Reconnaissances d'abeilles dans des images par apprentissage profond

Fabio PEREIRA DE ARAUJO

Deuxième année - Département Sciences du Numérique

2021-2022

Table des matières

1 Entraînement

La base convolutive associée à chacun des modèles est celle de **ResNet50** pré-entraînée sur ImageNet :

```
conv_base = keras.applications.resnet50.ResNet50(
    include_top=False,
    weights='imagenet',
    input_tensor=None,
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    pooling=None,
    classes=nb_classes,
)
```

1.1 ResNet50 sans augmentation de données

1.1.1 Description de l'architecture du modèle

Le premier réseau entraîné est un **ResNet50** sans augmentation de données, mais avec une régularisation L_2 de $1e-4$:

```
# Creation du modele
model = keras.Sequential(
    [
        conv_base,
        layers.GlobalAveragePooling2D(),
        layers.Dense(nb_classes, kernel_regularizer=regularizers.L2(1e-4),
    activation='softmax')
    ]
)
```

C'est un réseau comportant 23,733,191 de paramètres dont 23,680,071 peuvent être entraînés.

L'optimiseur choisi est un Stochastic Gradient Descent (SGD) avec un learning rate $\alpha = 1e-2$ et un momentum de $\eta = 0.9$:

```
model.compile(optimizers.SGD(learning_rate=1e-2, momentum=0.9), loss='
    categorical_crossentropy', metrics=['categorical_accuracy', tf.keras.metrics.
    Precision(), tf.keras.metrics.Recall()])
```

Pour l'entraînement du modèle, trois callbacks ont été utilisés : un checkpoint (pour sauvegarder les poids au meilleur moment (contrôle de la validation_accuracy), un early stopping ainsi qu'un callback pour diviser le learning rate par 10 si on atteint un plateau.

```
# Les callbacks
model_checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(
    filepath='./drive/MyDrive/Stage2A/ResNet50/best_model',
    save_weights_only=True,
    monitor='val_categorical_accuracy',
    mode='max',
    verbose=1,
    save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.01,
    patience=7,
    mode="auto")

reduce_lr_cb = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1,
     patience=5, min_lr=0.001)
```

1.1.2 Résultats obtenus

Le modèle développé permet d'obtenir les résultats suivants :

categorical accuracy	0.9993
val accuracy	0.49210
test accuracy	0.47987

1.2 ResNet50 avec augmentation de données

1.2.1 Description de l'architecture du modèle

Le second réseau entraîné est un **ResNet50** avec augmentation de données : un RandomFlip horizontal ainsi qu'un RandomRotation entre 0 et 100 degrés. Une régularisation L_2 de $1e-4$ est également utilisée :

```
model = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        conv_base,
        layers.GlobalAveragePooling2D(),
        layers.Dense(nb_classes, kernel_regularizer=regularizers.L2(1e-4),
    activation='softmax')
    ]
)
```

L'optimiseur choisit ainsi que les callbacks utilisés sont les mêmes que le modèle précédent.

1.2.2 Résultats obtenus

Le modèle développé permet d'obtenir les résultats suivants :

categorical accuracy	0.9698
val accuracy	0.58294
test accuracy	0.57380

1.2.3 Analyse des résultats

Pour analyser les résultats, on peut utiliser plusieurs outils comme la matrice de confusion et les notions de précision, sensibilité et F1-score. Ce qui serait intéressant d'étudier, c'est pour les classes avec les plus mauvaises précisions et sensibilités, déterminer si la confusion de l'espèce se fait souvent avec une autre espèce.

En fixant à 5 le nombre minimum de confusion entre deux espèces par exemple, on constate que l'on confond souvent : *Chelostoma florisomne* et *Heriades truncorum* (5 fois pour des échantillons de 17 pour Chelostoma florisomne et 15 pour Heriades truncorum). On confond également souvent : *Xylocopa violacea* et *Xylocopa valga* (7 fois pour des échantillons de 20 pour les deux espèces). Néanmoins, ce ne sont pas les espèces avec la précision la plus faible (précision comprise entre 0.4 et 0.6 environ). Certaines espèces ont des précisions nettement inférieurs et ont été confondus avec plusieurs autres espèces (*Bombus pascuorum* par exemple).

1.3 ResNet101 avec augmentation de données

1.3.1 Description de l'architecture du modèle

Le troisième réseau qui a permi de battre les résultats précédents est un **ResNet101** avec augmentation de données de même type que le second réseau (RandomRotation entre 0 et 100 degrés, RandomFlip horizontal). Une régularisation L_2 de $1e-4$ est aussi utilisée :

```
conv_base = keras.applications.resnet.ResNet101(
    include_top=False,
    weights='imagenet',
    input_tensor=None,
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    pooling=None,
    classes=nb_classes,
```

```
)

model = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        conv_base,
        layers.GlobalAveragePooling2D(),
        layers.Dense(nb_classes, kernel_regularizer=regularizers.L2(1e-4),
    activation='softmax')
    ]
)
```

L'optimiseur choisi est encore une fois SGD mais cette fois $lr = 1e - 3$, de ce que j'ai pu expérimenté, cette valeur de learning rate est plutôt efficace ! Les callbacks utilisés sont : Early Stopping mais cette fois-ci avec une patience de 8 et une surveillance de l'accuracy de la validation car sinon l'entraînement s'arrêtait alors qu'il y avait encore des progrès possible. Il y a aussi un checkpoint et une réduction du learning rate si la loss de la validation ne décroit plus.

