Rapport de Modélisation

Nous ne traiterons pas ici du modèle utilisé pour le nettoyage des images du dataset pour éliminer les “enveloppes”, il sera présent dans le rapport final. A noter qu’il a permis d’écarter 2% des images de notre dataset.

Modèles testés

Pour mener à bien notre projet nous nous sommes basés sur des modèles pré-entraînés. Chacun des membres de l’équipe a utilisé un modèle différent afin de pouvoir comparer les performances respectives. Les 3 modèles utilisés sont les suivants :

- VGG16
- ResNet50
- EfficientNetB0

Nous avons fait en sorte de garder un code le plus proche possible pour les 3 modèles afin de faciliter la comparaison des résultats.

Pré-processing

Le générateur d’images par défaut ImageDataGenerator a été utilisé pour générer les images pour chaque Batch. Le générateur peut prendre en charge plusieurs opérations d’augmentation des images comme les rotations, les translations, les symétries horizontales ou verticales, les zooms ou encore la luminosité.
Le générateur d’image étant gourmand en temps de calculs, nous avons limité le nombre d’opérations :

- preprocess_input() nécessaire pour effectuer les opérations
- Et pour les augmentations :
    - une rotation des images avec un paramètre de 180°
    - un zoom maximum de 20%
    - une variation de la luminosité de 10%

La rotation permet d’éviter la prédominance de photos avec des champignons représentés pied en bas et chapeau en haut.
Nous avons essayé de remplacer le générateur standard par d’autres solutions. Nous avons dû mettre cette partie en standby faute de temps.
Il semble aussi que le preprocessing de nos modèles inséré dans ImageDataGenerator, fasse une normalisation des formats de photos. Si nous voulons réussir à nous séparer de ImageDataGenerator afin de gagner en temps d'entraînement, il nous faudra réussir à faire nous même cette conversion. Pour le moment, nous essayons deux méthodes. La première est de faire rentrer le pré-processing, de nos modèles pré-entraînés, dans la création de nos datasets. La seconde est de faire une vérification de nos photos en amont de la création de nos datasets. Chacune de ces méthodes nous a amené à des erreurs différentes. Nous espérons pouvoir en parler avec toi ce Lundi.

Entraînement et résultats

Ajout de Callbacks

Nous avons ajouté 3 fonctions qui sont évaluées à la fin de chaque Epochs :

- **EarlyStopping** permet d'évaluer si le modèle continue de s'améliorer ou si l'entraînement peut être arrêté. Nous l'avons paramétré avec la "val_acc" comme valeur à évaluer et avons choisi une patience de 5 Epochs avant de décider de stopper l'entraînement. Cette valeur a été choisie en raison du nombre d'Epochs généralement utilisés (Entre 20 et 100)
- **ModelCheckpoint** permet de sauvegarder le modèle en cours d'apprentissage à chaque fois qu'il améliore sa performance. Il évalue également la "val_acc".
- **ReduceLROnPlateau** permet d'ajuster le paramètre de Learning rate de l'optimizer Adam que nous avons utilisé. Le facteur multiplicatif (factor) a été fixé à 0,5. Nous avons remarqué qu'à 0.1, le learning rate changeait trop radicalement l'entraînement et l'amenait souvent sur un "faux plateau". Enfin, le paramètre de Learning Rate initial a été fixé à 0.001, amenant rapidement le modèle à 65% d'accuracy en 10 epochs. Voir courbe dans les résultats.

