Detection

Section	Description
1. Dataset	• Source du Dataset : Kaggle : https://www.kaggle.com/datasets/sujaradha/thermal-images-diseased-healthy-leaves-paddy/data • Procédure d'acquisition : input au niveau des serveurs Kaggle • Statistiques de Dataset : distribustion des classes de classification • Prétraitement : trier les images et séparer les sets de train test et validate
2. Méthodologie	• Techniques et algorithmes utilité : **ResNet-50 CNN, transfer learning**
3. Résultat	• Performance des modèles : loss: 0.3527 - acc: 0.8583 loss: 0.4013 - acc: 0.8740 • Visualisation des Résultats :

4. Discussion et interprétations :	• Analyse des résultats : augmentation de la précision après augmentation du nombre des epoche et déblocage des couche a été prévu car le modele a pu capter les nouveau paternes de classification des images en comparaison avec la premiere evaluation avec 15 epoches seulement et avec couche bloques • Interprétation :

Classification

Section	Description
1. Dataset	**1. Chargement des données** Images : Les images d'entraînement ont été chargées depuis un répertoire spécifique. Chaque image a été associée à une étiquette correspondant à une des quatre classes : healthy, multiple_diseases, rust, ou scab. Étiquettes : Les étiquettes ont été générées à partir des colonnes healthy, multiple_diseases, rust, et scab en prenant la colonne avec la valeur maximale pour chaque image (utilisation de idxmax). **2. Équilibrage des classes** Les données sont déséquilibrées, avec certaines classes comme multiple_diseases sous-représentées. Une tentative a été faite pour calculer les class weights avec compute_class_weight pour compenser ce déséquilibre lors de l'entraînement du modèle. **3. Division des données** Les données d'entraînement ont été divisées en train et validation, avec 20% des données réservées pour la validation. L'utilisation de ImageDataGenerator a permis de créer deux générateurs : train_datagen pour l'entraînement. val_datagen pour la validation. **4. Prétraitement des images** Les images ont subi plusieurs transformations pour standardiser leur format et améliorer l'apprentissage du modèle : Redimensionnement : Toutes les images ont été redimensionnées à une taille standard de 256x256 pixels. Normalisation : Les valeurs des pixels ont été normalisées en les divisant par 255.0 pour les ramener dans la plage [0, 1]. Augmentation de données (avec ImageDataGenerator) : Rotation, translation, et autres transformations appliquées automatiquement pour diversifier les données d'entraînement et éviter le surapprentissage. **5. Organisation des fichiers**

	Les images ont été triées dans des sous-dossiers correspondant à leurs classes (healthy, multiple_diseases, rust, scab). Cela a permis de structurer les données pour les générateurs de données (basés sur le flux des répertoires). **6. Gestion des classes déséquilibrées** Un poids de classe a été calculé et appliqué dans le modèle pour accorder plus d'importance aux classes sous-représentées. **7 Conversion et préparation pour Keras** Les images du dossier test ont été chargées, redimensionnées et préparées dans un format compatible avec TensorFlow/Keras : Normalisation. Extension de la dimension pour correspondre au format batch attendu par le modèle ((batch_size, 256, 256, 3)). **8. Vérification des données** Vous avez visualisé plusieurs images des différentes classes après redimensionnement pour valider leur qualité et leur prétraitement. Résumé des objectifs du prétraitement : Uniformiser les données pour qu'elles soient compatibles avec le modèle (même taille, même format). Réduire le déséquilibre des classes pour améliorer la performance. Diversifier les données grâce à l'augmentation pour éviter le surapprentissage.
2. Méthodologie	**3. Techniques et algorithmes utilité :** Un réseau de neurones convolutionnel (CNN) composé de plusieurs couches convolutionnelles et de pooling pour extraire les caractéristiques des images. Enfin, il y a une couche dense qui prend les caractéristiques extraites et produit une sortie avec 4 neurones, correspondant aux 4 classes de classification (healthy, multiple_diseases, rust, scab). La fonction d'activation softmax est utilisée à la sortie pour obtenir des probabilités pour chaque classe.
4. Résultat	• Performance des modèles :

	Precision	Recall	F1-score	support
helathy	0.42	0.61	0.5	148
Multiple_diseases	0.31	0.17	0.22	29
rust	0.53	0.66	0.59	185
scab	0.55	0.43	0.48	185