1.3.2 Résultats obtenus

categorical_accuracy	0.9684
val_accuracy	0.6153
test_accuracy	0.62589

1.3.3 Analyse des résultats

On utilise les mêmes outils que précédemment : la précision, la sensibilité et le F1_score.

Mon objectif est ici de déterminer pour une espèce donnée ayant une faible précision (i.e, une faible proportion de bonne détection), avec quelles autres espèces on a pu la confondre dans un objectif de déterminer les problèmes qui peuvent exister au sein du dataset par exemple ou encore tenter de trouver des raisons pour lesquelles on a pu confondre ces espèces.

Pour faire cela, j'utilise une fonction qui me permet pour une matrice de confusion donnée, d'obtenir toutes les espèces pour lesquelles on a eu un certain nombre (déterminé par un seuil à fixer) de confusions avec d'autres espèces (par exemple : *Bombus hortorum* est souvent confondue avec *Bombus ruderatus* en fixant le seuil à 5.

Voici ce que l'on obtient en fixant le seuil à 3 :

```
{'Andrena bicolor': {'Andrena clarkella': 3},
 'Andrena vaga': {'Andrena cineraria': 3},
 'Anthidium florentinum': {'Anthidium manicatum': 4},
 'Anthidium septemspinosum': {'Anthidium manicatum': 3},
 'Anthophora bimaculata': {'Anthophora plumipes': 3},
 'Bombus campestris': {'Bombus vestalis': 3},
 'Bombus hortorum': {'Bombus ruderatus': 5},
 'Bombus humilis': {'Bombus muscorum': 5,
                    'Bombus rupestris': 3},
 'Bombus lapidarius': {'Bombus bohemicus': 3,
                       'Bombus rupestris': 3},
 'Bombus ruderatus': {'Bombus hortorum': 3},
 'Bombus vestalis': {'Bombus bohemicus': 3},
 'Chelostoma florisomne': {'Heriades truncorum': 3},
 'Melecta albifrons': {'Andrena cineraria': 3,
                       'Xylocopa violacea': 4},
 'Osmia caerulescens': {'Heriades truncorum': 3},
 'Xylocopa valga': {'Xylocopa violacea': 3},
 'Xylocopa violacea': {'Xylocopa valga': 8}}
```

FIGURE 1 – Confusions obtenues, seuil = 3

On constate ici par exemple que l'espèce *Xylocopa violacea* est souvent confondue avec l'espèce *Xylocopa valga* (8 confusions pour 20 images à prédire). C'était déjà le cas précédemment.

Étudions un peu plus en détail les images qui ont été mal classées pour essayer d'expliquer cette mauvaise classification. Par exemple, regardons ce qu'il s'est passé pour l'espèce *Xylocopa violacea* :

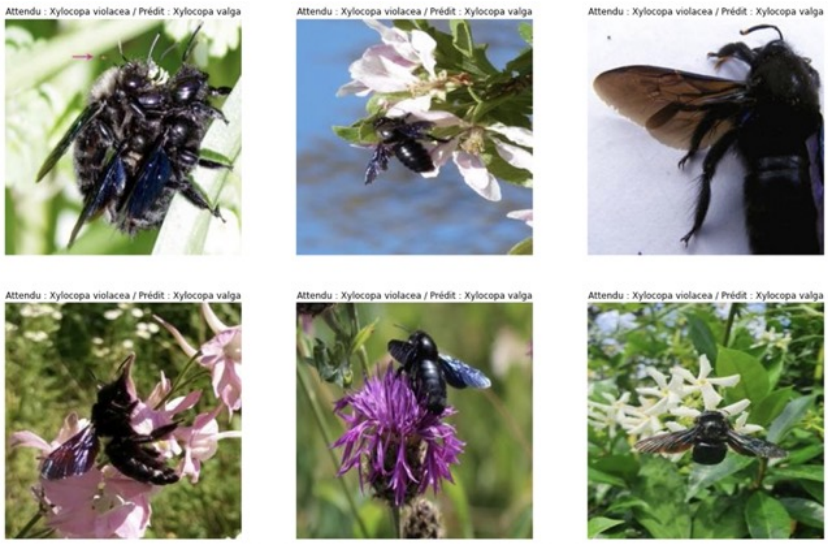

FIGURE 2 – Attendu : Xylocopa violacea, Prédiction : Xylocopa valga

La première image a peut être posée problème car il y a deux abeilles sur la même image. La seconde image et la dernière sont quant à elles peut être un peu trop éloignées ce qui a pu induire en erreur le réseau de neurones si les images de l'ensemble d'entraînement étaient proches par exemple. Néanmoins, il serait intéressant d'étudier pourquoi la confusion se fait souvent avec l'espèce *Xylocopa valga*. En effet, les images peuvent paraître d'assez bonne qualité pour permettre une bonne prédiction, le réseau de neurones a peut être du mal à différencier les deux espèces par rapport aux caractéristiques apprises.