Impact du nombre de classes à classer

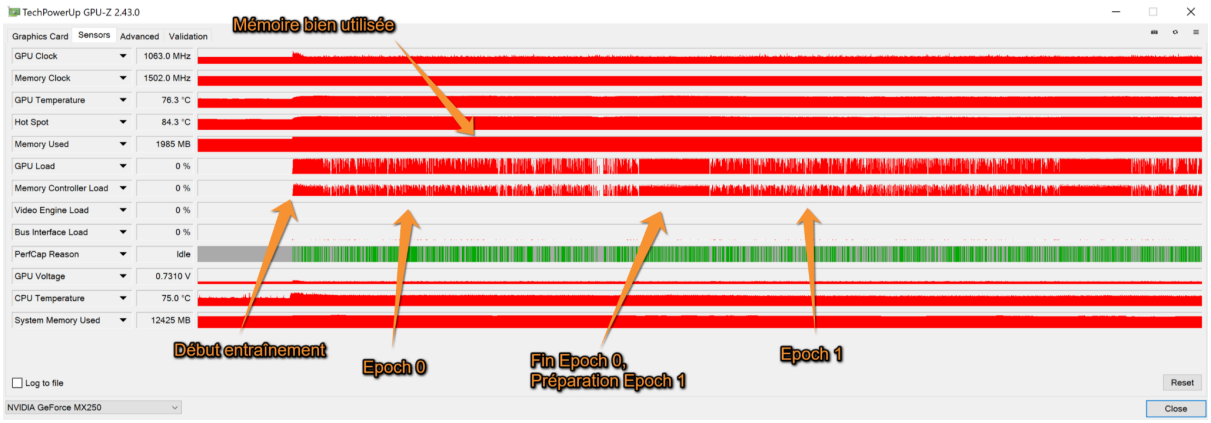
Nous avons progressivement augmenté le nombre de classes à entraîner au fur et à mesure des contenus téléchargés. Nous avons pu constater que la précision du modèle diminue avec le nombre de classes entraînées :

- Plus de 80% de précision avec 10 classes représentant des "Genus/Genres"
- Autour de 60-70% sur 20 classes représentant des "Genus/Genres"
- L'entraînement sur 64 classes est en cours

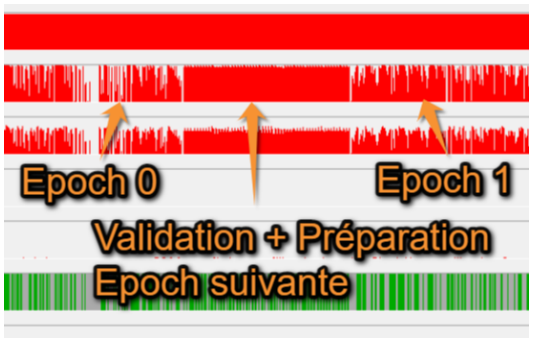
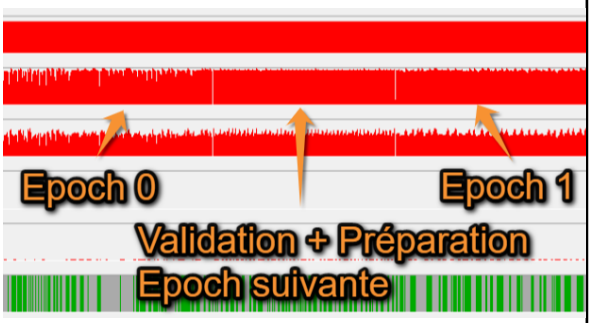
Optimisation de la charge du GPU

Nous avons pu constater que plusieurs paramètres peuvent influencer l'utilisation optimale du GPU. Les performances vont différer en fonction des points suivants :

- HDD lent vs SSD rapide : Les performances du disque sont particulièrement importantes pour un tel projet en raison du grand nombre de petits fichiers que nous utilisons. Un disque lent serait le goulet d'étranglement. On notera aussi de vérifier si son antivirus ne ralenti pas les accès au disque.
- GPU : The faster the better.
- Le générateur d'images comme vu précédemment pour ne pas que le GPU soit en attente.
- Batch Size : Ce paramètre doit être testé sur chaque config pour optimiser la charge du GPU. Exemple visuel ci-dessous avec un batch de 20 puis un batch de 10.



