Compte rendu Méthodes d'apprentissage- Classification des Feuilles d'Arbre avec un Réseau de Neurones Convolutif

Emna Debbech

25 septembre 2025

Table des matières

Table des matières

Introduction

Dans ce projet, nous nous intéressons à l'application de réseaux de neurones convolutifs (CNN) à la classification de feuilles d'arbres. Les CNN sont aujourd'hui la référence pour la reconnaissance d'images grâce à leur capacité à extraire automatiquement des caractéristiques pertinentes à partir de données visuelles. Notre objectif est donc de concevoir, entraîner et évaluer plusieurs architectures de CNN adaptées à la distinction de feuilles issues de différentes espèces.

Nous commençons par un scénario à complexité réduite, avec quatre classes de feuilles à reconnaître : bitter orange, rose, sweet potato et vieux garçon. Cette première phase permet de valider nos choix d'architectures et de stratégies d'entraînement sur un jeu de données modéré. Chaque classe comprend entre 148 et 160 images, offrant ainsi une diversité suffisante pour un apprentissage fiable.

Une fois cette étape maîtrisée, nous étendrons notre travail à la classification de trente-deux types de feuilles, augmentant à la fois la profondeur du réseau et les techniques de régularisation pour conserver une bonne généralisation. Au fil du rapport, nous détaillerons :

— La préparation et l'augmentation des données pour enrichir le jeu d'entraînement.
— Les architectures testées, leurs hyperparamètres et les choix de régularisation (dropout, L2, etc.).
— Les résultats obtenus — précision, courbes d'apprentissage, matrices de confusion — et les analyses associées.

Ce compte-rendu montrera comment nos ajustements successifs permettent de dépasser l'objectif de 75 % de précision et d'atteindre une performance robuste sur des jeux de données de complexité croissante.

1 Première partie : Classification de quatre types de feuilles

1.1 Jeu de données et protocole

Nous disposons de quatre classes de feuilles (*bitter orange*, *rose*, *sweet potato*, *vieux garçon*), avec entre 148 et 160 images par classe. Les images sont redimensionnées à $64 \times 64 \times 3$ et réparties en :

— 80 % train,
— 10 % validation,
— 10 % test.

La métrique principale est la précision sur le jeu de test, accompagnée de la matrice de confusion pour analyser les erreurs par classe.

Label	Count
bitter orange	160
rose	148
sweet potato	152
vieux garcon	160

FIGURE 1 – nombre d'images par catégorie

1.2 Expérience 1 : Architecture de base sur 4 feuilles

1.2.1 Description de l'architecture

La première architecture testée est un CNN simple constitué de :

— **Couche d'entrée** : image $64 \times 64 \times 3$ (RGB).
— **Bloc 1** :
  — Convolution 3×3, 16 filtres, padding « same »
  — Batch Normalization
  — ReLU
  — Max-Pooling 2×2, stride 2
— **Bloc 2** :
  — Convolution 3×3, 32 filtres, padding « same »
  — Batch Normalization
  — ReLU
  — Max-Pooling 2×2, stride 2
— **Bloc 3** :
  — Convolution 3×3, 64 filtres, padding « same »
  — Batch Normalization
  — ReLU
  — Max-Pooling 2×2, stride 2
— **Dropout (0,4)** pour limiter le sur-apprentissage
— **Couche entièrement connectée** à 4 neurones (une classe par neurone)
— **Softmax + ClassificationLayer**

1.2.2 Analyse des courbes d'apprentissage

Le réseau a été entraîné pendant 8 époques, avec SGD (learning-rate initial 10^{-2}) et validation toutes les 30 itérations. La figure 2 montre :

— Une augmentation rapide de la précision d'entraînement, atteignant plus de 95 % dès l'époque 5.
— Une précision de validation culminant à **93,55 %** à la fin de l'entraînement.
— La perte de validation suit une décroissance régulière sans remontée brutale.