5. Discussion et interprétations :	Avec une accuracy de 0.4771 et les résultats du Classification Report, voici une analyse des performances du modèle : Points clés de l'analyse : Précision (Precision) : La précision pour la classe healthy est relativement faible (0.42), ce qui signifie que parmi toutes les prédictions pour la classe "healthy", seulement 42% sont correctes. Les autres classes, notamment multiple_diseases, rust, et scab, présentent aussi des précisions faibles, en particulier pour la classe multiple_diseases avec une précision de seulement 0.31. Rappel (Recall) : Le modèle semble mieux identifier la classe healthy avec un rappel de 0.61, ce qui signifie que 61% des images de cette classe sont bien classifiées comme "healthy". Toutefois, ce n'est pas optimal. Le rappel pour multiple_diseases est très faible (0.17), ce qui suggère que le modèle ne parvient pas bien à détecter cette classe, même si elle représente une petite proportion du dataset. Les classes rust et scab ont des rappels respectifs de 0.66 et 0.43, ce qui est un peu mieux que pour healthy, mais encore loin d'être idéal. F1-Score : Le F1-score est une moyenne harmonique entre la précision et le rappel. Il est faible pour toutes les classes, ce qui indique un compromis entre la capacité à détecter correctement une classe et à ne pas classer des images incorrectement dans cette classe. Les meilleures performances sont observées pour rust avec un F1-score de 0.59 et healthy avec 0.50, mais elles restent en dessous des niveaux souhaités. Support : Le support, c'est-à-dire le nombre d'exemples dans chaque classe, varie de 29 pour multiple_diseases à 185 pour rust et scab. Une petite taille de l'échantillon pour la classe multiple_diseases pourrait expliquer ses mauvaises performances. Le modèle peut avoir du mal à bien généraliser sur des classes moins représentées. Accuracy globale : L'accuracy globale du modèle est faible à 0.4771. Cela signifie que moins de la moitié des prédictions sont correctes. Le modèle a du mal à faire des prédictions correctes globalement, surtout pour les classes déséquilibrées.

	Conclusion : Le modèle a du mal à bien classer les différentes classes, particulièrement les classes moins fréquentes (comme multiple_diseases). Le faible F1-score et la faible précision pour multiple_diseases montrent qu'il y a un déséquilibre dans les données ou un manque d'exemples pour certaines classes. L'amélioration de l'accuracy pourrait nécessiter : Un meilleur traitement des données, comme l'augmentation de données pour équilibrer les classes. L'ajout de techniques avancées comme des class weights ou des modèles préentraînés pour mieux capturer les caractéristiques complexes des classes moins représentées. Un fine-tuning des hyperparamètres du modèle pour améliorer la performance globale.

Section	Description
1. Dataset	• Source **du Dataset:** Le dataset utilisé provient de Kaggle : `/kaggle/input/leaf-disease-segmentation-dataset` • Procédure **d'acquisition:** Les images et les masques sont chargés depuis les répertoires respectifs, redimensionnés à une taille cible de 128x128 pixels, et normalisés dans l'intervalle [0, 1]. • Statistiques **de Dataset:** Le dataset contient des images RGB des feuilles affectées par des maladies ainsi que leurs masques correspondants. • Prétraitement **:** - Redimensionnement des images et masques à 128x128. - Normalisation des valeurs de pixels. - Génération de lots pour l'entraînement avec un batch size de 16.
2. Méthodologie	**Techniques et algorithmes utilisés :** - Modèle basé sur l'architecture U-Net. - Utilisation de convolutions 2D, d'up-sampling, et de connexions de type skip. - Fonction de perte : binary_crossentropy. - Optimiseur : Adam. - Métriques: précision (accuracy).

3. Résultat	• Performance **des modèles :**
	
	- Après 10 époques d'entraînement:
	  - **Précision d'entraînement:** ~84.61%.
	  - **Précision de validation:** ~84.66%.
	  - **Perte de validation finale:** ~0.3313.
	
	• Visualisation **des résultats :**
	
	- Visualisation des courbes d'entraînement:
	  - Courbes de précision et de perte.
	- Matrice de confusion:
	  - Affichée pour évaluer les prédictions binaires.
	- Masques prédits:
	  - Comparaison entre les masques prédis et les masques de vérité terrain pour des exemples spécifiques.

4. Discussion et interprétations :	• Analyse **des résultats :** • Le modèle U-Net est efficace pour la segmentation binaire. • Les performances sont cohérentes avec une précision stable entre l'entraînement et la validation. • Interprétation **:** • La qualité des résultats dépend de la taille et de la diversité des données utilisées. • Des améliorations pourraient inclure : o Utilisation de données augmentées. o Entraînement sur un nombre d'époques plus élevé. o Exploration d'autres architectures comme DeepLabV3+.