Voici une image de l'espèce *Xylocopa valga* utilisée dans l'ensemble d'apprentissage (provenance : iNaturalist) :

FIGURE 3 – Une image d'une *Xylocopa valga*

À priori, on pourrait penser que la couleur noire est un facteur déterminant pour la classification de cette espèce d'abeille, ce serait peut-être la raison pour laquelle les confusions se font avec cette espèce ! Par ailleurs, en cherchant une image de l'ensemble d'entraînement, je suis tombé sur quelques images très éloignées sur lesquelles l'abeille est assez peu visible, voici quelques exemples :

Ces images n'ont peut être pas permis au réseau de neurones d'apprendre des caractéristiques suffisantes sur cette espèce et ne prend peut être sa décision qu'en grande partie sur la couleur noire.

Regardons maintenant les mauvaises classifications de l'espèce *Bombus humilis* :

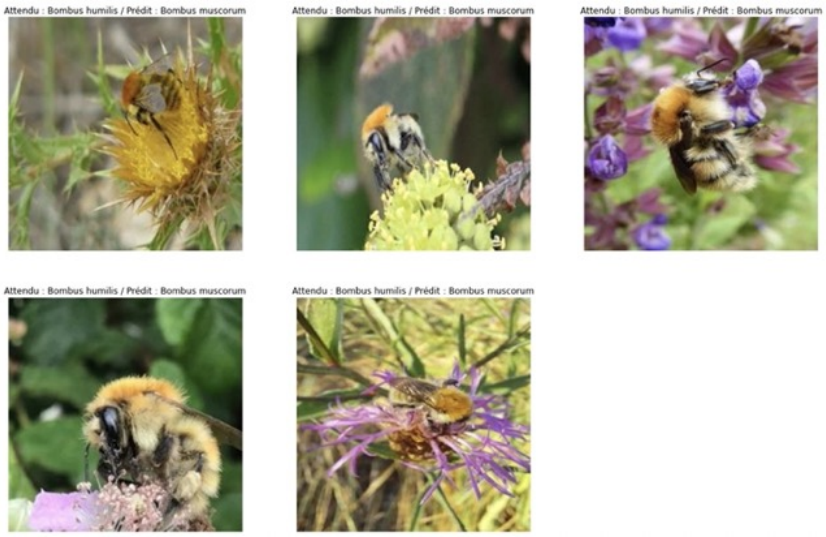

FIGURE 5 – Attendu : Bombus humilis, Prédiction : Bombus muscorum

Pour la première et la dernière image, il est difficile de distinguer l'abeille de la fleur sur laquelle elle se trouve, cela a peut être joué en faveur d'une mauvaise prédiction du réseau de neurones. Par ailleurs, les deux premières images ainsi que la dernière sont également assez éloignées de l'abeille, ceci ne doit pas aider le réseau de neuronnes à trouver les caractéristiques permettant d'effectuer une bonne prédiction.

Voici une image de *Bombus muscorum* de l'ensemble d'apprentissage (provenance : iNaturalist) :

FIGURE 6 – *Bombus muscorum*

Sur la première image par exemple, il me semble que l'abdomen de la *Bombus humilis* est similaire à celui de la *Bombus muscorum*, c'est peut être une caractéristique qui a été retenue et qui est à l'origine de l'erreur de prédiction. Par ailleurs, le thorax des deux abeilles a une forme et une "texture" similaire bien qu'une couleur différente, le réseau de neurones regarde peut être la présence d'un tel thorax pour prendre sa décision ce qui peut provoquer une erreur de prédiction entre les deux espèces.

Par ailleurs, encore une fois, lors de la recherche d'images dans l'ensemble d'apprentissage, je suis tombé sur des images probablement peu pertinentes voire contre-productives pour le réseau de neurones, en voici quelques exemples :

Les abeilles sont très peu visibles sur ces images, elles sont donc peu pertinentes pour apprendre des caractéristiques permettant de classifier cette espèce ! Ainsi, un travail sur la base de données utilisées pour l'apprentissage est nécessaire pour améliorer les résultats !

1.4 ResNet50 avec augmentation de données (utilisation de Sequence)

1.4.1 Description de l'architecture du modèle

Cette fois-ci, j'ai repris un ResNet50 avec une augmentation de données en utilisant Sequence pour avoir des augmentations de données un peu plus poussées et pouvoir également randomiser les indices à chaque epoch :

```
from albumentations import (Compose, Rotate, HorizontalFlip, Affine)
import albumentations as A

AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5)
])
```

À chaque epoch, les indices sont randomisés, la probabilité choisie est arbitraire, en effet, aucun des deux articles lus ne fournissaient de précisions quant aux probabilités utilisées pour ces augmentations de données. J'ai de nouveau utilisé l'early stopping (patience de 8 sur la validation accuracy), un checkpoint ainsi qu'une division du learning rate si plateau au niveau de la validation loss (on divise par 10 à chaque fois, début à $1e-3$ limitation à $1e-5$ avec une patience de 5).