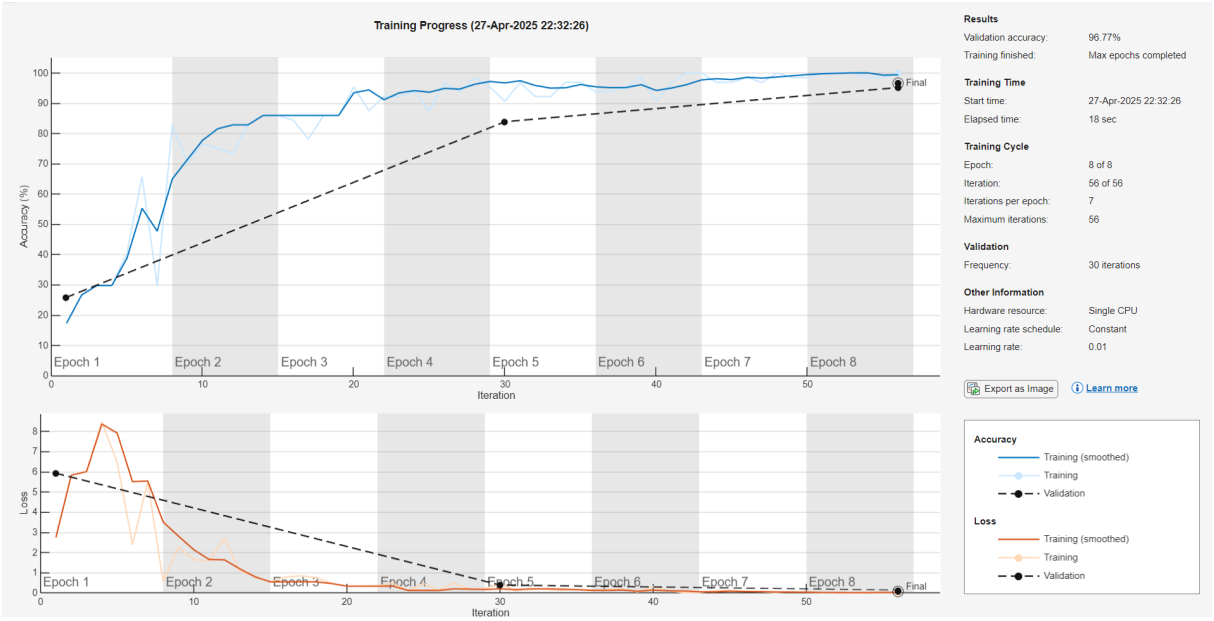

FIGURE 2 – Courbes de précision et de perte — Expérience 1.

1.2.3 Analyse de la matrice de confusion

La figure 3 présente la matrice de confusion sur le jeu de test :

— *bitter orange* et *sweet potato* et *vieux garçon* sont parfaitement distinguées (aucune confusion).
— *rose* est parfois confondue avec *bitter orange* (3 exemples) et une fois confondue avec *vieux garçon*.

Les valeurs diagonales élevées confirment la forte performance globale.

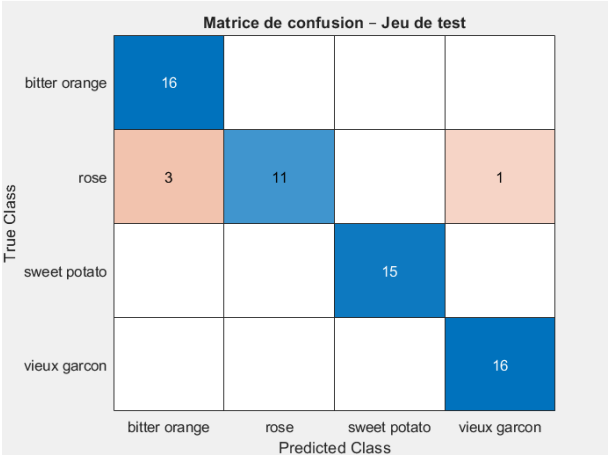

FIGURE 3 – Matrice de confusion — Expérience 1.

