

---

# ImageAI Documentation

*Release 2.0.2*

**"Moses Olafenwa" "John Olafenwa"**

Oct 29, 2018



---

## Contents:

---

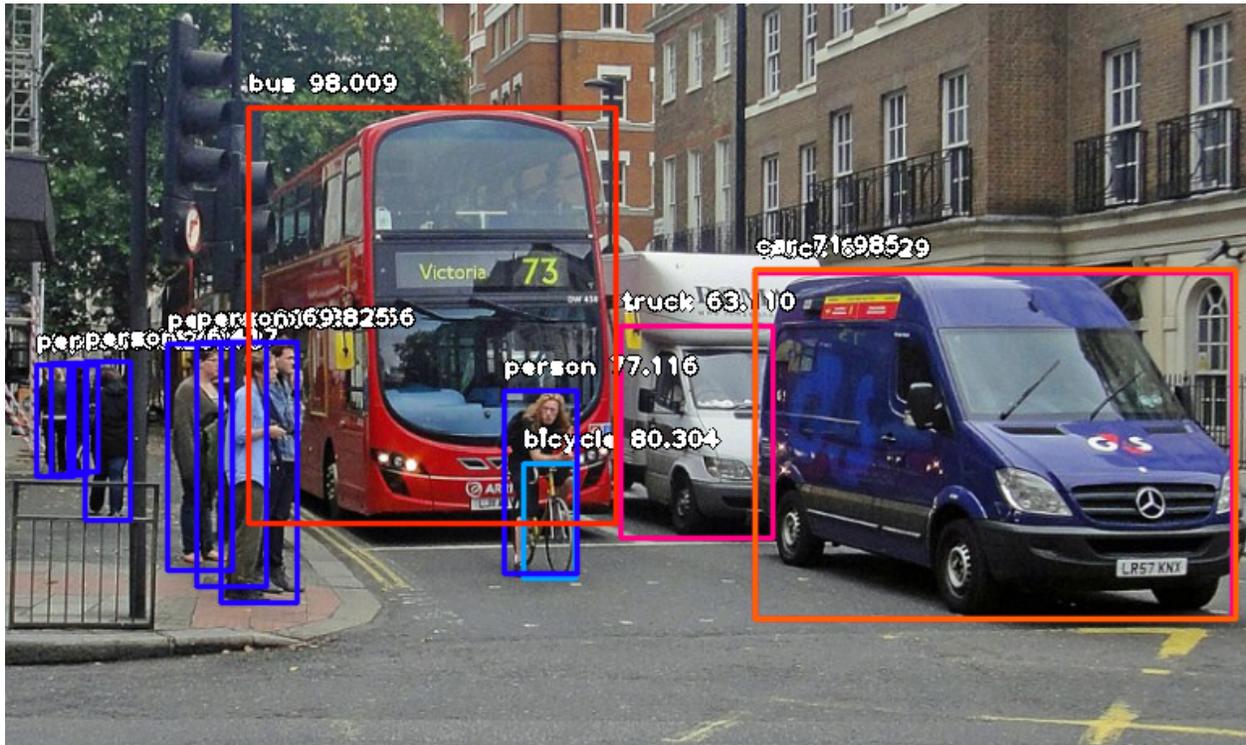
<b>1</b>	<b>Installation de ImageAI</b>	<b>3</b>
1.1	Classes de pr?diction . . . . .	6
1.2	Les Classes de detection . . . . .	9
1.3	Analyse et d?tection sur Vid?o et flux vid?o temp r?el Vid?o . . . . .	13
1.4	Apprentissage personnalis? et pr?diction des Classes . . . . .	22
<b>2</b>	<b>Indices and tables</b>	<b>29</b>



**ImageAI** est une bibliothèque python développée pour permettre aux développeurs, chercheurs, étudiants de construire des applications et des systèmes qui intègrent l'apprentissage profond et la vision assistée par ordinateur en utilisant simplement quelques lignes de code. Cette documentation est fournie pour donner assez de détails sur toutes classes et fonctions disponibles dans **ImageAI**, couplés à un certain nombre d'exemples de code.

**ImageAI** est un projet développé par Moses Olafenwa et John Olafenwa, the AI Commons team.

Le dossier officiel GitHub de **ImageAI** est <https://github.com/OlafenwaMoses/ImageAI>





---

## Installation de ImageAI

---

**ImageAI** nécessite que vous ayez Python 3.5.1 ou supérieur installé ainsi que d'autres bibliothèques python. Avant de procéder à l'installation de **ImageAI** vous devez installer les éléments suivants:

- **Python** 3.5.1 or supérieur, [telecharger Python](#)
- **pip3**, ' telecharger PyPi <<https://pypi.python.org/pypi/pip/>> '\_
- **Tensorflow** 1.4.0 or supérieur

```
pip3 install --upgrade tensorflow
```

- **Numpy** 1.13.1 or supérieur

```
pip3 install numpy
```

- **SciPy** .19.1 or supérieur

```
pip3 install scipy
```

- **OpenCV**

```
pip3 install opencv-python
```

- **Pillow**

```
pip3 install pillow
```

- **Matplotlib**

```
pip3 install matplotlib
```

- **h5py**

```
pip3 install h5py
```

- **Keras**

```
pip3 install keras
```

Une fois que vous avez installé tous ces packages sur votre ordinateur, vous pouvez installer **ImageAI** en utilisant la commande pip ci-dessous. Installation de **ImageAI**

```
pip3 install https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-  
↪2.0.2-py3-none-any.whl
```

Une fois que **ImageAI** est installé, vous pouvez en quelques lignes de code accomplir les tâches de vision assistée par ordinateur les plus puissantes comme vous pouvez le voir ci-dessous. **Reconnaissance d'Image**

*Retrouver tous les codes et la documentation via les liens dans la section en bas de page.*



- convertible : 52.459555864334106
- sports\_car : 37.61284649372101
- pickup : 3.1751200556755066
- car\_wheel : 1.817505806684494
- minivan : 1.7487050965428352

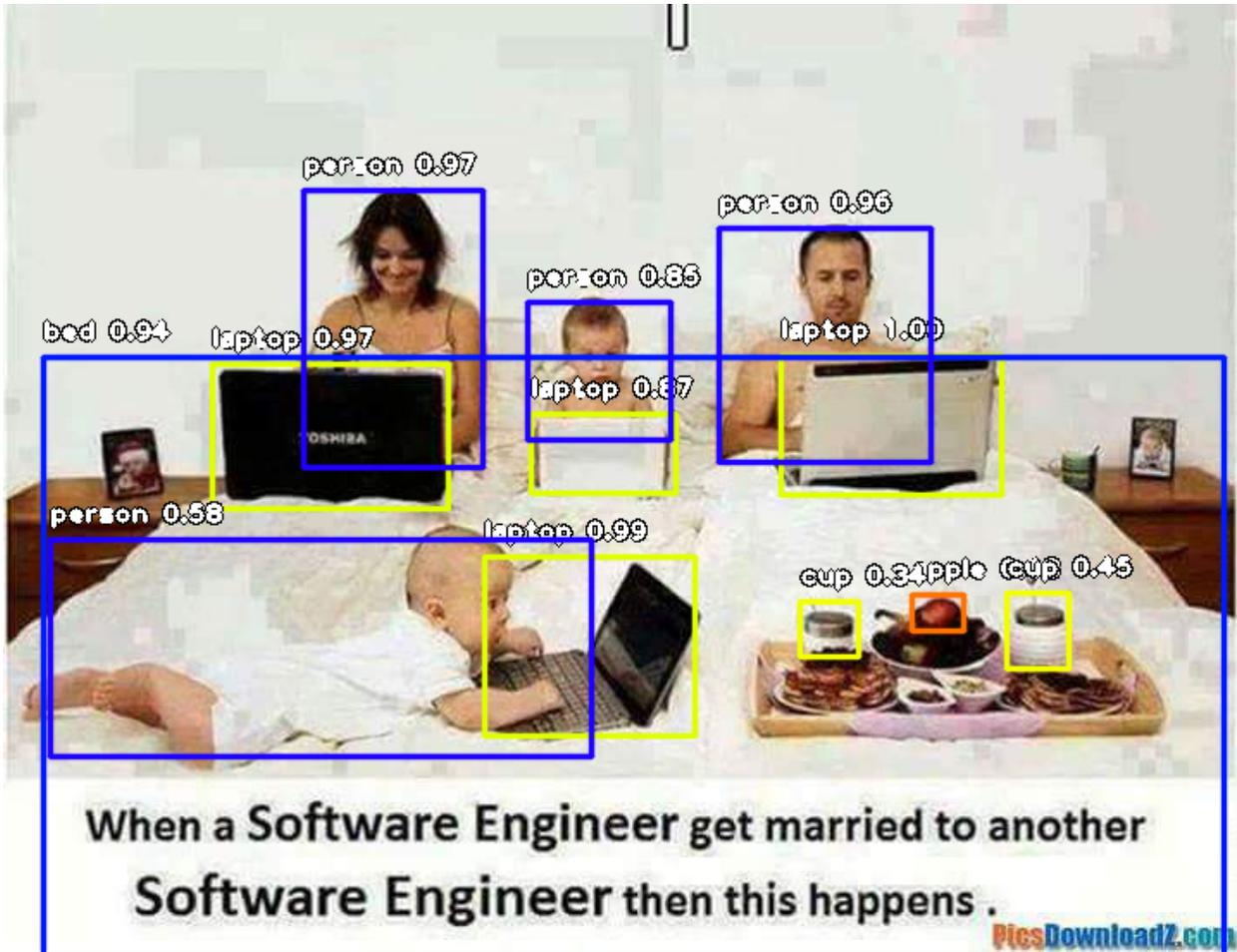
### Détection d'objets sur Image

*Retrouver tous les codes et la documentation via les liens dans la section en bas de page.*

### Détection d'objets sur Vidéo

*Retrouver tous les codes et la documentation via les liens dans la section en bas de page.*

### Analyse de détection Vidéo



Retrouver tous les codes et la documentation via les liens dans la section en bas de page.