L'optimiseur est toujours SGD (momentum de 0.9, learning rate de $1e-3$). Cependant, pas de pondération des classes cette fois.

1.4.2 Résultats obtenus

Le modèle développé permet d'obtenir les résultats suivants :

categorical accuracy	0.9332
val accuracy	0.6351
test accuracy	0.6180

On a eu un gain sur la validation accuracy mais une perte sur la test accuracy, les résultats sont donc très comparables, néanmoins, il y a peut être des améliorations si on applique la Sequence au ResNet101.

1.4.3 Analyse des résultats

En étudiant la matrice de confusion, on peut en fixant le seuil à 3, extraire les informations suivantes :

```
{'Andrena nitida': {'Andrena haemorrhoa': 3},
 'Andrena thoracica': {'Andrena bicolor': 3},
 'Bombus argillaceus': {'Bombus ruderatus': 3},
 'Bombus hortorum': {'Bombus ruderatus': 3},
 'Bombus humilis': {'Bombus muscorum': 4,
                    'Bombus pascuorum': 3},
 'Bombus hypnorum': {'Bombus pascuorum': 5},
 'Bombus ruderatus': {'Bombus argillaceus': 3,
                      'Bombus hortorum': 3},
 'Bombus terrestris lusitanicus': {'Bombus hortorum': 3},
 'Bombus vestalis': {'Bombus bohemicus': 3,
                     'Bombus lucorum': 3},
 'Dasypoda hirtipes': {'Andrena clarkella': 3},
 'Xylocopa valga': {'Xylocopa violacea': 7},
 'Xylocopa violacea': {'Xylocopa valga': 5}}
```

FIGURE 8 – Confusions obtenues, seuil = 3

On constate comme pour le ResNet101 la présence de certaines espèces : *Bombus humilis* confondue avec *Bombus muscorum* mais aussi confondue ici avec *Bombus pascuorum* alors que ResNet101 la confondait plutôt avec *Bombus rupestris*, *Bombus ruderatus* confondue avec *Bombus hortorum*, *Bombus vestalis* avec *Bombus bohemicus*, ou encore *Xylocopa valga* avec *Xylocopa violacea*. En fait, on constate pas mal de confusions au sein des Bombus et entre les Bombus, ou encore entre les Andrena ou entre les Xylocopa. Il serait peut-être intéressant de ne pas regarder uniquement le top 1-accuracy mais plutôt un top 2-accuracy ou top 3-accuracy.

On retrouve d'ailleurs pour *Bombus humilis* pratiquement les mêmes images qui ont été mal classifiées qu'avec le ResNet101 :

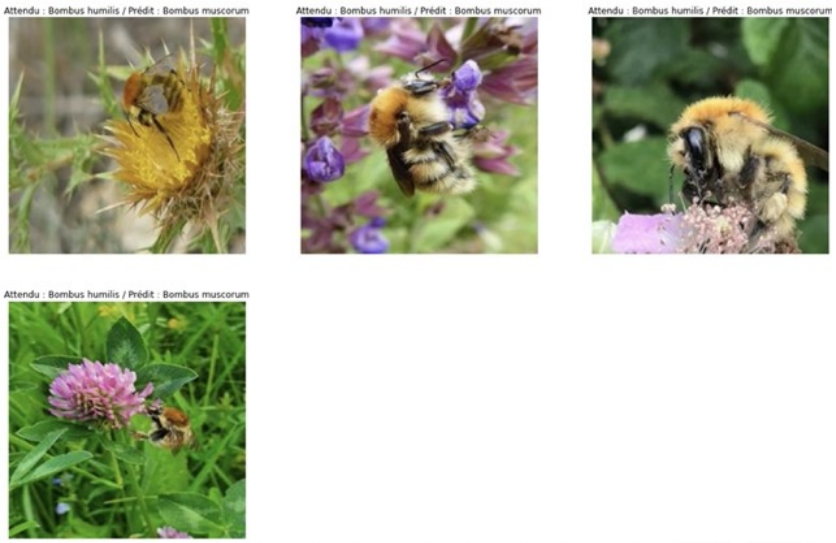

FIGURE 9 – *Attendu : Bombus humilis, Prédiction : Bombus muscorum*

Il semblerait donc que les deux réseaux aient appris des caractéristiques similaires.

1.5 ResNet101 avec augmentation de données (Sequence)

L'utilisation de la Sequence pour ResNet50 avait permis d'améliorer les résultats, essayons donc d'effectuer une augmentation de données avec Sequence mais cette fois en utilisant ResNet101.