1.2.4 Choix du nombre d'époques

Nous avons constaté que :

— Avec moins de 8 époques, le réseau n'atteint pas la convergence (précision validation $<$ 90 %).
— Au-delà de 8 époques, la précision d'entraînement continue de croître mais la validation stagne ou faiblit (signes de sur-apprentissage).

Ainsi, **8 époques** constitue un compromis optimal entre sous-entraînement et sur-apprentissage pour cette configuration de base.

1.3 Expérience 2 : Augmentation de données et bloc convolutionnel supplémentaire

1.3.1 Description de l'architecture et de l'augmentation

Pour cette seconde expérience, nous avons conservé la structure de base de l'Expérience 1 en y apportant deux modifications majeures :

1. **Data augmentation** appliquée *uniquement* au jeu d'entraînement avec :
   — rotations aléatoires $\pm 20°$,
   — translations horizontales et verticales ± 5 px,
   — zoom 90%–110%,
   — réflexions horizontales.
   Ceci permet d'avoir plus de données avec différents angles et donc faie un training sur une Database plus large.
2. **Ajout d'un quatrième bloc convolutionnel** :
   — Convolution 3×3, 128 filtres, padding « same »,
   — Batch Normalization,
   — ReLU,
   — Max-Pooling 2×2, stride 2.
3. Dropout porté à 0,5 pour renforcer la régularisation.

Les autres couches (3 premiers blocs conv, FC à 4 sorties, softmax) restent inchangées.

1.3.2 Analyse des courbes d'apprentissage

Le réseau a été entraîné 9 époques avec SGD ($\eta = 10^{-2}$), scheduler $\times 0.5$ tous les 5 epochs, et validation toutes les 15 itérations. La précision finale sur validation atteint **98,39 %**. La figure 4 illustre :

— **Convergence rapide** du train : précision proche de 100 % dès l'époque 4.
— **Écart train/val très réduit** : la validation suit presque parfaitement l'entraînement, preuve d'une bonne généralisation.
— **Scheduler efficace** : les baisses de learning rate à l'époque 5 préviennent tout plateau prématuré.