### Inférence et entraînement personnalisé pour reconnaissance d'images

Retrouver tous les codes et la documentation via les liens dans la section en bas de page.



Suivez les liens dans la section contenu ci-dessous pour retrouver tous les exemples et la documentation complète des classes et fonctions disponibles.

## 1.1 Classes de prédiction

**ImageAI** fournit un ensemble de classes puissantes et faciles à utiliser pour accomplir les tâches de *reconnaissance sur les images*. Vous pouvez accomplir toutes ces tâches de pointe de vision assistée par ordinateur avec du code python allant de 5 à 12 lignes de code. Une fois que le python est installé, d'autres bibliothèques et **ImageAI** installés dans votre ordinateur, il n'y a aucune limite aux applications incroyables que vous pouvez créer. Trouvez ci-dessous les classes et leur fonction respective rendues disponibles pour votre utilisation. Ces classes peuvent être intégrées dans n'importe quel programme python traditionnel que vous développez, que cela soit un site internet, une application Windows/Linux/macOS ou un système qui supporte ou fait partie d'un réseau local.

===== **imageai.Prediction.ImagePrediction** =====

La classe **ImagePrediction** vous fournit des fonctions pour utiliser les modèles de reconnaissance d'images les plus pointus tel **SqueezeNet**, **ResNet**, **InceptionV3** et **DenseNet** qui ont été **pré-entraînés** sur la base de données

**ImageNet-1000.** Ceci pour dire que vous pouvez utiliser ces classes pour d?tecter et reconnaitre plus de 1000 diff?rents objets sur n?importe quelle image ou ensemble d?images. Pour initialiser la classe dans votre code, vous allez cr?er une instance dans votre code comme suit?:

```
from imageai.Prediction import ImagePrediction
prediction = ImagePrediction()
```

Nous avons fourni les mod?les pr?-entra?nes de reconnaissance d?images des algorithmes suivants **SqueezeNet**, **ResNet**, **InceptionV3** et **DenseNet** que vous allez utiliser dans la classe **ImagePrediction** pour faire la reconnaissance sur les images. Trouvez ci-dessous le lien pour t?!?charger les mod?les. Vous pouvez t?!?charger le mod?le que vous voulez utiliser.

Telechargez le modele SqueezeNet

T?!?chargez le mod?le ResNet

‘ T?!?chargez le mod?le InceptionV3 <<https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0> />‘\_

‘ T?!?chargez le mod?le DenseNet <<https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0> />‘\_

Apr?s avoir cr?? une nouvelle instance de la classe **ImagePrediction**, Vous pouvez utiliser les fonctions ci-dessous pour d?finir les valeurs des propri?t?s et commencer la reconnaissance.

- **.setModelTypeAsSqueezeNet()** , cette fonction ?tablit comme mod?le pour votre instance de reconnaissance d?image que vous avez cr??, le mod?le **SqueezeNet?**; ce qui veut dire que vous accomplirez vos taches de pr?diction en utilisant les mod?les pr?-entra?ns de **SqueezeNet** que vous avez t?!?charg? avec le lien ci-dessus. Trouvez le code ci-dessous?:

```
prediction.setModelTypeAsSqueezeNet ()
```

- **.setModelTypeAsResNet()** , cette fonction ?tablit comme mod?le pour votre instance de reconnaissance d?image que vous avez cr??, le mod?le **ResNet?**; ce qui veut dire que vous accomplirez vos taches de pr?diction en utilisant les mod?les pr?-entra?ns de **ResNet** que vous avez t?!?charg? avec le lien ci-dessus. Trouvez le code ci-dessous?:

```
prediction.setModelTypeAsResNet ()
```

- **.setModelTypeAsInceptionV3()** , cette fonction ?tablit comme mod?le pour votre instance de reconnaissance d?image que vous avez cr??, le mod?le **InceptionV3?**; ce qui veut dire que vous accomplirez vos taches de pr?diction en utilisant les mod?les pr?-entra?ns de **InceptionV3** que vous avez t?!?charg? avec le lien ci-dessus. Trouvez le code ci-dessous?:

```
prediction.setModelTypeAsInceptionV3 ()
```

- **.setModelTypeAsDenseNet()** , cette fonction ?tablit comme mod?le pour votre instance de reconnaissance d?image que vous avez cr??, le mod?le **DenseNet?**; ce qui veut dire que vous accomplirez vos taches de pr?diction en utilisant les mod?les pr?-entra?ns de **DenseNet** que vous avez t?!?charg? avec le lien ci-dessus. Trouvez le code ci-dessous?:

```
prediction.setModelTypeAsDenseNet ()
```

- **.setModelPath()** , cette fonction accepte une chaine de caract?re qui doit ?tre le chemin vers le fichier mod?le que vous avez t?!?charg?.  

```
prediction.setModelPath("resnet50_weights_tf_dim_ordering_tf_kernels.h5")
```

– param?tre **model\_path** (requis) : Il s?agit du chemin vers votre fichier mod?le t?!?charg?.
- **.loadModel()** , Cette fonction charge le mod?le ? partir du chemin que vous avez sp?cifi? dans l?appel de fonction ci-dessus dans votre instance de pr?diction. Trouvez un exemple de code ci-dessous?:

```
prediction.loadModel()
```

– *paramètre prediction\_speed* (optionnel) : Ce paramètre vous permet de réduire jusqu’à 80% le temps qu’il faut pour la tâche de prédiction sur une image, ce qui conduit à une légère réduction de la précision. Ce paramètre accepte les chaînes de caractères. Les valeurs disponibles sont “normal”, “fast”, “faster” et “fastest”. La valeur par défaut est “normal”.

- **.predictImage()**, C’est la fonction qui effectue la tâche de prédiction à proprement parler sur une image. Elle peut être appelée plusieurs fois sur plusieurs images une fois que le modèle a été chargé dans l’instance de prédiction. Trouvez un exemple de code, et paramètres de fonction ci-dessous:

```
predictions, probabilities = prediction.predictImage("image1.jpg", result_
↳count=10)
```

– *paramètre image\_input* (requis) : Il fait référence au chemin vers votre fichier images, tableau Numpy de votre image ou le fichier flux de votre image, indépendamment du type que vous avez choisi.

—*paramètre result\_count* (optionnel) : Il fait référence au nombre possible de prédictions qui doivent être retournées. Le paramètre a une valeur par défaut de 5.

– *paramètre input\_type* (optionnel) : Il fait référence au type de la valeur d’entrée dans le paramètre **image\_input**. Il est par défaut et accepte `array` et `stream` aussi.

—*valeur retournée prediction\_results* (une liste python) : La première valeur renvoyée par la fonction **predictImage** est une liste qui contient tous les résultats possibles de prédiction. Les résultats sont rangés dans l’ordre descendant de probabilité de pourcentage.

– *valeur retournée prediction\_probabilities* (une liste python) :

La seconde valeur renvoyée par la fonction **predictImage** est une liste qui contient les pourcentages de probabilité correspondant à toutes les prédictions possibles dans **prediction\_results**

- **.predictMultipleImages()**, Cette fonction pour accomplir la prédiction sur 2 ou plusieurs images à la fois. Trouvez un exemple de code, et paramètres de fonction ci-dessous:

```
results_array = multiple_prediction.predictMultipleImages(all_images_array,
↳result_count_per_image=5)

for each_result in results_array:
    predictions, percentage_probabilities = each_result["predictions"], each_
↳result["percentage_probabilities"]
    for index in range(len(predictions)):
        print(predictions[index], " : ", percentage_probabilities[index])
    print("-----")
```

– *paramètre sent\_images\_array* (requis) : Il fait référence à une liste qui contient le chemin vers les fichiers images, les tableaux Numpy de vos images ou les fichiers de flux de vos images, indépendamment de type spécifique pour la valeur d’entrée.

