**Guide d’utilisation pour l’Office Français de la Biodiversité**

*Auteur : Marie Joigneau (stagiaire encadrée par Olivier Gimenez)*

Ci-dessous un guide d’utilisation afin d’appliquer l’Intelligence Artificielle pour l’identification de lynx avec une base de données déjà annotée.

**Le meilleur modèle nécessite des images de taille 260 x 260.**

**Le seuil est de 0.53.**

**Etape 1 : Préparer les images**

Cette étape consiste à préparer les images afin qu’elle puisse être conforme au modèle et pertinente.

**Option 1 : Les images sont déjà centrées sur le lynx et ont la taille demandée par le modèle.**

Créez un fichier *OFB*. Créez un sous fichier *dataset\_ready* comme ci-dessous.

Une image contenant texte, Police, conception

Description générée automatiquement

Mettez vos images prêtes dans le sous fichier nommé *dataset\_ready*. Les images doivent être classées comme ci-dessous. Toutes les images d’un même individu sont rassemblées dans un dossier de cet individu. Les images à prédire sont dans le dossier *0new*.

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

**Option 2 : Les images sont déjà centrées sur le lynx mais ont des tailles différentes de celle demandée pour le modèle.**

**Mise en place des images**

Créez un fichier *OFB*. Créez un sous fichier *dataset\_raw* comme ci-dessous.

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

Mettez vos images dans le sous fichier nommé *dataset\_raw*. Les images doivent être classées comme ci-dessous. Toutes les images d’un même individu sont rassemblées dans un dossier de cet individu. Les images à prédire sont dans le dossier *0new*.

Une image contenant texte, Police, capture d’écran, blanc

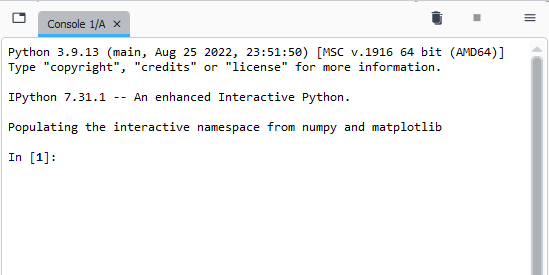
Description générée automatiquement

**Utilisation du code**

Ouvrez le fichier *1\_Pre-processing\_OFB\_resize-only.py* grâce à un utilitaire Python comme Spyder.

Lancez le code .

Deux questions vont vous être posées dans la console (voir exemple de la console ci-dessous).



1ère question : *Write here the size you want / Ecrivez ici la taille que vous désirez.*

Vous répondez 260 par exemple si le modèle nécessite des images de taille 260 x 260.

2ème question : *Write here the directory of the OFB folder (end by '/OFB') / Ecrivez ici le chemin d’acces du dossier OFB (fini par /OFB)*.

Vous répondez par exemple *D:/my\_code/OFB*.

Les résultats sont dans un nouveau dosser *dataset\_ready*.

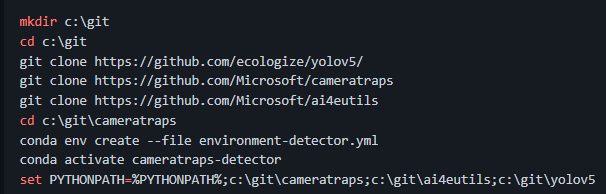
Vos images sont prêtes.

**Option 3 : Les images ne sont pas centrées sur le lynx et ont des tailles différentes de celle demandée pour le modèle.**

**Installer Megadetector**

Sur <https://github.com/microsoft/CameraTraps/blob/main/megadetector.md>, télécharger le modèle Megadetector mV5a. Le fichier, une fois téléchargé, se nomme *md\_v5a.0.0.pt*.

Importer les utilitaires pour finir d’installer Megadetector (voir ci-dessous) dans votre cmd (invite de commande).



**Mise en place des images**

Créez un fichier *OFB*. Créez un sous fichier *dataset\_raw* comme ci-dessous.

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

Mettez vos images dans le sous fichier nommé *dataset\_raw*. Les images doivent être classées comme ci-dessous. Toutes les images d’un même individu sont rassemblées dans un dossier de cet individu. Les images à prédire sont dans le dossier *0new*.

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

**Obtenir un fichier Megadetector**

Pour chaque obtention d’un fichier Megadetector, vous devez écrire sur votre terminal d’Anaconda, les 3 lignes suivantes :

To use run\_detector.py on Windows, when you open a new Anaconda prompt, don't forget to do this:

cd c:\git\CameraTraps

conda activate cameratraps-detector

set PYTHONPATH=%PYTHONPATH%;c:\git\cameratraps;c:\git\ai4eutils;c:\git\yolov5

Puis vous tapez dans le terminal :

python detection\run\_detector\_batch.py "c:\megadetector\md\_v5a.0.0.pt" "c:\some\_image\_folder" "c:\megadetector\test\_output.json" --output\_relative\_filenames --recursive --checkpoint\_frequency 10000

c:\megadetector\md\_v5a.0.0.pt : chemin d’accès du modèle Megadetector (doit se terminer par « OFB/md\_v5a.0.0.pt »)

c:\some\_image\_folder : chemin d’accès du dossier regroupant vos images (/!\ vous devez bien mettre un chemin d’accès se terminant par « …/OFB/dataset\_raw », étant donné que vos images sont dans ce dossier).

c:\megadetector\test\_output.json : chemin d’accès du résultat de Megadetector (doit se terminer par « OFB/*megadetector\_results.json*»)

Renommez si ce n’est déjà fait le fichier du résultat de Megadetector *megadetector\_results.json* et mettez-le dans le dossier */OFB*.

