concret (sans te surcharger)

* **Phase A – CNN baseline**
  + Modèle léger (MobileNetV2 ou ResNet18) + **augmentations** adaptées histo (flip/rot, jitter couleur, **stain augmentation** légère, blur/ JPEG).
  + Métriques : **macro-F1**, balanced accuracy, confusion matrix.
  + Tracke l’**overfit** (courbes train/val, early stopping).
* **Phase B – PyRadiomics baseline**
  + Extraction sur patchs (grayscale + éventuellement canaux H/E séparés).
  + Classifieur simple (LogReg multinomiale **ou** Linear SVM), class\_weight="balanced".
  + Même protocole d’éval (split stratifié / CV).
* **Phase C – SSM/LSSM (EDA + baseline génération forme)**
  + Construire SSM par classe à partir de **masques binaires** (proxy simples si pas d’annotations : régions tissulaires/“gland-like” via SimpleITK + seuillage).
  + Publier : **modes de variation** (±3σ), **erreur de reconstruction**, **classement par vraisemblance** (distance de Mahalanobis) en bonus.
  + Si tu veux une métrique de génération : échantillonner des formes SSM, rasteriser, **peindre une texture simple** (optionnel) → calculer **KID** (plus stable que FID en petit n), **LPIPS** (à prendre avec recul en histo).

## Petit tableau repères

| Approche | Ce que tu mesures le mieux | Forces | Faiblesses |
| --- | --- | --- | --- |
| **CNN (MobileNet/ResNet)** | **Downstream** (macro-F1, gen) | Référence actuelle, exerce ton DataGenerator | Surapprentissage si variabilité faible |
| **PyRadiomics + LR/SVM** | Downstream + **interprétabilité** | Simple, robuste, standard du domaine | Insensible à certains motifs complexes |
| **SSM/LSSM** | **Morphologie** + baseline **génération (forme)** | Très explicable, peu de données, détection anomalies | Pas d’apparence/texture → FID/LPIPS faibles |

Spec DataGenerator

* **Qualité d’image** : tu calcules les métriques (Laplacian variance, entropie, Tenengrad, ratios de blanc/saturation, blockiness spatial & DCT, fraction tissulaire).
* **Normalisation Vahadane** via torch\_staintools, fallback propre si indisponible.
* **Calibration par classe** du blockiness, avec z-scores.
* **Échantillonnage équilibré** (balance\_per\_class + samples\_per\_class\_per\_epoch).
* **Gestion fine du split** train / val / test.
* **Filtrage qualité dynamique** (5 essais avant abandon).
* **Visualisation intégrée** pour vérification rapide.
* **CPU/GPU agnostique** (fonctionne dans Docker GPU, et sans GPU sur AWS).
* **Lorsque le dataset est échantilloné de nouvelles images sont chargées à chaque epoch**, donc pas un simple shuffle – évite le sur-apprentissage.

Config des seuils de filtres par classe :  
🎯 Objectif : cohérence biomédicale et apprentissage robuste

Le pipeline :

* **écarte les artefacts réels** (flou, bruit JPEG, tuiles vides),
* **garde les classes BACK et ADI**, car elles sont **sémantiquement importantes** pour la détection complète sur lame (fond / tissu adipeux sain),
* tout en **conservant un filtrage strict pour les classes pathologiques et tissulaires** (TUM, STR, LYM, etc.).

| Classe | Avant | Après |
| --- | --- | --- |
| **BACK** | rejetée à 100 % | toutes les tuiles de fond passent ✅ |
| **ADI** | 5/50 OK | ~100 % OK, car les tissus adipeux sont naturellement peu texturés |
| Autres classes | inchangées | le filtrage reste strict (seulement les tuiles “propres”) |