– *paramètre result\_count\_per\_image* (optionnel) : Il fait référence au nombre de possible de prédictions renvoyées pour chaque image. Ce paramètre a pour valeur par défaut 2.

– *paramètre input\_type* (optionnel) : Il fait référence au format dans lequel vos images sont représentées dans la liste contenu dans le paramètre **sent\_images\_array**. Il est par défaut `file` et accepte aussi `array` et `stream`.

– *valeur retournée output\_array* (une liste python) : La valeur retournée par la fonction **predictMultipleImages** est une liste qui contient des dictionnaires. Chaque dictionnaire correspond à une image contenue dans le tableau transmis à **sent\_images\_array**. Chaque dictionnaire a une propriété “prediction\_results” qui est la liste de tous les résultats de prédictions sur l’image à cet indice ainsi que la “prediction\_probabilities” qui est la liste correspondant au pourcentage de probabilité de chaque résultat.

## Exemple de code

Trouver ci-dessous un ?chantillon de code pour la pr?diction sur une image?:

```
from imageai.Prediction import ImagePrediction
import os

execution_path = os.getcwd()

prediction = ImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath(os.path.join(execution_path, "resnet50_weights_tf_dim_
↳ordering_tf_kernels.h5"))
prediction.loadModel()

predictions, probabilities = prediction.predictImage(os.path.join(execution_path,
↳"image1.jpg"), result_count=10)
for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , " : " , eachProbability)
```

Trouvez ci-dessous un ?chantillon de code pour la d?tection/pr?diction sur plusieurs images?:

```
from imageai.Prediction import ImagePrediction
import os

execution_path = os.getcwd()

multiple_prediction = ImagePrediction()
multiple_prediction.setModelTypeAsResNet()
multiple_prediction.setModelPath(os.path.join(execution_path, "resnet50_weights_tf_
↳dim_ordering_tf_kernels.h5"))
multiple_prediction.loadModel()

all_images_array = []

all_files = os.listdir(execution_path)
for each_file in all_files:
    if(each_file.endswith(".jpg") or each_file.endswith(".png")):
        all_images_array.append(each_file)

results_array = multiple_prediction.predictMultipleImages(all_images_array, result_
↳count_per_image=5)

for each_result in results_array:
    predictions, percentage_probabilities = each_result["predictions"], each_result[
↳"percentage_probabilities"]
    for index in range(len(predictions)):
        print(predictions[index] , " : " , percentage_probabilities[index])
    print("-----")
```

## 1.2 Les Classes de detection

**ImageAI** fournit un ensemble de classes et fonctions puissantes et faciles ? utiliser pour la **D?tection et Extraction d?objets dans une image**. **ImageAI** vous permet d?utiliser les algorithmes de pointe en apprentissage profond tel que **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Avec **ImageAI** vous pouvez accomplir des taches de d?tection et d?analyse d?images. Trouvez ci-dessous les classes et leurs fonctions respectives mise ? votre disposition pour votre utilisation.

Ces classes peuvent être intégrées dans tout programme Python traditionnel que vous développez ; que ce soit un site internet, une application Windows/Linux/macOS ou un système qui supporte ou fait partir d'un réseau local.

### ===== imageai.Detection.ObjectDetection =====

Cette classe **ObjectDetection** vous fournit les fonctions pour accomplir la détection d'objets sur une image ou un ensemble d'images, utilisant les modèles **pr-entraînés** sur la base de données **COCO**. Les modèles supportés sont **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Ceci veut dire que vous pouvez détecter et reconnaître 80 différents types communs d'objets de tous les jours. Pour commencer, télécharger n'importe quel modèle que vous voulez utiliser via les liens ci-dessous:

Télécharger le modèle RetinaNet - resnet50\_coco\_best\_v2.0.1.h5

Télécharger le modèle YOLOv3 - yolo.h5

Télécharger le modèle TinyYOLOv3 - yolo-tiny.h5

Une fois que vous avez téléchargé le modèle de votre choix, vous devez créer une instance de la classe **ObjectDetection** comme dans l'exemple ci-dessous:

```
from imageai.Detection import ObjectDetection

detector = ObjectDetection()
```

Une fois que vous avez créé une instance de la classe, vous pouvez utiliser les fonctions ci-dessous pour choisir convenablement les propriétés d'instance et de commencer la détection d'objets sur les images.

- **.setModelTypeAsRetinaNet()** , cette fonction agit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle **RetinaNet**, ceci veut dire que vous accomplirez la tâche de détection d'objets à l'aide de modèle pré-entraîné de **RetinaNet** que vous avez téléchargé par les liens ci-dessus. Trouvez un exemple de code ci-dessous:

```
detector.setModelTypeAsRetinaNet()
```

- **.setModelTypeAsYOLOv3()** , cette fonction agit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle **YOLOv3**, ceci veut dire que vous accomplirez la tâche de détection d'objets à l'aide de modèle pré-entraîné de **YOLOv3** que vous avez téléchargé par les liens ci-dessus. Trouvez un exemple de code ci-dessous:

```
detector.setModelTypeAsYOLOv3()
```

- **.setModelTypeAsTinyYOLOv3()** , cette fonction agit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle **TinyYOLOv3**, ceci veut dire que vous accomplirez la tâche de détection d'objets à l'aide de modèle pré-entraîné de **TinyYOLOv3** que vous avez téléchargé par les liens ci-dessus. Trouvez un exemple de code ci-dessous:

```
detector.setModelTypeAsTinyYOLOv3()
```

- **.setModelPath()** , Cette fonction prend en argument une chaîne de caractères qui doit être le chemin vers le fichier modèle que vous avez téléchargé et doit correspondre au type de modèle choisi pour votre instance de détection d'objets. Trouvez un exemple de code et de paramètres de fonction ci-dessous :

```
detector.setModelPath("yolo.h5")
```

– paramètre **model\_path** (requis) : c'est le chemin vers votre modèle téléchargé.

- **.loadModel()** , Cette fonction charge le modèle à partir du chemin que vous avez spécifié dans l'appel de fonction ci-dessus de votre instance de détection d'objets. Trouvez un exemple de code ci-dessous:

```
detector.loadModel()
```

– *param?tre* **detection\_speed** (optionnel) : Ce param?tre vous permet de r?duire jusqu 80% le temps qu’il faut pour d?tecter les objets sur une image ce qui conduit ? une l?g?re r?duction de la pr?cision. Ce param?tre accepte des valeurs de type chaines de caract?res. Les valeurs disponibles sont **normal**, **fast**, **faster**, **fastest** et **flash**. La valeur par d?faut est **normal**

- **.detectObjectsFromImage()** ,C’est la fonction qui accomplit la d?tection d’objets apr?s que le mod?le ait ?t? charg?. Elle peut ?tre appel?e plusieurs fois pour d?tecter les objets dans plusieurs images. Trouvez un exemple de code ci-dessous?

```
detections = detector.detectObjectsFromImage(input_image="image.jpg", output_
↪image_path="imagenew.jpg", minimum_percentage_probability=30)
```

– *param?tre* **input\_image** (requis) : Il fait r?f?rence au chemin vers le fichier image sur lequel vous voulez faire la d?tection. Ce param?tre peut ?tre le tableau **Numpy** ou le fichier flux de l’image si vous donner la valeur “array” ou “stream” au param?tre **input\_type**.

—*param?tre* **output\_image\_path** (requis seulement si **input\_type** = “file”) : Il fait r?f?rence au chemin vers le lieu de sauvegarde de l’image d?tect?e ou d?tection. Il n’est requis que si **input\_type** = “file”.

– *param?tre* **minimum\_percentage\_probability** (optionnel) : Ce param?tre est utilis? pour d?terminer l’int?grit? des r?sultats de d?tections. R?duire cette valeur permettra de d?tecter plus d’objets alors que l’augmenter permet d’avoir des objets d?tect?s avec la plus grande pr?cision. La valeur par d?faut est 50.

—*param?tre* **output\_type** (optionnel) : ce param?tre permet de d?finir le format dans lequel l’image de d?tections sera produit. Les valeurs disponibles sont ?file?(fichier) et ?array?(tableau). La valeur par d?faut est ?file?. Si ce param?tre est d?fini comme ?array?, la fonction va renvoyer un tableau Numpy pour l’image de d?tection. Retrouvez un exemple ci-dessous?:

```
returned_image, detections = detector.detectObjectsFromImage(input_image="image.jpg",
output_type="array", minimum_percentage_probability=30)
```

—*param?tre* **display\_percentage\_probability** (optionnel) : Ce param?tre peut ?tre utilis? pour cacher le pourcentage de probabilit? de chaque objet d?tect? dans l’image de d?tect?e si sa valeur est d?finie ? ?False?. La valeur par d?faut est ?True?.