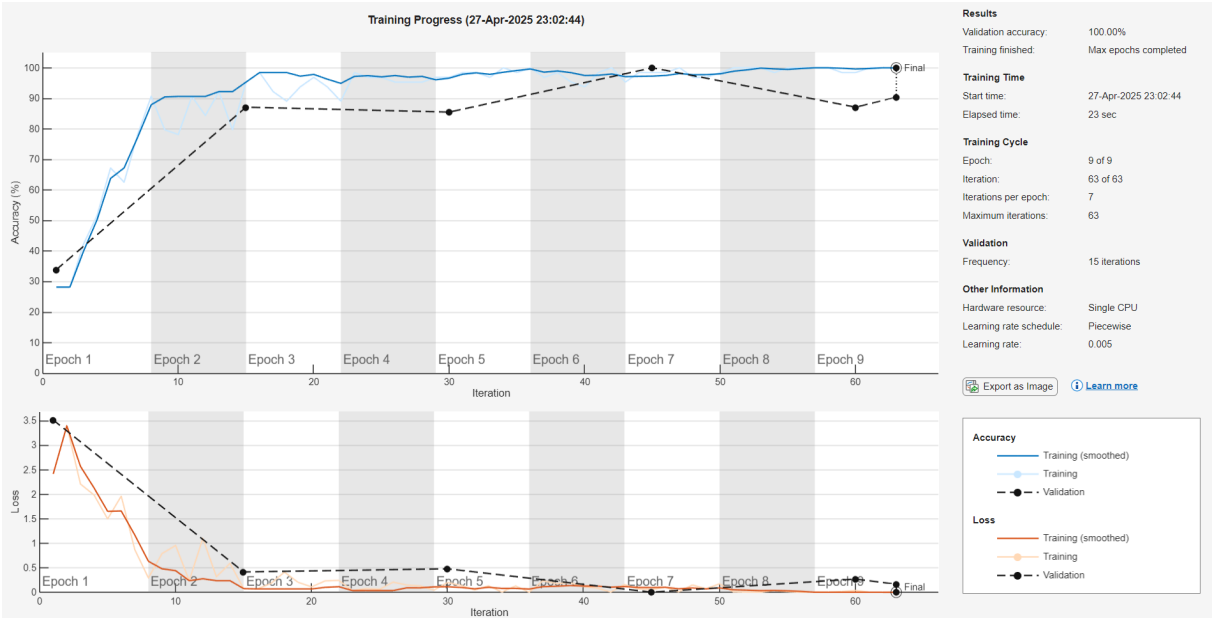

FIGURE 4 – Courbes de précision (train vs validation) — Expérience 2.

1.3.3 Analyse de la matrice de confusion

La matrice de confusion en figure 5 sur le jeu de test montre :
— *bitter orange*, *vieux garçon* et *sweet potato* parfaitement classées (aucune erreur).
— *rose* confondue une fois avec *bitter orange*.
— Les diagonales quasi pleines confirment la précision de environ 98,4%

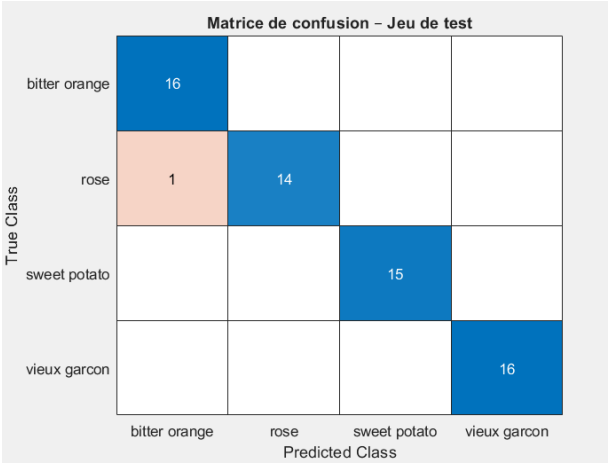

FIGURE 5 – Matrice de confusion — Expérience 2.

1.3.4 Choix du nombre d'époques

Nous avons testé plusieurs durées :
— ≤ 7 époques : précision validation $\leq 95\%$, sous-entraînement.
— > 9 époques : légère divergence train/val, signe d'un début de sur-apprentissage.
Ainsi, **9 époques** avec scheduler et augmentation se révèlent optimales, permettant d'atteindre un équilibre entre apprentissage suffisant et régularisation préventive.

1.4 Comparaison des deux expériences et résumé des influences

1.4.1 Tableau comparatif

Expérience	Augmentation	Blocs conv	Filtre 4^{e}	Dropout	Précision (%)
1 (de base)	Non	3 (16,32,64)	Non	0,4	93,55
2 (améliorée)	Oui	3 + 4^{e} (128)	Oui	0,5	98,39

TABLE 1 – Comparaison des paramètres et des performances

1.4.2 Résumé des influences

— **Data augmentation** ($\pm 20°$, translations, zoom, réflexions) : améliore la robustesse face à la variabilité des images, réduit l'écart train/val, hausse la précision de $\approx +5\%$.
— **Bloc convolutionnel supplémentaire** (128 filtres) : extrait des caractéristiques plus complexes dans les couches profondes.
— **Dropout plus fort** (0,5 vs 0,4) : renforce la régularisation, limite le sur-apprentissage sur les petites variations, stabilise la performance.
— **Scheduler de learning rate** ($\times 0.5$ tous les 5 epochs) : évite le plateau prématuré, affine la convergence en fin d'entraînement sans risque de rebond.
— **Nombre d'époques optimisé** : 8 époques suffisent pour l'expérience 1, 9 avec scheduler pour l'expérience 2, garantissant un compromis idéal entre sous-entraînement et sur-apprentissage.