1.5.1 Augmentation de données

Voici les augmentations de données considérées :

```
AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    VerticalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5),
])
```

1.5.2 Résultats obtenus

Le modèle développé permet d'obtenir les résultats suivants :

categorical accuracy	0.9629
val accuracy	0.6682
test accuracy	0.6488

1.6 ResNet101 avec augmentation de données (Sequence) et normalisation des données

Pour le moment, je n'ai pas normalisé les données pour l'entraînement du modèle, il serait intéressant de le faire pour essayer encore de gagner en précision.

J'ai également ajouté une autre augmentation de données sur la luminosité et le contraste, voici désormais les augmentations considérées :

```
AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    VerticalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5),
    RandomBrightnessContrast(p=0.5)
])
```

En ce qui concerne la normalisation des données, elle se fait dans la Sequence de la manière suivante :

```
def __getitem__(self, idx):
    ...
    # Normalisation des donees
    batch_x = tf.keras.applications.resnet.preprocess_input(batch_x)

    return batch_x, batch_y
```

Il faut également normaliser les données de validation en utilisant cette fonction (par exemple en récupérant les données avec as_numpy_iterator().

1.6.1 Résultats obtenus

categorical accuracy	0.9372
val accuracy	0.6754
test accuracy	0.6582

1.6.2 Analyse des résultats

Comme pour les autres modèles, on s'intéresse à l'analyse de la matrice de confusion avec la même méthode que précédemment. Voici ce que l'on obtient en fixant le seuil à 3 :

```
{'Andrena bicolor': {'Andrena clarkella': 3},
 'Andrena nitida': {'Andrena thoracica': 3},
 'Andrena thoracica': {'Andrena bicolor': 3},
 'Anthidium florentinum': {'Anthidium septemspinosum': 5},
 'Bombus hortorum': {'Bombus ruderatus': 3},
 'Bombus hypnorum': {'Bombus pascuorum': 4},
 'Bombus lapidarius': {'Bombus rupestris': 3},
 'Bombus pascuorum': {'Bombus humilis': 5},
 'Bombus vestalis': {'Bombus bohemicus': 3},
 'Nomada goodeniana': {'Nomada lathburiana': 3},
 'Xylocopa valga': {'Xylocopa violacea': 3},
 'Xylocopa violacea': {'Xylocopa valga': 6}}
```

FIGURE 10 – Confusions obtenues, seuil = 3

On constate une fois de plus que l'espèce *Xylocopa violacea* est souvent confondue avec l'espèce *Xylocopa valga* (6 confusions pour un support de 20 dans le test dataset). Voici les images qui ont été mal classées, ainsi que leurs heatmaps pour s'intéresser à ce qu'à regarder le réseau pour prendre sa décision :

FIGURE 11 – Attendu : *Xylocopa violacea*, Prédiction : *Xylocopa valga*

On constate que le réseau prend sa décision en observant les zones de l'image qui sont identifiées comme étant des abeilles. Les erreurs semblent donc provenir bel et bien de features apprises très similaires pour les deux classes ce qui pose problème au réseau pour les différencier. Il serait donc peut être intéressant d'analyser les performances sur un top2 accuracy ou un top3 accuracy par exemple, voici ce que l'on obtient pour ce modèle :

top 2 test accuracy	0.7932
top 3 test accuracy	0.8374

On constate également que pour l'image en haut et au milieu, le réseau s'est focalisé sur une des deux abeilles pour prendre sa décision.

1.7 ResNet50, Sequence, initialisation des poids INat2021

categorical_accuracy	0.8458
val_accuracy	0.60506
test_accuracy	

1.8 ResNet101, cap500 (pas de class weight)

1.8.1 Modèle

```
conv_base = keras.applications.resnet.ResNet101(
    include_top=False,
    weights='imagenet',
    input_tensor=None,
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    pooling=None,
    classes=nb_classes,
)

model = keras.Sequential(
    [
        conv_base,
        layers.GlobalAveragePooling2D(),
        layers.Dense(nb_classes, kernel_regularizer=regularizers.L2(1e-4),
    activation='softmax')
    ]
)
```

1.8.2 Augmentation de données

```
AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    VerticalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5),
    RandomBrightnessContrast(p=0.5),
    ChannelShuffle(p=0.5)
])
```

1.8.3 Optimiseur et loss function

```
model.compile(optimizers.SGD(learning_rate=1e-3, momentum=0.9), loss='
    categorical_crossentropy', metrics=['categorical_accuracy', tf.keras.metrics.
    Precision(), tf.keras.metrics.Recall()])
```

1.8.4 Callbacks

Les callbacks utilisés sont : un checkpoint, un early stopping contrôlant la l'accuracy de la validation avec une patience de 8 et une réduction du learning rate contrôlant la validation loss avec une patience de 5 (division du learning rate par 10, limitation à $1e-5$).