– *param?tre* **display\_object\_name** (optionnel) : Ce param?tre peut ?tre utilis? pour cacher le nom de chaque objet d?tect? dans l’image de d?tection si sa valeur est d?finie comme ?False?. La valeur par d?faut est ?True?.

—*param?tre* **extract\_detected\_objects** (optionnel) : ce param?tre peut ?tre utilis? pour extraire et sauvegarder/ retourner chaque objet d?tect? dans une image dans une image s?par?e. Sa valeur par d?faut est ?False?.

– *valeurs renvoy?es* : Les valeurs renvoy?es vont d?pendre des param?tres envoy?s dans la fonction **detectObjectsFromImage()**. Retrouvez les commentaires et le code ci-dessous?:

```
.....
```

Si tous les param?tres n?cessaires sont d?finis et ‘output\_image\_path’ est d?fini vers le chemin o? le fichier de d?tection sera sauvegard?, la fonction va renvoyer?: 1. un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets d?tect?s dans l’image. Chaque dictionnaire a les propri?t?s suivantes?:

**\*Nom (chaine de caract?res – string)**

- **percentage\_probability (float)**

– **box\_points** (tuple de coordonn?es x1,y1,x2 et y2)

```
.....
```

```
detections = detector.detectObjectsFromImage(input_image="image.jpg",
output_image_path="imagenew.jpg", minimum_percentage_probability=30)
"""
```

Si tous les paramètres requis sont définis et `output_type = 'array'`, la fonction va renvoyer 1. Un tableau Numpy de l'image de détection. 2. Un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets détectés dans l'image. Chaque dictionnaire contient les propriétés suivantes: \* Nom (string ? chaîne de caractères) \* `percentage_probability` (float) \* `box_points` (tuple de coordonnées `x1,y1,x2` et `y2`)

```
""" returned_image, detections = detector.detectObjectsFromImage(input_image="image.jpg",
output_type="array", minimum_percentage_probability=30)
"""
```

Si `extract_detected_objects = True` et `'output_image_path'` est défini par le chemin vers le lieu de sauvegarde de l'image de détection, la fonction renvoie:

1. Un tableau de dictionnaires, chaque dictionnaire correspond aux objets détectés dans l'image. Chaque dictionnaire contient les propriétés suivantes: \* Nom (string) \* `percentage_probability` (float) \* `box_points` (tuple de coordonnées `x1,y1,x2` et `y2`) 2. Un tableau de chaînes de caractères représentant les chemins des images de chaque objet extrait de l'image de départ.

```
""" detections, extracted_objects = detector.detectObjectsFromImage(input_image="image.jpg",
output_image_path="imagenew.jpg", extract_detected_objects=True,
minimum_percentage_probability=30)
"""
```

Si `extract_detected_objects = True` et `output_type = 'array'`, la fonction va renvoyer:

1. Un tableau Numpy de l'image de détection.
  2. Un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets détectés dans l'image. Chaque dictionnaire contient les propriétés suivantes: \* `nom`(string) \* `percentage_probability` (float) \* `box_points` (tuple de coordonnées `x1,y1,x2` et `y2`)
  3. Un tableau de tableaux Numpy de chaque objet détecté dans l'image
- ```
""" returned_image, detections,
extracted_objects = detector.detectObjectsFromImage(input_image="image.jpg", output_type="array",
extract_detected_objects=True, minimum_percentage_probability=30)
"""
```

- **.CustomObjects()** C'est la fonction que vous utiliserez lorsque vous ne voulez faire la détection que d'un nombre limité d'objets. Elle renvoie un dictionnaire d'objets et leur valeur `True` ou `False`. Pour détecter les objets sélectionnés dans une image, vous devrez utiliser les dictionnaires renvoyés par cette fonction avec la fonction `** detectCustomObjectsFromImage(**)`. Retrouvez les détails dans les commentaires et le code ci-dessous:

```
"""
```

Il y a 80 possible objets que vous pouvez détecter avec la classe `ObjectDetection`, vous pouvez les voir ci-dessous.

Person(personne), bicycle(vélo), car(voiture), motorcycle(moto), airplane(avion), Bus(bus), train(train), truck(camion), boat(bateau), traffic light(feux de signalisation), fire hydrant (bouche d'incendie), stop sign (panneau stop), parking meter (parc mètre), bench (banc), bird (oiseau), cat (chat), dog (chien), horse (cheval), sheep (brebis), cow (vache), elephant (éléphant), bear (ours), zebra (zebre), giraffe (girafe), backpack (sac à dos), umbrella (parapluie), handbag (sac à main), tie (cravate), suitcase (valise), frisbee (frisbee), skis (skis), snowboard (snowboard), sports ball(balle de sport), kite (cerf - volant), baseball bat (batte de baseball), baseball glove (gant de baseball), skateboard (skateboard), surfboard (planche de surf), tennis racket (raquette de tennis), bottle (bouteille), wine glass (verre de vin), cup (gobelet), fork (fourchette), knife (couteau), spoon (cuillère), bowl (bol), banana (banane), apple (pomme), sandwich (sandwich), orange (orange), broccoli (brocoli), carrot (carotte), hot dog (hot dog), pizza (pizza), donut (beignet), cake(gâteau), chair (chaise), couch(canapé), potted plant(plante à pot), bed(lit), dining table(table de

diner), toilet(toilette), tv(t?l?vision), laptop(ordinateur), mouse(souris), remote(t?l?commande), keyboard(clavier), cell phone(t?l?phone portable), microwave(micro-onde), oven (four), toaster(grille pain), sink(?vier), refrigerator (r?frig?rateur), book (cahier), clock (horloge), vase(vase) , scissors(ciseaux), teddy bear(ours en peluche), hair dryer(s?che cheveux), toothbrush(brosse ? dent).

Pour d?tecter uniquement certains des objets ci-dessus, vous devrez instancier la fonction CustomObjects et d?finir le ou les noms des objets que vous voulez d?tecter. Le reste sera d?fini ? ?False? par d?faut. Dans l'exemple ci-dessous, nous d?tectons uniquement ?person?(personne) et ?dog?(chien).

```
custom = detector.CustomObjects(person=True, dog=True)
```

- **.detectCustomObjectsFromImage()**, Cette fonction a tous les param?tres et renvoie toutes les valeurs de la fonction **\*\* detectObjectsFromImage()\*\***, avec une petite diff?rence. Cette fonction ne fait la d?tection sur une image que d?objets s?lectionn?s. Contrairement ? la fonction **\*\* detectObjectsFromImage()\*\***, elle a besoin d'un param?tre suppl?mentaire qui est **custom\_objet** qui lui r?cup?re le dictionnaire renvoy? par la fonction **\*\* CustomObjects()\*\***. Dans l'exemple ci-dessous, nous avons d?fini la fonction de d?tection pour qu'elle ne reconnaisse que ?les personnes et les chiens?:

```
custom = detector.CustomObjects(person=True, dog=True)

detections = detector.detectCustomObjectsFromImage( custom_objects=custom, input_
↪ image=os.path.join(execution_path , "image3.jpg"), output_image_path=os.path.
↪ join(execution_path , "image3new-custom.jpg"), minimum_percentage_
↪ probability=30)
```

**\*\* Exemple de code pour la d?tection d'objets sur Image \*\***

Trouvez ci-dessous un exemple de code pour d?tecter les objets sur une image?:

```
from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath( os.path.join(execution_path , "yolo.h5"))
detector.loadModel()
detections = detector.detectObjectsFromImage(input_image=os.path.join(execution_path ,
↪ "image.jpg"), output_image_path=os.path.join(execution_path , "imagenew.jpg"),
↪ minimum_percentage_probability=30)

for eachObject in detections:
    print(eachObject["name"] , " : ", eachObject["percentage_probability"], " : ",
↪ eachObject["box_points"] )
    print("-----")
```

a.. **ImageAI documentation master file, created by sphinx-quickstart** on Tue Jun 12 06:13:09 2018. You can adapt this file completely to your liking, but it should at least contain the root *toctree* directive.

## 1.3 Analyse et d?tection sur Vid?o et flux vid?o temp r?el Vid?o

ImageAI fournit un ensemble de classes et de fonctions puissantes et facile ? utiliser pour faire de la **D?tection et du tracking d'objets dans une Vid?o** et L?\*"analyse vid?o"\*. ImageAI vous permet d'utiliser ou d'employer tous les algorithmes de pointes d'apprentissage profond tel que **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Avec ImageAI vous pouvez effectuer des taches de d?tection et analyser des vid?os et des flux vid?o temps r?el ? partir des cameras IP et de celle de vos appareils. Trouvez ci-dessous les diff?rentes classes et les fonctions respectives mise ? votre

disposition pour votre utilisation. Ces classes peuvent être intégrées à tout programme python traditionnel que vous développez, que cela soit un site internet, une application Windows/Linux/macOS ou un système qui supporte ou fait partie d'un réseau local.