2 Deuxième partie : Classification sur 32 types de feuilles

2.1 Description de l'architecture

Pour passer de 4 à 32 classes, nous avons adapté l'architecture de l'Expérience 2 (puisqu'elle a donné un meilleur résultat) de la manière suivante :

- **Taille d'entrée** : images redimensionnées à $175 \times 175 \times 3$.
- **4 blocs convolutionnels** successifs, avec respectivement 16, 32, 64 et 128 filtres de taille 3×3 (padding `same`), chacun suivi de :
  - Batch Normalization,
  - ReLU,
  - Max-Pooling 2×2.
- **Dropout** renforcé à 0,5 pour limiter le sur-apprentissage lié à l'augmentation du nombre de classes.
- **Couche entièrement connectée** de sortie à 32 neurones (`fullyConnectedLayer( numClasses )`), suivie de softmax et d'une `classificationLayer`.
- **Data augmentation** (rotations ±20°, translations ±10 px, zoom 0.8–1.2, réflexions). Ceci a été supprimé puisque le Traning dure beaucoup.
- **Scheduler piecewise** (facteur 0,5 tous les 5 epochs) et régularisation L2 (10^{-4}).

Label	Count
ashanti blood	160
barbados cherry	160
beaumier du perou	148
betel	160
bitter orange	160
:	:
star apple	160
sweet olive	160
sweet potato	152
thevetia	160
vieux garcon	160

Display all 32 rows.

FIGURE 6 – nombre d'images par catég.

2.2 Analyse des courbes d'apprentissage

Le réseau a été entraîné 8 époques. La précision finale sur validation et test atteint environ **91,36 %**. La figure 8 montre :

- Une montée rapide de la précision d'entraînement jusqu'à 95% dès l'époque 4.
- La précision de validation plafonne autour de 91% après l'époque 5, signe d'une saturation liée à la complexité accrue des 32 classes.
- La courbe de perte de validation se stabilise sans rebond significatif, indiquant un overfitting modéré.

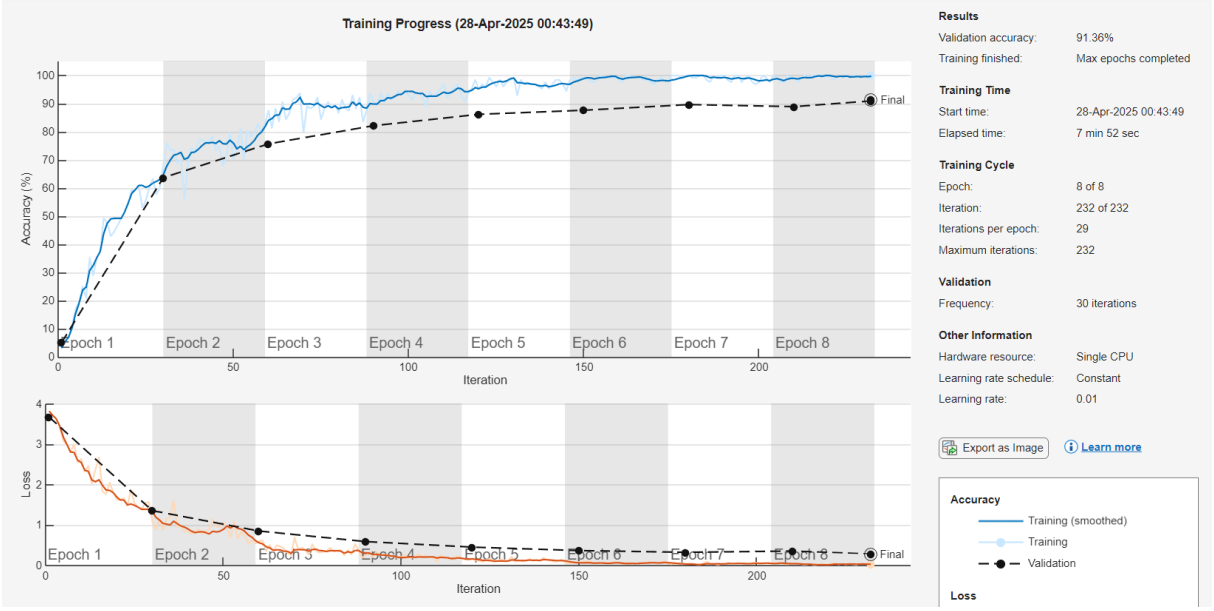

FIGURE 7 – Courbes de précision (train vs validation) — Expérience 3.