1.8.5 Constitution du dataset cap500

Pour constituer le dataset cap500, j'ai tout d'abord récupéré l'intégralité des images de la base de données concernant les 71 espèces étudiées. J'ai ensuite utilisé le ResNet101 de la partie 1.6 (Sequence et normalisation des données) afin d'effectuer des prédictions sur ces images.

Une fois ces prédictions effectuées, j'ai récupéré les images ayant une probabilité supérieure à 0.9, et j'ai défini une limitation à 500 images pour les classes avec le plus d'images. L'objectif de ce procédé est de récupérer les images dont on était le plus sûr de la prédiction afin d'essayer d'entraîner le modèle avec de meilleures images, et donc potentiellement améliorer les résultats.

1.8.6 Résultats obtenus

categorical_accuracy	0.9035
val_accuracy	0.84552
test_accuracy	0.84443

On constate une grande amélioration dans les résultats à l'aide d'un nouveau dataset d'apprentissage (le premier (cap200) ayant été obtenu en sélectionnant des images aléatoirement dans la base de données).

1.8.7 Analyse des résultats

Étudions un peu plus en détails les résultats obtenus. Tout d'abord, analysons un peu la matrice de confusion et affichons quelques heatmaps d'images mal classifiées.

```
{'Andrena clarkella': {'Colletes cunicularius': 3},
 'Anthidiellum strigatum': {'Anthidium florentinum': 3},
 'Bombus bohemicus': {'Bombus vestalis': 3},
 'Bombus lucorum': {'Bombus vestalis': 3},
 'Dasypoda hirtipes': {'Halictus scabiosae': 3}}
```

FIGURE 12 – Confusions obtenues, seuil = 3

Par exemple, regardons les mauvaises classifications de l'espèce Anthidiellum Strigatum :

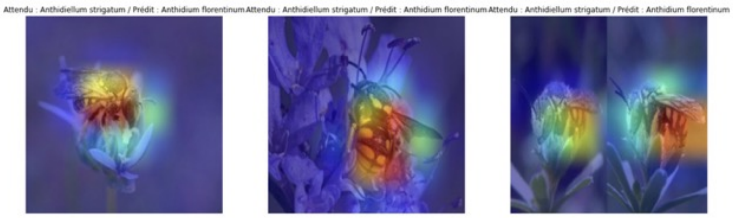

FIGURE 13 – Attendu : *Anthidiellum Strigatum*, Prédiction : *Anthidium Florentinum*

FIGURE 14 – Attendu : *Anthidiellum Strigatum*, Prédiction : *Anthidium Florentinum*

On constate encore des problèmes dans la base de données avec la troisième image qui en contient en fait deux. Le réseau se focalise cependant bien sur l'abeille pour prendre sa décision, mais il échoue tout de même. Pour la troisième image, en observant la heatmap, on constate que le réseau a regardé les deux abeilles mais semble plutôt s'être concentré sur l'abeille de droite. On pourrait peut être expliquer ces mauvaises classifications à cause d'un manque d'images dans le dataset malgré une volonté d'atténuer le phénomène de déséquilibre des classes à l'aide d'une pondération des classes.

Voici une image d'une abeille *Anthidium Florentinum* :

FIGURE 15 – *Anthidium Florentinum*

On peut voir effectivement de très grandes similitudes entre les deux espèces.
Observons maintenant les mauvaises classifications de l'espèce *Andrena Clarkella* :

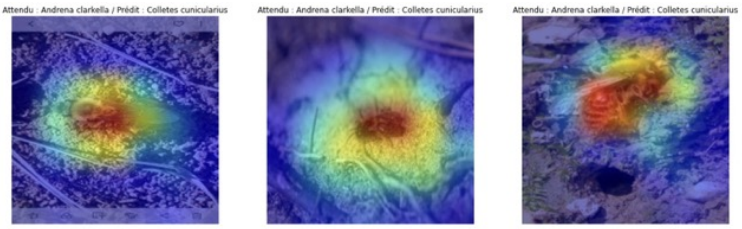

FIGURE 16 – Attendu : *Andrena Clarkella*, Prédiction : *Colletes cunicularius*

FIGURE 17 – Attendu : *Andrena Clarkella*, Prédiction : *Colletes cunicularius*

Les images sont relativement bien cropées et le réseau de neurones a bien identifié l'abeille pour prendre sa décision. Observons donc à quoi ressemble une abeille *Colletes cunicularius* :

FIGURE 18 – *Colletes Cunicularius*

Ce sont encore des abeilles qui se ressemblent beaucoup trop pour que je puisse voir des différences :).