Avec un batch de taille 20, on remarque que le GPU oscille très rapidement entre 0 et 100%, il est sous-utilisé. Avec un batch de taille 10 le graphe est plus dense mais on remarque qu'il utilise moins de mémoire du GPU.

Batch size = 20 Memory Used 99%	Batch size = 10 Memory Used 88%
	

La taille des batchs devra donc être choisie pour s'adapter à chaque configuration. Nous conseillons de faire 1 ou 2 Epochs avec plusieurs valeurs pour définir une taille de batch adaptée.

Genus vs Species

Nous avons d'abord travaillé sur la reconnaissance de "Genus/Genres", c'est-à-dire 1 niveau au-dessus des "Species/Espèces", les espèces représentant les champignons que l'on ramasse dans la nature. Une fois les téléchargements plus avancés, nous avons pu faire un entraînement au niveau le plus bas de notre hiérarchie (les species). Cet essai avait pour but de voir si une grande différence de résultat était visible ou non entre reconnaître un "genus" ou une "species". Cette idée nous est apparue lors du saut de nombre de classes, à reconnaître, de 10 à 20. Lors du modèle avec 10 "genus", nous avions choisi individuellement chaque classe sur des critères visuels (Des spécificités très prononcées à l'œil). Alors que lors des 20 "genus", ces classes étaient prisent sans choix visuels de notre part. La baisse de l'accuracy dans ce nouveau modèle pouvait être dû alors à deux facteurs :

- Le premier est l'augmentation du nombre de classes.

- Et le fait de ne pas avoir choisi visuellement nos 20 classes pouvait aussi être un facteur.

Pour nous donner une idée sans avoir à rechercher 20 classes très spécifiques, nous avons pensé à utiliser 20 “species” plutôt que 20 “genus”. Chaque “species” étant plus spécifique, et donc potentiellement plus proche visuellement, cela nous permettait de ressentir ce qui était dû au nombre de classes, où alors à la spécificité de chaque classe.

Nous avons pu vérifier que les performances s'améliorent en travaillant au niveau des Espèces :

- 70% de précision sur 20 Genus
- plus de 80% de précision sur 20 Species

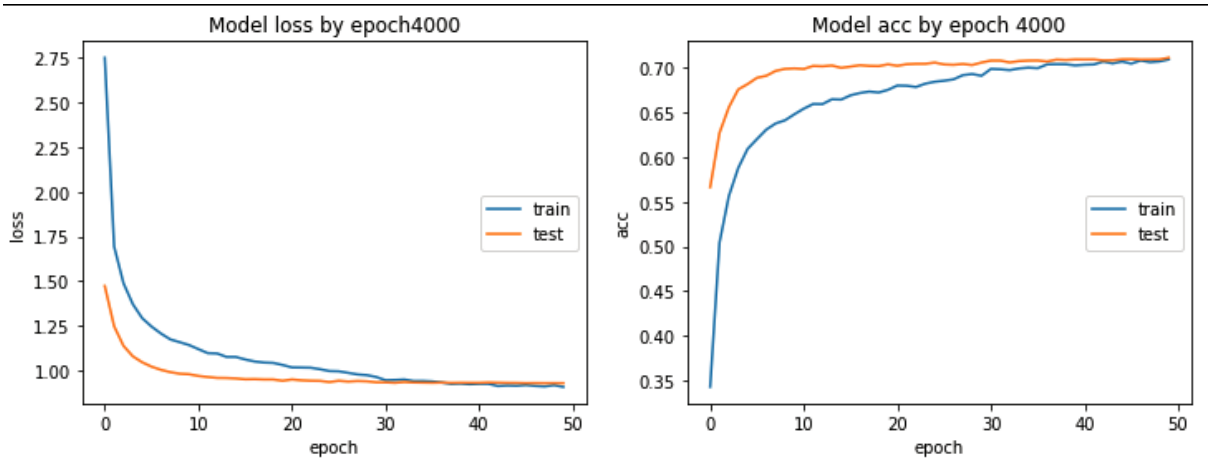
Cela peut s’expliquer par de plus grandes différences visuelles entre chaque species plutôt qu’entre chaque genre, donc une plus grande facilité à les classer.
La sensibilité du modèle semble donc plus être orientée sur la cohérence de chaque classe que par le nombre de classes à découvrir.
Cela ajoute de la pertinence à notre nettoyage sur les enveloppes.