**Utilisation du code**

Ouvrez le fichier *1\_Pre-processing\_OFB\_crop-resize.py* grâce à un utilitaire Python comme Spyder.

Lancez le code .

Plusieurs questions vont vous être posées dans la console (voir exemple de la console ci-dessous).

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

1ère question : W*rite here the size you want / Ecrivez ici la taille que vous désirez.*

Vous répondez 260 par exemple si le modèle nécessite des images de taille 260 x 260.

2ème question : *Write here the directory of the OFB folder (end by '/OFB') / Ecrivez ici le chemin d’acces du dossier OFB (fini par /OFB)*.

Vous répondez par exemple *D:/my\_code/OFB*.

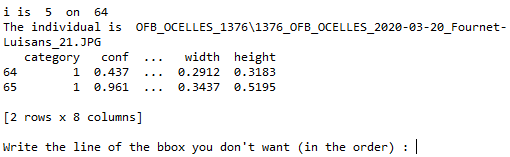
3ème question : Do you want to filter automatically? (yes/no) / voulez-vous filtrer automatiquement? (yes/no)

Si vous voulez que le code prenne automatiquement l’emplacement le plus probable de l’individu sur la photo, vous écrivez *yes*. Sinon vous écrivez *no*.

Si vous avez choisis le côté manuel, on va vous demandez d’écrire une lettre quand vous êtes prêt (Note the pictures you want to keep, and when you are ready, enter a letter / Notez les images à garder et écrivez une lettre quand vous êtes prêts).

Vous allez avoir toutes les propositions d’images croppées dans le dossier */OFB/ bbox\_pb*. Vous allez avoir par exemple les images idx10\_1376\_OFB\_OCELLES\_2020-03-20\_Fournet-Luisans\_21 et idx11\_1376\_OFB\_OCELLES\_2020-03-20\_Fournet-Luisans\_21 dans le dossier */OFB/bbox\_pb*.

Et vous allez devoir écrire l’index des propositions que vous ne gardez pas (ci-dessous si vous voulez garder la 1ère image vous notez 65.



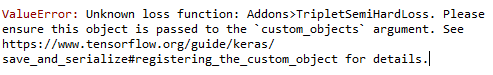
Une fois fini, supprimez le fichier */OFB/bbox\_pb*.

Les résultats sont dans un nouveau dosser *dataset\_ready*.

Vos images sont prêtes.

**Etape 2 : Lancer un modèle quelconque**

Dû à un problème de non-reconnaissance de la fonction de perte du modèle, vous devez lancer un modèle avec cette fonction de perte pour pouvoir réaliser l’étape suivante.



Ouvrez le fichier *2\_Model-construction\_OFB.py* grâce à un utilitaire Python comme Spyder.

Lancez le code .

Une question va être posées dans la console (voir exemple de la console ci-dessous).

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Question : *Write here the directory of the OFB folder (end by '/OFB') / Ecrivez ici le chemin d’acces du dossier OFB (fini par /OFB)*.

Vous répondez par exemple *D:/my\_code/OFB*.

**Etape 3 : Lancer le modèle et regarder les résultats**

**Option 1 : Vous voulez filtrer votre jeu de données par les dates / lieux pour la prédiction**

**Mise en place des métadonnées**

Mettez vos métadonnées sous la forme d’un csv nommé *lynx\_metadata.csv*. Il doit ressembler à l’exemple ci-dessous avec l’identifiant du lynx (*lynx\_ID*), les lieux où il a été identifié (*place*), les dates où il a été identifié (*date*) et le nom des différentes images (*picture*).

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Mettez les métadonnées dans le dossier */OFB*.

**Utilisation du code**

Ouvrez le fichier *5\_Model-prediction-already-dataset\_manuel\_OFB.py* grâce à un utilitaire Python comme Spyder.

Lancez le code .

Plusieurs questions vont vous être posées dans la console (voir exemple de la console ci-dessous).

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

1ère question : Write here the treshold / Ecrivez le seuil

Vous répondez le seuil donné.

2ème question : *Write here the directory of the OFB folder (end by '/OFB') / Ecrivez ici le chemin d’acces du dossier OFB (fini par /OFB)*.

Vous répondez par exemple *D:/my\_code/OFB*.

3ème question : Do you want to filter individuals? (yes/no) / Voulez-vous filtrer les individus (écrire yes/no).

Vous répondez par *yes.*

4ème question : Choose the year you want (none if not) / Choisissez l’année à filtrer (écrivez none sinon)

Vous répondez par exemple 2006.

5ème question : Choose the place you want (none if not) / Choisissez la date pour filtrer (écrivez none sinon)

Vous répondez par exemple *A.*

Les résultats sont sous la forme d’un csv nommé *resultats\_k-voisins\_manuel* dans */OFB*.

**Option 2 : Vous ne voulez pas filtrer votre jeu de données**

Ouvrez le fichier *5\_Model-prediction-already-dataset\_manuel\_OFB.py* grâce à un utilitaire Python comme Spyder.

Lancez le code .

Deux questions vont vous être posées dans la console (voir exemple de la console ci-dessous).

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

1ère question : Write here the treshold / Ecrivez le seuil

Vous répondez le seuil donné.

2ème question : *Write here the directory of the OFB folder (end by '/OFB') / Ecrivez ici le chemin d’acces du dossier OFB (fini par /OFB)*.

Vous répondez par exemple *D:/my\_code/OFB*.

3ème question : Do you want to filter individuals? (yes/no) / Voulez-vous filtrer les individus (écrire yes/no).

Vous répondez par *no.*

Les résultats sont sous la forme d’un csv nommé *resultats\_k-voisins\_manuel* dans */OFB*.