#### ===== imageai.Detection.VideoObjectDetection =====

La classe **VideoObjectDetection** vous fournit des fonctions pour détecter les objets dans une vidéo ou un flux vidéo provenant d'une caméra ou d'une caméra IP en utilisant les modèles **pr-entraînés** à partir de la base de données **COCO**. Les modèles supportés sont **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Ceci veut dire que vous pouvez détecter et reconnaître 80 différents types d'objets de la vie de tous les jours dans les vidéos. Pour commencer, téléchargez un des modèles **pr-entraînés** que vous voulez utiliser via les liens ci-dessous.

Télécharger le modèle **RetinaNet - resnet50\_coco\_best\_v2.0.1.h5**

‘ Télécharger le modèle **YOLOv3 - yolo.h5** <<https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0> />‘ \_

‘ Télécharger le modèle **TinyYOLOv3 - yolo-tiny.h5** <<https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0> />‘ \_

Une fois que vous avez téléchargé le modèle que vous choisissez d'utiliser, créez une instance de **VideoObjectDetection** comme vous pouvez le voir ci-dessous:

```
from imageai.Detection import VideoObjectDetection

detector = VideoObjectDetection()
```

Une fois que vous avez créé une instance de la classe, vous pouvez appeler les fonctions ci-dessous pour paramétrer ses propriétés et détecter les objets dans une vidéo.

- **.setModelTypeAsRetinaNet()**, cette fonction établit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle

**RetinaNet**, ce qui veut dire que vous accomplirez votre tâche de détection d'objets en utilisant le modèle **pr-entraîné RetinaNet** que vous avez précédemment téléchargé par le lien ci-dessus. Trouvez ci-dessous un exemple de code:

```
detector.setModelTypeAsRetinaNet()
```

- **.setModelTypeAsYOLOv3()**, cette fonction établit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle

**YOLOv3**, ce qui veut dire que vous accomplirez votre tâche de détection d'objets en utilisant le modèle **pr-entraîné YOLOv3** que vous avez précédemment téléchargé par le lien ci-dessus. Trouvez ci-dessous un exemple de code:

```
detector.setModelTypeAsYOLOv3()
```

- **.setModelTypeAsTinyYOLOv3()**, cette fonction établit comme type de modèle pour l'instance de détection d'objets que vous avez créé le modèle

**TinyYOLOv3**, ce qui veut dire que vous accomplirez votre tâche de détection d'objets en utilisant le modèle **pr-entraîné TinyYOLOv3** que vous avez précédemment téléchargé par le lien ci-dessus. Trouvez ci-dessous un exemple de code:

```
detector.setModelTypeAsTinyYOLOv3()
```

- **.setModelPath()**, cette fonction accepte une chaîne de caractères qui doit être le chemin vers le fichier modèle que vous avez téléchargé et doit correspondre au type de modèle choisi pour votre instance de détection d'objets. Trouver un exemple de code et de paramètres de la fonction ci-dessous:

```
detector.setModelPath("yolo.h5")
```

– paramètre **model\_path** (requis) : C'est le chemin vers le fichier modèle téléchargé.

- **.loadModel()** , Cette fonction charge le modèle ? partir du chemin que vous avez sp?cifi? dans l'appel ci-dessus de fonction de votre instance de d?tection d'objets. Trouver un exemple de code ci-dessous:

```
detector.loadModel()
```

– *param?tre* **detection\_speed** (optionnel) : Ce param?tre vous permet de r?duire de 80% le temps qu'il faut pour d?tecter les objets sur une vid?o, ce qui conduira ? une l?g?re baisse de la pr?cision. Ce param?tre accepte des valeurs de type chaine de caract?res. Les valeurs disponibles sont “normal”, “fast”, “faster”, “fastest” et “flash”. La valeur par d?faut est??normal?.

- **.detectObjectsFromVideo()** , Il s'agit de la fonction qui accomplit la d?tection d'objets sur un fichier vid?o ou un flux vid?o direct apr?s que le mod?le ait ?t? charg? dans l'instance que vous avez cr??. Retrouvez ci-apr?s un exemple de code complet?:

```
from imageai.Detection import VideoObjectDetection
import os

execution_path = os.getcwd()

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath( os.path.join(execution_path , "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(input_file_path=os.path.
→join(execution_path, "traffic.mp4"),
output_file_path=os.path.join(execution_path,
→"traffic_detected")
, frames_per_second=20, log_progress=True)

print(video_path)
```

– *parametre* **input\_file\_path** (requis si vous ne tenez pas compte de **camera\_input**) : Il fait r?f?rence au chemin vers le fichier vid?o sur lequel vous voulez faire la d?tection.

—*param?tre* **output\_file\_path** (requis si **save\_detected\_video** n'a pas la valeur False) : Il fait r?f?rence vers l'emplacement de l'enregistrement du fichier vid?o d?tect?. Par d?faut, cette fonction enregistre les vid?os dans le format **.avi**.

– *param?tre* **frames\_per\_second** (optionnel, mais recommand?) :

Ce param?tre vous permet de d?finir le nombre de frames par seconde pour la vid?o d?tect?e qui sera enregistr?e. Sa valeur par d?faut est 20 mais nous recommandons que vous d?finissiez la valeur qui sied pour votre vid?o ou vid?o-direct.

– *param?tre* **log\_progress** (optionnel) : D?finir ce param?tre a ?True? affiche le progr?s de la vid?o ou flux direct pendant qu'il est d?tecter dans la console. Il fera un rapport sur chaque frame d?tect? pendant qu'il progresse. La valeur par d?faut est ?False?

—*param?tre* **return\_detected\_frame** (optionnel) :

Ce param?tre vous permet de retourner le frame d?tect? comme tableau Numpy pour chaque frame, seconde et minute de la vid?o d?tect?e. Le tableau Numpy retourne sera envoy? respectivement a **per\_frame\_function**, **per\_second\_function** et **per\_minute\_function** (d?tails ci-dessous)

– *param?tre* **camera\_input** (optionnel) : Ce param?tre peut ?tre assigne en remplacement de **input\_file\_path** si vous voulez d?tecter des objets dans le flux vid?o de la cam?ra. Vous devez instancier la fonction **VideoCapture()** d' OpenCV et de charger l'objet dans ce param?tre.

Ci-dessous un exemple complet de code?:

```

from imageai.Detection import VideoObjectDetection
import os
import cv2

execution_path = os.getcwd()

camera = cv2.VideoCapture(0)

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path , "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(camera_input=camera,
      output_file_path=os.path.join(execution_path, "camera_detected_
↪video")
      , frames_per_second=20, log_progress=True, minimum_percentage_
↪probability=30)

print(video_path)

```

- param?tre* **minimum\_percentage\_probability** (optionnel) : Ce param?tre est utilis? pour d?terminer l'intensit? des r?sultats de d?tection. Diminuer la valeur permet d'afficher plus d'objets alors que l'accroitre permet de s'assurer que les objets d?tect?s ont la pr?cision la plus ?lev?e. La valeur par d?faut est 50.
- *param?tre* **display\_percentage\_probability** (optionnel) : Ce param?tre peut ?tre utilis? pour cacher le pourcentage de probabilit? pour chaque objet d?tect? dans la vid?o d?tect? si sa valeur est ?False?. La valeur par d?faut est ?True?.
- param?tre* **display\_object\_name** (optionnel) : Ce param?tre peut ?tre utilis? pour cacher le nom de chaque objet d?tection dans la vid?o s'il est d?fini ? False. La valeur par d?faut est True.
- *param?tre* **save\_detected\_video** (optionnel) : Ce param?tre peut ?tre utilis? pour sauvegarder ou non la vid?o de d?tection. Sa valeur par d?faut est True.
- param?tre* **per\_frame\_function** (optionnel) : Ce param?tre permet de transmettre le nom d'une fonction que vous d?finissez. Puis, pour chaque frame de la vid?o qui est d?tect?e, la fonction sera d?finie dans le param?tre qui sera ex?cut? et la donn?e analytique de la vid?o sera transmise ? la fonction. Les donn?es renvoy?es peuvent ?tre visualis?es ou enregistr?es dans une base de donn?es NoSQL pour une utilisation et visualisation ult?rieure.