Enfin, on peut essayer de regarder les mauvaises classifications de l'espèce *Bombus lucorum* :

FIGURE 19 – Attendu : *Bombus lucorum*, Prédiction : *Bombus vestalis*

FIGURE 20 – Attendu : *Bombus lucorum*, Prédiction : *Bombus vestalis*

L'image du milieu est peut être trop éloignée pour être bien analysée, cependant, le réseau a bien focalisé son attention sur la zone où se situe l'abeille. Voici à quoi ressemble une abeille *Bombus vestalis* :

FIGURE 21 – *Bombus vestalis*

Maintenant, analysons un peu plus en détail les précisions, sensibilités et f1-score. En observant la précision (0.66) et la sensibilité (0.71), on constate qu'elles sont inférieures aux précisions et sensibilités moyennes (precision : 0.81 (macro) et 0.85 (weighted), sensibilité : 0.78 (macro) et 0.84 (weighted)). Or, le support de cette classe n'est que de 14, ce qui signifie que le training dataset ne comporte qu'un peu plus de 100 images (112 plus précisément). Néanmoins, bien que ce phénomène puisse s'observer avec d'autres classes peu représentées (par exemple l'espèce *Osmia Aurulenta* ou encore *Osmia Caerulescens* n'ont des précisions et des sensibilités respectives que de p = 0.56 et s = 0.56 ainsi que p = 0.6 et s = 0.55 pour des supports respectifs de 9 et 11), il ne s'observe pas pour toutes les classes "sous-représentées". En l'occurrence, on constate pour l'espèce *Andrena Bicolor* (support de 10) une précision parfaite, mais une sensibilité moins bonne (s = 0.6) ce qui signifie que toutes les images détectées comme étant des *Andrena Bicolor* le sont effectivement, mais que l'on en a raté quelques unes. Pour l'espèce *Anthidium septemspinosum* dont le support est de 14, on obtient un F1-score de 0.85 ce qui est supérieur à la macro moyenne. En fait, on peut calculer que 74% des espèces sous-représentées (<= 15 images dans le test dataset) ont un F1-score inférieur au F1-score moyen (macro average). Voici les espèces en question :

	precision	recall	f1-score	support
Andrena bicolor	1.000000	0.600000	0.750000	10.0
Andrena denticulata	1.000000	0.375000	0.545455	8.0
Andrena nycthemera	0.700000	0.636364	0.666667	11.0
Andrena ventralis	0.642857	0.692308	0.666667	13.0
Anthidiellum strigatum	0.666667	0.714286	0.689655	14.0
Anthidium oblongatum	0.562500	0.642857	0.600000	14.0
Anthophora bimaculata	0.500000	0.416667	0.454545	12.0
Anthophora furcata	0.833333	0.555556	0.666667	9.0
Bombus bohemicus	0.733333	0.733333	0.733333	15.0
Bombus muscorum	0.555556	0.555556	0.555556	9.0
Bombus ruderatus	0.250000	0.222222	0.235294	9.0
Chelostoma florisomne	0.625000	0.714286	0.666667	14.0
Halictus rubicundus	0.714286	0.666667	0.689655	15.0
Halictus sexcinctus	0.714286	0.555556	0.625000	9.0
Heriades truncorum	0.666667	0.923077	0.774194	13.0
Lithurgus chrysurus	0.800000	0.363636	0.500000	11.0
Melecta albifrons	0.692308	0.642857	0.666667	14.0
Osmia aurulenta	0.555556	0.555556	0.555556	9.0
Osmia caerulescens	0.600000	0.545455	0.571429	11.0
Rhodanthidium sticticum	0.909091	0.666667	0.769231	15.0

------- Proportion de : 74.07407407407408 % -------

1.9 ResNet50 poids INat2021

1.9.1 Résultats obtenus

categorical_accuracy	0.9023
val_accuracy	0.80950
test_accuracy	

1.10 ResNet101, hierarchical loss

SGD, lr = 1e-3, juste reduce_lr_on_plateau sur val_accuracy loss : hierarchical loss, alpha = 0.5 69 epochs, 300s par epoch

```
AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    VerticalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5),
    RandomBrightnessContrast(p=0.5)
])
```

Augmentation Sequence, cap500, loss début : 0.3985, loss fin : 0.0353

categorical_accuracy	0.9317
val_accuracy	0.86245
test_accuracy	0.8493

2 Modèle pour la reconnaissance des genres d'abeilles

L'objectif est ici de tester un modèle pour reconnaître les espèces d'un même genre, par exemple, reconnaître les espèces du genre Andrena ou encore Bombus.

2.1 ResNet50 modèle genre Andrena, pas de pondération des classes

On commence avec le ResNet50 afin de voir les résultats qu'il est possible d'obtenir.

2.1.1 Résultats obtenus

categorical_accuracy	0.8889
val_accuracy	0.85343
test_accuracy	

On obtient des résultats qui sont déjà bons, néanmoins, on peut essayer de faire mieux en utilisant le ResNet101.

2.2 ResNet101 modèle genre Andrena, pas de pondération des classes

2.2.1 Résultats obtenus

categorical_accuracy	0.9460
val_accuracy	0.90307
test_accuracy	0.89125

Les résultats sont très bons, il serait intéressant d'étudier quelles sont les espèces que le modèle confond et essayer de chercher des explications dans ces confusions.