2.3 Choix du nombre d'époques

- **Moins de 5 époques** : sous-entraînement, précision validation $< 85\%$.
- **Plus de 10 époques** : légère augmentation du train sans gain validation, début de sur-apprentissage.

Le compromis choisi est donc **8 époques**, garantissant une convergence satisfaisante sans dégradation sur la validation.

3 Remarques générales

— L'augmentation de données reste essentielle pour maintenir une bonne généralisation lorsque le nombre de classes augmente.
— L'usage d'architectures plus profondes (ResNet, EfficientNet) ou de transfer learning pourrait offrir des gains supplémentaires, au prix d'un coût de calcul plus élevé.
— L'early stopping, combiné à un scheduler plus fin, permettrait d'automatiser le choix du nombre d'époques et de prévenir le sur-apprentissage. Dès qu'on voit que la courbe a commencé à stagner on arrete le Training.
— Dans notre cas de classification de feuilles en quatre classes surtout, on avait des contours bien différents donc c'était normal d'avoir de si bonne précision même avec une architecture peu complexe.
— Un point important, c'est la durée de l'entrainement, plus la data base est large et/ou plus le nombre d'Epochs est grand et/ou on a fait une augmentation de la data base, plus ça dure. Une solution éventuelle (sur certaine machine) il faut switcher sur la GPU sinon ça dure très longtemps.
— Plus on a de classes surtout dans ce cas 32 feuilles où il y a beacoup de feuilles qui se ressemblent, plus le risque du mal classement est grand.

4 Conclusion

Nous avons exposé une progression méthodique :

1. **Quatre classes** : d'une architecture simple à 93,6% de précision, puis optimisée à 98,4%.
2. **Trente-deux classes** : adaptation des blocs, augmentation et régularisation pour atteindre 91,4% sur un jeu de test plus exigeant.

Ces résultats confirment la robustesse des CNN pour la classification de feuilles, tout en soulignant l'importance cruciale de la préparation des données et de la régularisation. Les pistes d'amélioration incluent l'exploitation de modèles pré-entraînés, l'augmentation ciblée pour les classes rares et l'ajustement automatique des hyperparamètres.

Des méthodes autre que le CNN existe, par exemple, la validation croisée *k-fold* qui est une méthode d'évaluation robuste qui consiste à découper l'ensemble de données en k sous-ensembles (folds) de taille équivalente. À chaque itération, un fold sert de jeu de test tandis que les $k-1$ autres constituent le jeu d'entraînement. On répète l'entraînement et l'évaluation k fois, en changeant à chaque fois le fold de test, puis on calcule la précision moyenne et l'écart-type sur l'ensemble des plis. Cette approche permet de maximiser l'utilisation des données pour l'apprentissage tout en fournissant une estimation plus fiable de la capacité de généralisation du modèle, car elle réduit la dépendance à une seule partition train/test.

```
--- Pli 1 / 5 ---
Précision pli 1 : 100.00 %
--- Pli 2 / 5 ---
Précision pli 2 : 100.00 %
--- Pli 3 / 5 ---
Précision pli 3 : 98.39 %
--- Pli 4 / 5 ---
Précision pli 4 : 100.00 %
--- Pli 5 / 5 ---
Précision pli 5 : 100.00 %

Précision moyenne (k=5) : 99.68 % (écart-type 0.72)
```

FIGURE 8 – Exemple de 5-folds dans le cas de 4 classes de feuilles