Ci-dessous un exemple complet de code?:

```

"""

```

Ce param?tre vous permet de d?finir dans une fonction ou vous voulez faire une ex?cution chaque fois qu'une image de vid?o est d?tect?e. Si ce param?tre est d?fini par une fonction, apr?s qu'une image de la vid?o soit d?tect?e, la fonction sera ex?cut?e avec les valeurs suivantes en entr?e? : \* Num?ro de position de la frame de la vid?o. \* Un tableau de dictionnaires, avec chaque dictionnaire correspondant ? chaque objet d?tect?. Chaque dictionnaire contient? : 'name', 'percentage\_probability' et 'box\_points' \* Un dictionnaire avec pour clefs le nom de chaque unique objet et le nombre d'instance de chaque objet pr?sent. \* Si return\_detected\_frame est d?fini a ?True?, le tableau Numpy de la frame d?tect?e sera transmis comme quatri?me valeur dans la fonction.

```

"""

```

```

from imageai.Detection import VideoObjectDetection import os

def forFrame(frame_number, output_array, output_count): print("FOR FRAME " , frame_number)

```

```

print("Output for each object : ", output_array) print("Output count for unique objects : ", output_count)
print("-----END OF A FRAME -----")

video_detector = VideoObjectDetection() video_detector.setModelTypeAsYOLOv3()
video_detector.setModelPath(os.path.join(execution_path, "yolo.h5")) video_detector.loadModel()

video_detector.detectObjectsFromVideo(input_file_path=os.path.join(execution_path, "traffic.mp4"),
output_file_path=os.path.join(execution_path, "video_frame_analysis") , frames_per_second=20,
per_frame_function=forFrame, minimum_percentage_probability=30)

```

Dans l'exemple ci-dessus, chaque image ou frame de la vid?o est trait?e et d?tect?e, la fonction va recevoir et renvoyer les donn?es analytiques pour chaque objet d?tect? dans la frame de la vid?o comme vous pouvez le voir ci-dessous?:

```

Output for each object : [{'box_points': (362, 295, 443, 355), 'name': 'boat',
↳ 'percentage_probability': 26.666194200515747}, {'box_points': (319, 245, 386, 296),
↳ 'name': 'boat', 'percentage_probability': 30.052968859672546}, {'box_points': (219,
↳ 308, 341, 358), 'name': 'boat', 'percentage_probability': 47.46982455253601}, {'box_
↳ points': (589, 198, 621, 241), 'name': 'bus', 'percentage_probability': 24.
↳ 62330162525177}, {'box_points': (519, 181, 583, 263), 'name': 'bus', 'percentage_
↳ probability': 27.446213364601135}, {'box_points': (493, 197, 561, 272), 'name': 'bus
↳ ', 'percentage_probability': 59.81815457344055}, {'box_points': (432, 187, 491,
↳ 240), 'name': 'bus', 'percentage_probability': 64.42965269088745}, {'box_points':
↳ (157, 225, 220, 255), 'name': 'car', 'percentage_probability': 21.150341629981995},
↳ {'box_points': (324, 249, 377, 293), 'name': 'car', 'percentage_probability': 24.
↳ 089913070201874}, {'box_points': (152, 275, 260, 327), 'name': 'car', 'percentage_
↳ probability': 30.341443419456482}, {'box_points': (433, 198, 485, 244), 'name': 'car
↳ ', 'percentage_probability': 37.205660343170166}, {'box_points': (184, 226, 233,
↳ 260), 'name': 'car', 'percentage_probability': 38.52525353431702}, {'box_points':
↳ (3, 296, 134, 359), 'name': 'car', 'percentage_probability': 47.80363142490387}, {
↳ 'box_points': (357, 302, 439, 359), 'name': 'car', 'percentage_probability': 47.
↳ 94844686985016}, {'box_points': (481, 266, 546, 314), 'name': 'car', 'percentage_
↳ probability': 65.8585786819458}, {'box_points': (597, 269, 624, 318), 'name':
↳ 'person', 'percentage_probability': 27.125394344329834}]

```

```

Output count for unique objects : {'bus': 4, 'boat': 3, 'person': 1, 'car': 8}

```

```

-----END OF A FRAME -----

```

Ci-dessous est le code complet qui a une fonction qui r?cup?re les donn?es analytiques et les visualise?; ainsi que la frame d?tect?e en temps r?el pendant que la vid?o est trait?e et analys?e?:

```

from imageai.Detection import VideoObjectDetection
import os
from matplotlib import pyplot as plt

execution_path = os.getcwd()

color_index = {'bus': 'red', 'handbag': 'steelblue', 'giraffe': 'orange',
↳ 'spoon': 'gray', 'cup': 'yellow', 'chair': 'green', 'elephant': 'pink', 'truck':
↳ 'indigo', 'motorcycle': 'azure', 'refrigerator': 'gold', 'keyboard': 'violet', 'cow
↳ ': 'magenta', 'mouse': 'crimson', 'sports ball': 'raspberry', 'horse': 'maroon',
↳ 'cat': 'orchid', 'boat': 'slateblue', 'hot dog': 'navy', 'apple': 'cobalt',
↳ 'parking meter': 'aliceblue', 'sandwich': 'skyblue', 'skis': 'deepskyblue',
↳ 'microwave': 'peacock', 'knife': 'cadetblue', 'baseball bat': 'cyan', 'oven':
↳ 'lightcyan', 'carrot': 'coldgrey', 'scissors': 'seagreen', 'sheep': 'deepgreen',
↳ 'toothbrush': 'cobaltgreen', 'fire hydrant': 'limegreen', 'remote': 'forestgreen',
↳ 'bicycle': 'olivedrab', 'toilet': 'ivory', 'tv': 'khaki', 'skateboard':
↳ 'palegoldenrod', 'train': 'cornsilk', 'zebra': 'wheat', 'tie': 'burlywood', 'orange
↳ ': 'melon', 'bird': 'bisque', 'dining table': 'chocolate', 'hair drier (containing text)
↳ ', 'cell phone': 'sienna', 'sink': 'coral', 'bench': 'salmon', 'bottle': 'brown',
↳ 'car': 'silver', 'bowl': 'maroon', 'tennis racket': 'palevioletred', 'airplane':
↳ 'lavenderblush', 'pizza': 'hotpink', 'umbrella': 'deeppink', 'bear': 'plum', 'fork
↳ ': 'purple', 'laptop': 'indigo', 'vase': 'mediumpurple', 'baseball glove':
↳ 'slateblue', 'traffic light': 'mediumblue', 'bed': 'navy', 'broccoli': 'royalblue',
↳ 'backpack': 'slategray', 'snowboard': 'skyblue', 'kite': 'cadetblue', 'teddy bear':

```

(continued from previous page)

```

resized = False

def forFrame(frame_number, output_array, output_count, returned_frame):

    plt.clf()

    this_colors = []
    labels = []
    sizes = []

    counter = 0

    for eachItem in output_count:
        counter += 1
        labels.append(eachItem + " = " + str(output_count[eachItem]))
        sizes.append(output_count[eachItem])
        this_colors.append(color_index[eachItem])

    global resized

    if (resized == False):
        manager = plt.get_current_fig_manager()
        manager.resize(width=1000, height=500)
        resized = True

    plt.subplot(1, 2, 1)
    plt.title("Frame : " + str(frame_number))
    plt.axis("off")
    plt.imshow(returned_frame, interpolation="none")

    plt.subplot(1, 2, 2)
    plt.title("Analysis: " + str(frame_number))
    plt.pie(sizes, labels=labels, colors=this_colors, shadow=True,
↳startangle=140, autopct="%1.1f%%")

    plt.pause(0.01)

    video_detector = VideoObjectDetection()
    video_detector.setModelTypeAsYOLOv3()
    video_detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
    video_detector.loadModel()

    plt.show()

    video_detector.detectObjectsFromVideo(input_file_path=os.path.join(execution_
↳path, "traffic.mp4"), output_file_path=os.path.join(execution_path, "video_frame_
↳analysis"), frames_per_second=20, per_frame_function=forFrame, minimum_
↳percentage_probability=30, return_detected_frame=True)

-- *param?tre* **per_second_function** (optionnel) : Ce param?tre vous permet de
↳transmettre le nom d'une fonction que vous d?finissez. Puis, pour chaque seconde de
↳la vid?o qui est d?tect?e, la fonction sera transmise dans le param?tre sera ex?cut?
↳e et les donn?es analytiques de la vid?o seront envoy?es dans la fonction. Les donn?
↳es renvoy?es peuvent ?tre visualis?es ou sauvegard?es dans une base de donn?es
↳NoSQL pour une utilisation et visualisation future.