Nombre de photos par classe

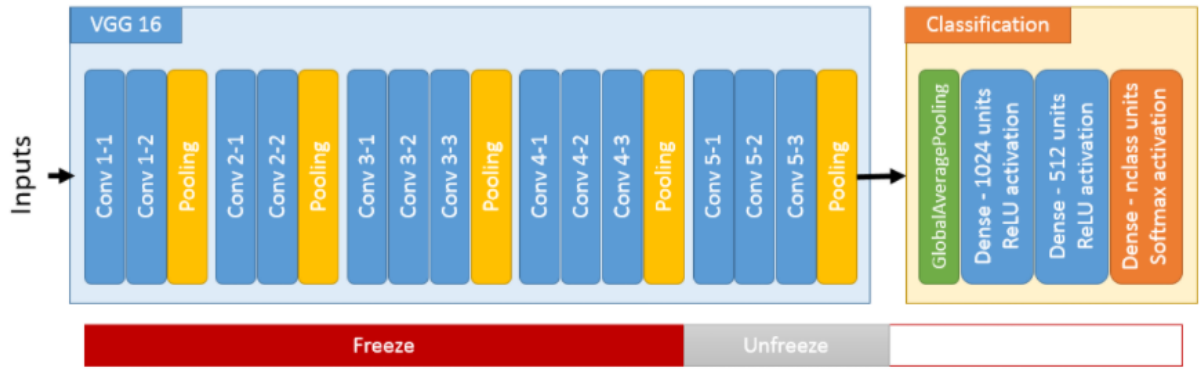
Nous avons progressivement augmenté le nombre de photos utilisées pour entraîner les modèles avec la progression des téléchargements. La seule différence notable est la baisse du taux d’accuracy quand on a augmenté le nombre de classes. Nous sommes passés de 82% à 70% en passant de 10 à 20 classes (les genus).

Unfreeze des dernières couches

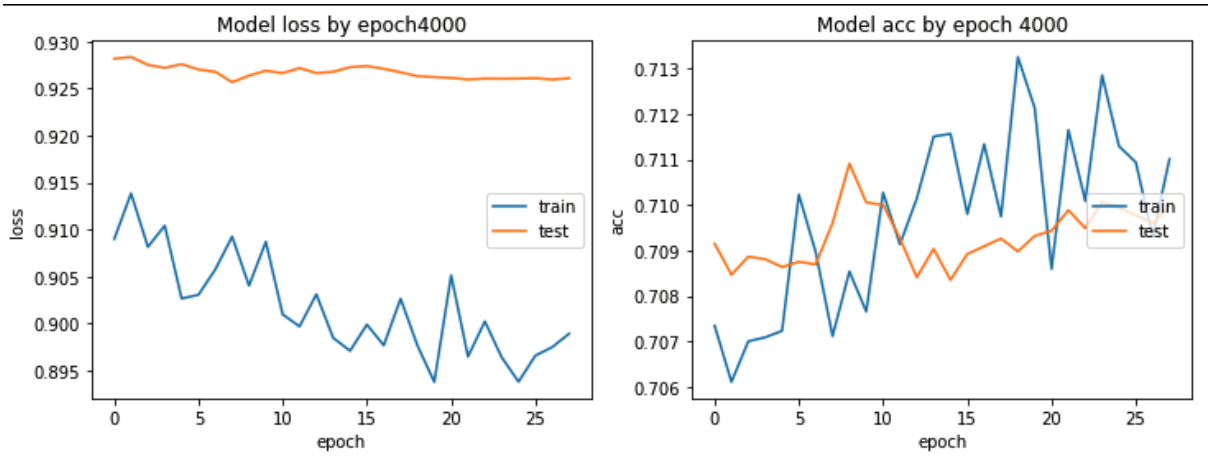
Nous avons voulu tester l’impact d’un second entraînement pratiqué en dégelant les 4 dernières couches du modèle. Pour ce test, le modèle utilisé est le VGG16. Voici les résultats de la première étape avec toutes les couches gelées:



Une fois terminé, nous autorisons la modification des poids des 4 dernières couches du modèle comme indiqué sur le schéma suivant :



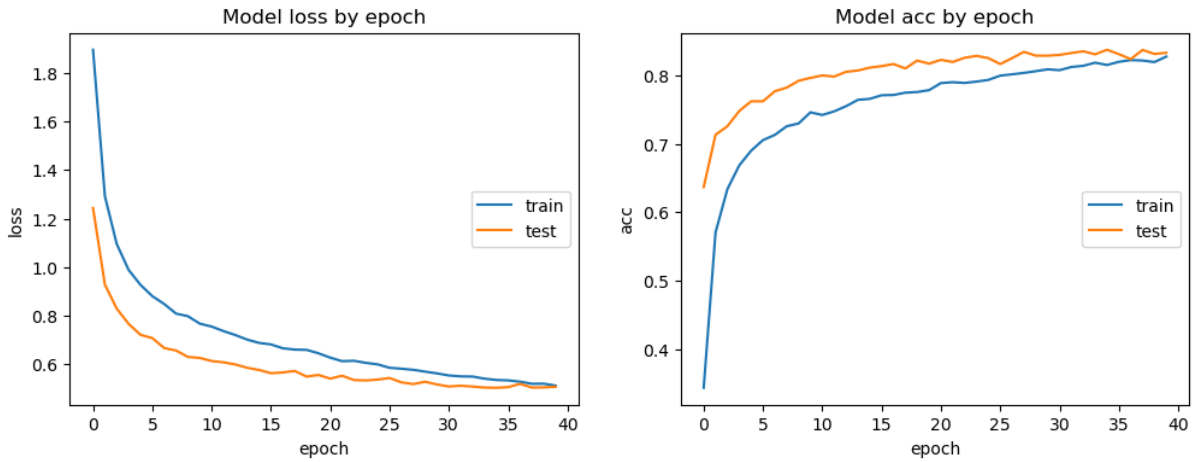
Puis nous relançons l'entraînement du modèle une seconde fois. Nous obtenons les graphes suivants :



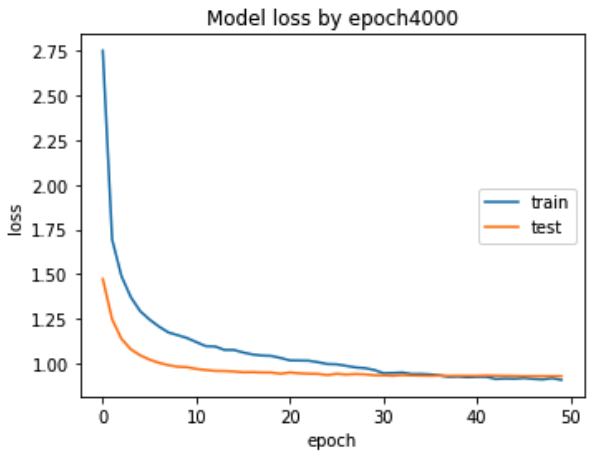
Nous avons constaté que le gain était relativement faible pour notre exemple avec 20 classes, nous verrons ce qu'il en est sur le calcul final avec 64 classes.