2.2.2 Analyse des résultats

Tout d'abord, contrairement au modèle reposant sur les 71 espèces développé précédemment, celui-ci n'a confondu qu'au plus 2 fois 2 espèces ensemble, voici les espèces concernées :

```
{'Andrena flavipes': {'Andrena hattorfiana': 2},
 'Andrena nigroaenea': {'Andrena nitida': 2},
 'Andrena thoracica': {'Andrena nitida': 2}}
```

FIGURE 22 – Confusions obtenues, seuil = 2

Observons quelques images mal classifiées, par exemple, les images mal classifiées pour l'espèce *Andrena Nigroaenea* :

Les images semblent être d'assez bonne qualité. Regardons à quoi ressemble une abeille de l'espèce *Andrena Nitida* :

Les espèces se ressemblent énormément au niveau du thorax et de la tête par exemple.

On peut aussi observer les mauvaises classifications de l'espèce *Andrena Flavipes* :

Ici, les images sont déjà de moins bonne qualité, en effet, l'abeille est loin, voire très loin sur l'image 1, par ailleurs, on constate par exemple sur l'image 2 que le modèle a eu du mal à détecter l'abeille, ou en tout cas, les zones où le modèle a concentré son attention sont plutôt à côté de l'abeille que sur l'abeille. Pour ces images, on pourrait alors supposer par exemple que l'abeille était trop loin pour pouvoir inférer correctement l'appartenance à une espèce.

Par ailleurs, voici à quoi ressemble une abeille de l'espèce *Andrena Hattorfiana* :

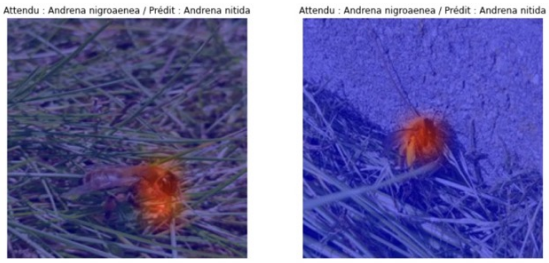

FIGURE 23 – Attendu : *Andrena Nigroaenea*, Prédiction : *Andrena Nitida*

FIGURE 24 – Attendu : *Andrena Nigroaenea*, Prédiction : *Andrena Nitida*

FIGURE 25 – *Andrena Nitida*

FIGURE 26 – Attendu : *Andrena Flavipes*, Prédiction : *Andrena Hattorfiana*

En fait, en regardant les images contenues dans l'ensemble de test pour l'espèce *Andrena Hattorfiana*, j'ai remarqué que les abeilles étaient souvent prises en photo sur une fleur rose. Le modèle a peut être donc appris que pour reconnaître une abeille ce cette espèce, il fallait plutôt observer la fleur sur laquelle se trouve l'abeille, ce qui expliquerait par exemple pourquoi le modèle a attiré son attention sur la fleur plutôt que sur l'abeille dans l'image 2.

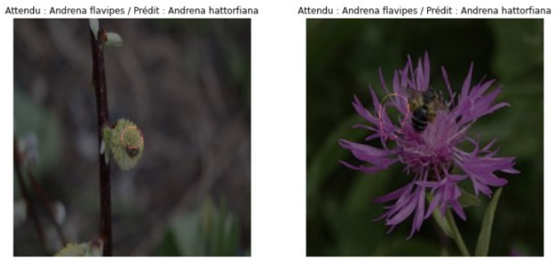

FIGURE 27 – Attendu : *Andrena Flavipes*, Prédiction : *Andrena Hattorfiana*

FIGURE 28 – *Andrena Hattorfiana*

2.3 ResNet101 modèle genre Andrena, pondération des classes

2.4 Résultats obtenus

categorical_accuracy	0.9260
val_accuracy	0.86288
test_accuracy	0.86525

La pondération des classes n'a pas permis d'améliorer les résultats.

2.5 ResNet101 modèle genre Bombus, pas de pondération des classes

2.5.1 Résultats obtenus

categorical_accuracy	0.9341
val_accuracy	0.87942
test_accuracy	0.86486

2.6 ResNet101 modèle genre Bombus, pondération des classes

2.7 Résultats obtenus

categorical_accuracy	0.8675
val_accuracy	0.81081
test_accuracy	0.79002

2.8 WideResNet50-2

```
AUGMENTATIONS_TRAIN = Compose([
    Rotate(limit=[0,100], p=0.5),
    HorizontalFlip(p=0.5),
    VerticalFlip(p=0.5),
    Affine(shear=[-45, 45], p=0.5),
    RandomBrightnessContrast(p=0.5)
])
```

```
model.compile(optimizers.SGD(learning_rate=1e-3, momentum=0.9), loss='
    categorical_crossentropy', metrics=['categorical_accuracy', tf.keras.metrics.
    Precision(), tf.keras.metrics.Recall()])
```

categorical_accuracy	0.6576
val_accuracy	0.64410
test_accuracy	

Arrêt : early stopping au bout de 60 epochs. Une epoch prend environ 210s.

WideResNet50-2 :

categorical_accuracy	0.9452
val_accuracy	0.75491
test_accuracy	