```

(continued from previous page)

```

Ci-dessous un exemple complet de code::

"""
    Ce parametre vous permet de transmettre dans une fonction o? vous voulez
    faire une ex?cution apr?s que chaque seconde de la vid?o soit d?tect?e. Si le param?
    tre est d?fini comme une fonction, seconde apr?s que le vid?o soit d?tect?, la
    fonction sera ex?cut?e avec les valeurs suivantes en argument?:

    -- num?ro de position du second
    -- un tableau de dictionnaire dont les clefs sont les num?ros de position de
    chaque frame pr?sente ? la derni?re seconde, et la valeur de chaque clef est le
    tableau de chaque frame qui contient les dictionnaires de chaque objet d?tect? dans
    la frame.

    -- Un tableau de dictionnaires, avec chaque dictionnaire ? chaque frame de la
    seconde pr?c?dente, et les cl?s pour chaque dictionnaire sont les noms de num?ros d?
    objets uniques d?tect?s dans chaque frame, et les cl?s sont le nombre d?instances
    des objets trouv?s dans le frame.

    -- Un dictionnaire avec pour cl? le nom de chaque unique objet d?tect? dans
    toutes les secondes pass?es, at les valeurs cl?s sont les moyennes d?instance d?
    objets trouv?s dans toutes les frames contenus dans les secondes pass?es.

    -- Si return_detected_frame est d?fini aa ?True?, le tableau Numpy du frame de
    d?tection sera envoy? comme cinqui?me param?tre dans la fonction.
"""

from imageai.Detection import VideoObjectDetection
import os

def forSeconds(second_number, output_arrays, count_arrays, average_output_
count):
    print("SECOND : ", second_number)
    print("Array for the outputs of each frame ", output_arrays)
    print("Array for output count for unique objects in each frame : ", count_
arrays)
    print("Output average count for unique objects in the last second: ",
average_output_count)
    print("-----END OF A SECOND -----")

    video_detector = VideoObjectDetection()
    video_detector.setModelTypeAsYOLOv3()
    video_detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
    video_detector.loadModel()

    video_detector.detectObjectsFromVideo(input_file_path=os.path.join(execution_
path, "traffic.mp4"), output_file_path=os.path.join(execution_path, "video_second_
analysis"), frames_per_second=20, per_second_function=forSecond, minimum_
percentage_probability=30)

```

(continues on next page)

(continued from previous page)

Dans l'exemple ci-dessus, chaque seconde dans la vid?o est trait?e et d?tect?e, la fonction va recevoir et renvoyer les donn?es analytiques des objets d?tect?s dans la vid?o comme vous pouvez le voir ci-dessous?:

```

Array for the outputs of each frame [{"box_points": (362, 295, 443, 355), 'name':
↳ 'boat', 'percentage_probability': 26.666194200515747}, {'box_points': (319, 245,
↳ 386, 296), 'name': 'boat', 'percentage_probability': 30.052968859672546}, {'box_
↳ points': (219, 308, 341, 358), 'name': 'boat', 'percentage_probability': 47.
↳ 46982455253601}, {'box_points': (589, 198, 621, 241), 'name': 'bus', 'percentage_
↳ probability': 24.62330162525177}, {'box_points': (519, 181, 583, 263), 'name': 'bus
↳ ', 'percentage_probability': 27.446213364601135}, {'box_points': (493, 197, 561,
↳ 272), 'name': 'bus', 'percentage_probability': 59.81815457344055}, {'box_points':
↳ (432, 187, 491, 240), 'name': 'bus', 'percentage_probability': 64.42965269088745}, {
↳ 'box_points': (157, 225, 220, 255), 'name': 'car', 'percentage_probability': 21.
↳ 150341629981995}, {'box_points': (324, 249, 377, 293), 'name': 'car', 'percentage_
↳ probability': 24.089913070201874}, {'box_points': (152, 275, 260, 327), 'name': 'car
↳ ', 'percentage_probability': 30.341443419456482}, {'box_points': (433, 198, 485,
↳ 244), 'name': 'car', 'percentage_probability': 37.205660343170166}, {'box_points':
↳ (184, 226, 233, 260), 'name': 'car', 'percentage_probability': 38.52525353431702}, {
↳ 'box_points': (3, 296, 134, 359), 'name': 'car', 'percentage_probability': 47.
↳ 80363142490387}, {'box_points': (357, 302, 439, 359), 'name': 'car', 'percentage_
↳ probability': 47.94844686985016}, {'box_points': (481, 266, 546, 314), 'name': 'car
↳ ', 'percentage_probability': 65.8585786819458}, {'box_points': (597, 269, 624, 318),
↳ 'name': 'person', 'percentage_probability': 27.125394344329834}],
  [{"box_points": (316, 240, 384, 302), 'name': 'boat', 'percentage_probability':
↳ 29.594269394874573}, {'box_points': (361, 295, 441, 354), 'name': 'boat',
↳ 'percentage_probability': 36.11513376235962}, {'box_points': (216, 305, 340, 357),
↳ 'name': 'boat', 'percentage_probability': 44.89373862743378}, {'box_points': (432,
↳ 198, 488, 244), 'name': 'truck', 'percentage_probability': 22.914741933345795}, {
↳ 'box_points': (589, 199, 623, 240), 'name': 'bus', 'percentage_probability': 20.
↳ 545457303524017}, {'box_points': (519, 182, 583, 263), 'name': 'bus', 'percentage_
↳ probability': 24.467085301876068}, {'box_points': (492, 197, 563, 271), 'name': 'bus
↳ ', 'percentage_probability': 61.112016439437866}, {'box_points': (433, 188, 490,
↳ 241), 'name': 'bus', 'percentage_probability': 65.08989334106445}, {'box_points':
↳ (352, 303, 442, 357), 'name': 'car', 'percentage_probability': 20.025095343589783},
↳ {'box_points': (136, 172, 188, 195), 'name': 'car', 'percentage_probability': 21.
↳ 571354568004608}, {'box_points': (152, 276, 261, 326), 'name': 'car', 'percentage_
↳ probability': 33.07966589927673}, {'box_points': (181, 225, 230, 256), 'name': 'car
↳ ', 'percentage_probability': 35.111838579177856}, {'box_points': (432, 198, 488,
↳ 244), 'name': 'car', 'percentage_probability': 36.25282347202301}, {'box_points':
↳ (3, 292, 130, 360), 'name': 'car', 'percentage_probability': 67.55480170249939}, {
↳ 'box_points': (479, 265, 546, 314), 'name': 'car', 'percentage_probability': 71.
↳ 47912979125977}, {'box_points': (597, 269, 625, 318), 'name': 'person', 'percentage_
↳ probability': 25.903674960136414}],.....,
  [{"box_points": (133, 250, 187, 278), 'name': 'umbrella', 'percentage_probability
↳ ': 21.518094837665558}, {'box_points': (154, 233, 218, 259), 'name': 'umbrella',
↳ 'percentage_probability': 23.687003552913666}, {'box_points': (348, 311, 425, 360),
↳ 'name': 'boat', 'percentage_probability': 21.015766263008118}, {'box_points': (11,
↳ 164, 137, 225), 'name': 'bus', 'percentage_probability': 32.20453858375549}, {'box_
↳ points': (424, 187, 485, 243), 'name': 'bus', 'percentage_probability': 38.
↳ 043853640556335}, {'box_points': (496, 186, 570, 264), 'name': 'bus', 'percentage_
↳ probability': 63.83994221687317}, {'box_points': (588, 197, 622, 240), 'name': 'car
↳ ', 'percentage_probability': 23.51653128862381}, {'box_points': (58, 268, 111, 303),
↳ 'name': 'car', 'percentage_probability': 24.538707733154297}, {'box_points': (2,
↳ 246, 72, 301), 'name': 'car', 'percentage_probability': 28.433072566986084}, {'box_
↳ points': (472, 273, 539, 323), 'name': 'car', 'percentage_probability': 87.
↳ 17672824859619}, {'box_points': (597, 270, 626, 317), 'name': 'person', 'percentage_
↳ probability': 27.459821105003357}]

```

(continues on next page)

(continued from previous page)

```

]
Array for output count for unique objects in each frame : [{'bus': 4, 'boat': 3,
↪ 'person': 1, 'car': 8},
  {'truck': 1, 'bus': 4, 'boat': 3, 'person': 1, 'car': 7},
  {'bus': 5, 'boat': 2, 'person': 1, 'car': 5},
  {'bus': 5, 'boat': 1, 'person': 1, 'car': 9},
  {'truck': 1, 'bus': 2, 'car': 6, 'person': 1},
  {'truck': 2, 'bus': 4, 'boat': 2, 'person': 1, 'car': 7},
  {'truck': 1, 'bus': 3, 'car': 7, 'person': 1, 'umbrella': 1},
  {'bus': 4, 'car': 7, 'person': 1, 'umbrella': 2},
  {'bus': 3, 'car': 6, 'boat': 1, 'person': 1, 'umbrella': 3},
  {'bus': 3, 'car': 4, 'boat': 1, 'person': 1, 'umbrella': 2}]

Output average count for unique objects in the last second: {'truck': 0.5, 'bus': 3.7,
↪ 'umbrella': 0.8, 'boat': 1.3, 'person': 1.0, 'car': 6.6}

-----END OF A SECOND -----

```

Ci-dessous est le code complet qui a une fonction qui analyse les données analytiques et les visualise, et le frame d'arrêt ? la fin de la seconde en temps réel pendant que la vidéo est traitée et analysée :

```

from imageai.Detection import VideoObjectDetection import os from matplotlib import pyplot
as plt

```

```

execution_path = os.getcwd()