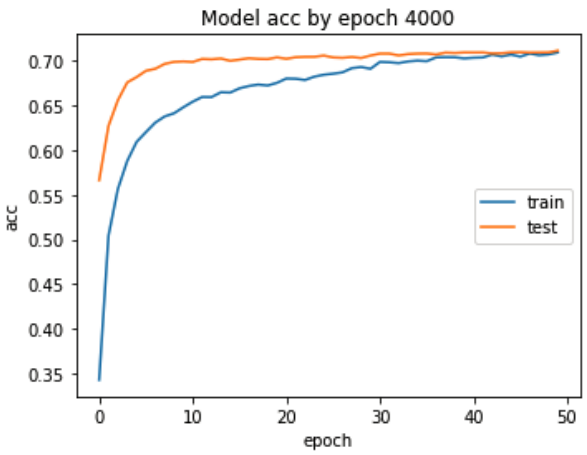
Résultats :

Résultats ResNet50 : LR 0,00001, 50 Epochs arrêté à la 39eme, 2000 images par classes ("Genus")

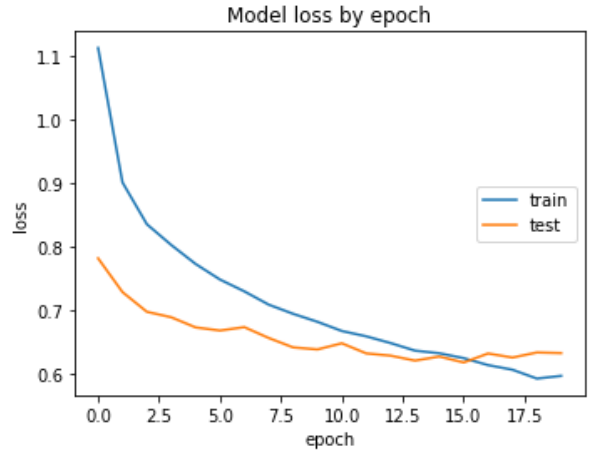


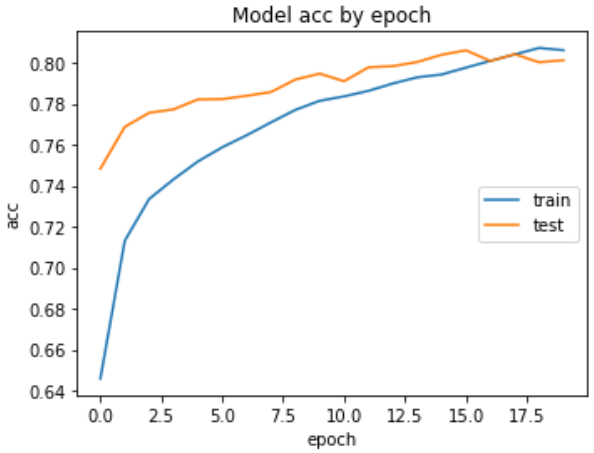
Résultats VGG16 : LR 0.001 puis ReduceLROnPlateau, 50 epochs, 4000 images, 22 genus





Résutats EfficientNetB0 : 20 Genus, 5000 photos par classes

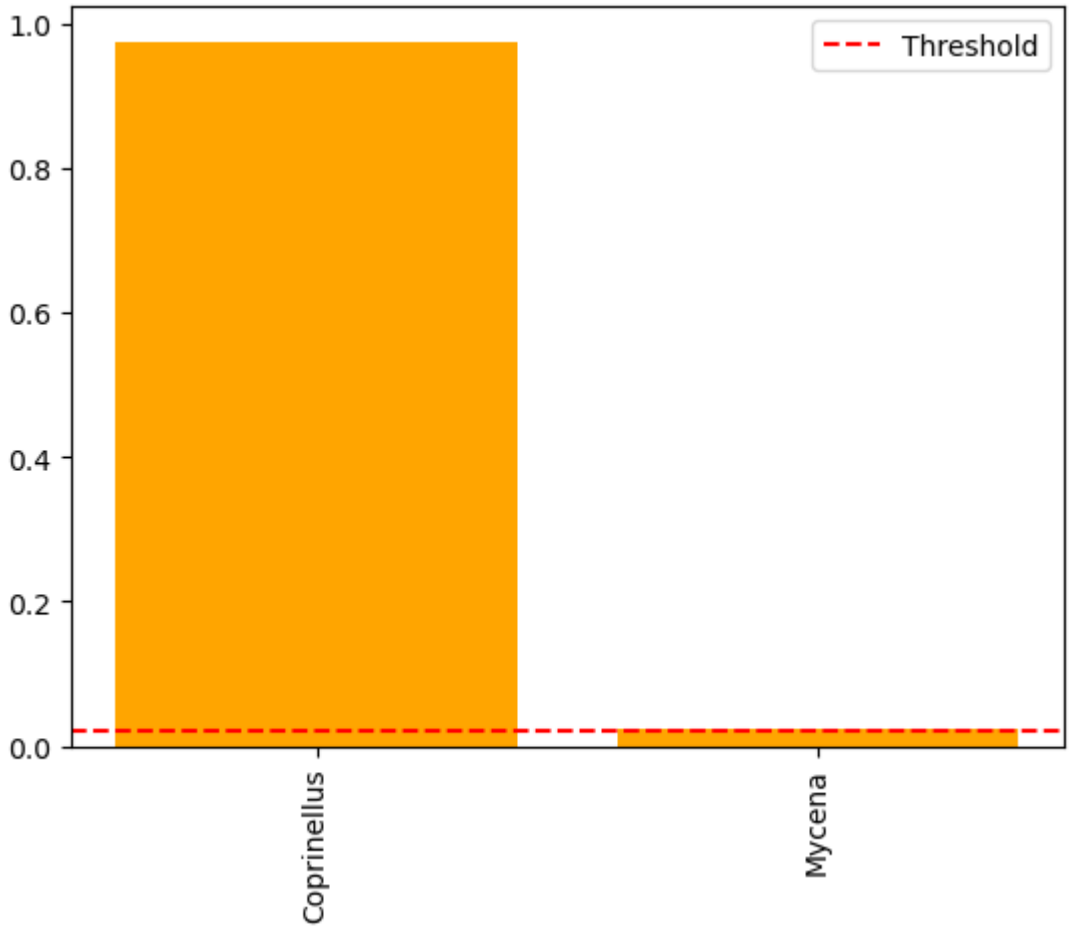




On remarque que les modèles ResNet50 et EfficientNetB0 nous donnent des résultats similaires et légèrement supérieurs au VGG16.
Pour le test sur 64 classes, nous ajouterons une métrique supplémentaire pour afficher le top-k en utilisant (tf.keras.metrics.TopKCategoricalAccuracy)

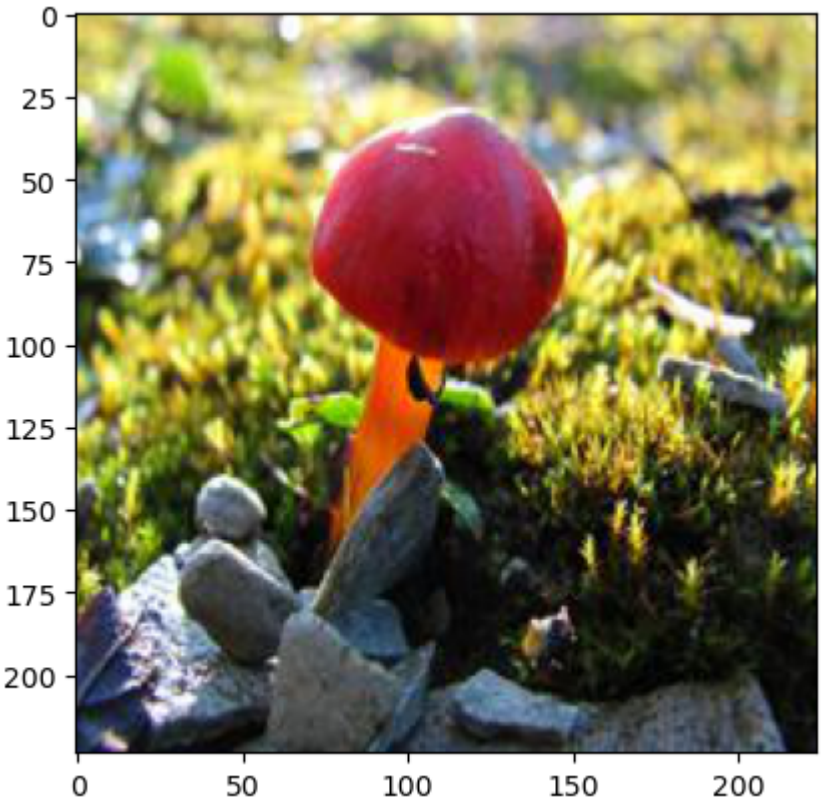
Prédictions

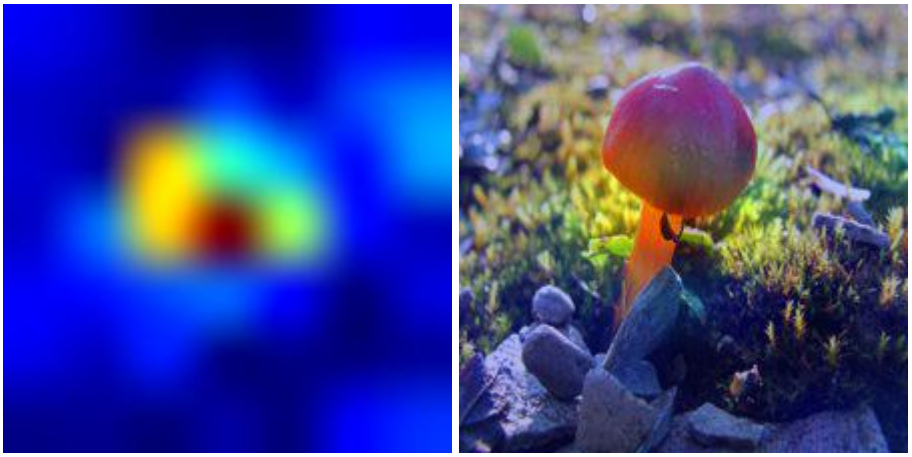
Une fois le modèle entraîné, nous avons un fichier “*.h5” qui peut être utilisé dans le cadre d’une application qui pourra prédire la classe d’un champignon.
La fonction model.fit() nous retourne la liste des probabilités de correspondance pour l’ensemble des classes. Nous avons écrit une fonction my_decode_prediction() pour nous retourner une liste ne contenant que les n meilleures probabilités et en supprimant les résultats en dessous d’un certain seuil.
Nous pouvons alors afficher un graphique avec la liste réduite, voici un exemple :



Interprétabilité

Les résultats que nous avons obtenus, que ce soit sur la valeur de la précision ou nos tests aléatoires, sont positifs. On rappelle qu'une prédiction aussi bonne que le hasard serait de 5% pour 20 classes. Nous avons souhaité vérifier les parties des images qui servaient à identifier les classes, notamment pour vérifier si les zones les plus actives correspondaient bien aux parties des images montrant un champignon ou bien un arbre ou des feuilles. Pour ce faire, nous avons intégré la méthode de Grad Cam pour calculer une heatmap qui permet de localiser les parties d'une photo qui ont servi à la classification. La palette utilisée est bleue dans les zones non pertinentes et de plus en plus rouge pour les zones pertinentes.





Cette fonction nous permettra ainsi de vérifier la robustesse de notre modèle. Cible-t-il les champignons (ou partie) ou l'environnement ?

Conclusion

Nous avons pu entraîner 3 modèles différents qui donnent des précisions supérieures à 70% sur les genres et plus de 80% sur les espèces. Les tout derniers résultats seront présentés dans le rapport final, notamment le résultat d'un entraînement sur 64 classes de 9000 photos par classe.