```

```

color_index = { 'bus': 'red', 'handbag': 'steelblue', 'giraffe': 'orange', 'spoon': 'gray', 'cup':
'yellow', 'chair': 'green', 'elephant': 'pink', 'truck': 'indigo', 'motorcycle': 'azure', 're-
frigerator': 'gold', 'keyboard': 'violet', 'cow': 'magenta', 'mouse': 'crimson', 'sports ball':
'raspberry', 'horse': 'maroon', 'cat': 'orchid', 'boat': 'slateblue', 'hot dog': 'navy', 'ap-
ple': 'cobalt', 'parking meter': 'aliceblue', 'sandwich': 'skyblue', 'skis': 'deepskyblue', 'mi-
crowave': 'peacock', 'knife': 'cadetblue', 'baseball bat': 'cyan', 'oven': 'lightcyan', 'car-
rot': 'coldgrey', 'scissors': 'seagreen', 'sheep': 'deepgreen', 'toothbrush': 'cobaltgreen',
'fire hydrant': 'limegreen', 'remote': 'forestgreen', 'bicycle': 'olivedrab', 'toilet': 'ivory',
'tv': 'khaki', 'skateboard': 'palegoldenrod', 'train': 'cornsilk', 'zebra': 'wheat', 'tie': 'burly-
wood', 'orange': 'melon', 'bird': 'bisque', 'dining table': 'chocolate', 'hair drier': 'sandy-
brown', 'cell phone': 'sienna', 'sink': 'coral', 'bench': 'salmon', 'bottle': 'brown', 'car': 'sil-
ver', 'bowl': 'maroon', 'tennis racket': 'palevioletred', 'airplane': 'lavenderblush', 'pizza':
'hotpink', 'umbrella': 'deeppink', 'bear': 'plum', 'fork': 'purple', 'laptop': 'indigo', 'vase':
'mediumpurple', 'baseball glove': 'slateblue', 'traffic light': 'mediumblue', 'bed': 'navy',
'broccoli': 'royalblue', 'backpack': 'slategray', 'snowboard': 'skyblue', 'kite': 'cadetblue',
'teddy bear': 'peacock', 'clock': 'lightcyan', 'wine glass': 'teal', 'frisbee': 'aquamarine',
'donut': 'mincream', 'suitcase': 'seagreen', 'dog': 'springgreen', 'banana': 'emeraldgreen',
'person': 'honeydew', 'surfboard': 'palegreen', 'cake': 'sagegreen', 'book': 'lawngreen', 'pot-
ted plant': 'greenyellow', 'toaster': 'ivory', 'stop sign': 'beige', 'couch': 'khaki' }

```

```

resized = False

```

```

def forSecond(frame2_number, output_arrays, count_arrays, average_count, returned_frame):

```

```

    plt.clf()

```

```

    this_colors = [] labels = [] sizes = []

```

```

    counter = 0

```

```

    for eachItem in average_count: counter += 1 labels.append(eachItem + " =
    " + str(average_count[eachItem])) sizes.append(average_count[eachItem])

```

```

        this_colors.append(color_index[eachItem])

global resized

if (resized == False): manager = plt.get_current_fig_manager() manager.resize(width=1000, height=500) resized = True

plt.subplot(1, 2, 1) plt.title("Second : " + str(frame_number)) plt.axis("off")
plt.imshow(returned_frame, interpolation="none")

plt.subplot(1, 2, 2) plt.title("Analysis: " + str(frame_number)) plt.pie(sizes,
labels=labels, colors=this_colors, shadow=True, startangle=140,
autopct="%1.1f%%")

plt.pause(0.01)

video_detector = VideoObjectDetection() video_detector.setModelTypeAsYOLOv3()
video_detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
video_detector.loadModel()

plt.show()

video_detector.detectObjectsFromVideo(input_file_path=os.path.join(execution_path,
"traffic.mp4"), output_file_path=os.path.join(execution_path, "video_second_analysis")
, frames_per_second=20, per_second_function=forSecond, minimum_percentage_probability=30, return_detected_frame=True, log_progress=True)

```

—*param?tre* **per\_minute\_function** (optionnel) : Ce param?tre peut ?tre transmis en argument du nom de la fonction que vous d?finissez. Puis, pour chaque frame de la vid?o qui est d?tect?e, le param?tre qui a ?t? transmis dans la fonction sera interpr?t? et les donn?es analytiques de la vid?o seront transmis ? la fonction. Les donn?es retourn?es sont de m?me nature que **per\_second\_function?**; la diff?rence est qu?elle ne tient compte que de tous les frames de la derni?re minute de la vid?o.

Retrouvez une exemple de fonction pour ces param?tres ci-dessous?:

```

def forMinute(minute_number, output_arrays, count_arrays, average_output_count):
    print("MINUTE : ", minute_number) print("Array for the outputs of each frame ",
    output_arrays) print("Array for output count for unique objects in each frame : ",
    count_arrays) print("Output average count for unique objects in the last minute: ",
    average_output_count) print("—————END OF A MINUTE —————")

```

—*param?tre* **video\_complete\_function** (optionnel) : Ce param?tre peut ?tre transmis en argument d?une fonction que vous d?finissez. Une fois que tous les frames de la vid?o sont totalement d?tect?s, le param?tre transmis sera interpr?t? et les donn?es analytiques de la vid?o seront transmis ? la fonction. Les donn?es retourn?es ont la m?me nature que **per\_second\_function** et **per\_minute\_function?**; les diff?rences sont qu?aucun index n?est renvoy? et ici tous les frames de la vid?o sont couvertes.

Retrouvez une exemple de fonction pour ces param?tres ci-dessous?:

```

def forFull(output_arrays, count_arrays, average_output_count): print("Array for the
    outputs of each frame ", output_arrays) print("Array for output count for unique objects in
    each frame : ", count_arrays) print("Output average count for unique objects in the entire
    video: ", average_output_count) print("—————END OF THE VIDEO —————")

```

## 1.4 Apprentissage personnalis? et pr?diction des Classes

**ImageAI** fournit des classes puissante et n?anmoins facile ? utiliser pour entrainer des algorithmes a la pointe de la technologies tel que **SqueezeNet**, **ResNet**, **InceptionV3** et **DenseNet** sur votre propre base de donn?es avec juste **5 lignes de code** pour g?n?rer votre propre mod?le personnalis?. Une fois que vous avez entrain? votre propre

modèle, vous pouvez utiliser la classe **CustomImagePrediction** fournit par **ImageAI** pour utiliser votre modèle pour reconnaître et faire la détection sur une image ou un ensemble d'images.

===== **imageai.Prediction.Custom.ModelTraining** =====

La classe **ModelTraining** vous permet d'entraîner un des quatre algorithmes d'apprentissage profond suivant (**SqueezeNet**, **ResNet**, **InceptionV3** et **DenseNet**) sur votre base de données d'images pour générer votre propre modèle. Votre base de données d'images doit contenir au moins deux différentes classes/types d'images (chat et chien) et vous devez ressembler au moins 500 images de chaque classe pour obtenir le maximum de précision possible.

Le processus d'entraînement génère un fichier JSON qui fait une correspondance entre les types d'objets dans votre base d'images et crée des modèles. Vous ferez le choix du modèle avec la précision la plus élevée et qui puisse faire la prédiction en utilisant le modèle et le fichier JSON généré.

Puisque la tâche d'apprentissage est gourmande en ressource, nous recommandons fortement de la faire à l'aide d'un ordinateur équipé d'un GPU NVIDIA et ayant la version GPU de Tensorflow installée. Faire l'apprentissage sur un CPU va demander beaucoup d'heures et de jours. Avec un système informatique équipé d'un GPU NVIDIA cela ne devrait prendre que quelques heures. Vous pouvez utiliser Google Colab pour cette expérience, puisqu'il est équipé d'un GPU NVIDIA K80.

Pour entraîner votre modèle de prédiction, vous devez préparer les images que vous voulez utiliser pour entraîner votre modèle. Vous préparerez les images comme suit:

- Créer un dossier avec le nom que vous aimeriez donner avec votre base de données (ex: Chats)
  - Dans le dossier que vous avez précédemment créé, créer un dossier que vous nommerez ?train?
- A côté du dossier train, créer un autre dossier et nommez le ?test?
  - Dans le dossier ?train?, créez un dossier pour chaque type d'objets que vous aimeriez que votre modèle reconnaisse et nommez le dossier selon la classe à prédire (ex: chien, chat, ?cureuil, serpents)
- Dans chaque dossier présent dans votre dossier ?train?, mettez-y les images de chaque objet ou classe. Ces images seront utilisées pour l'apprentissage de votre modèle.
  - pour générer un modèle qui puisse être viable pour des applications robustes, Je vous recommande d'avoir au moins 500 images ou de plus par objets. 1000 images par objets serait mieux.
- Dans le dossier ?Test?, créer des dossiers et nommez les selon les noms que vous avez utilisés pour le dossier ?Train?. Mettez environ 100 à 200 images correspondantes dans chaque dossier. Ces images seront celles utilisées pour tester le modèle après l'avoir entraîné.
  - Une fois que vous avez fait cela, la structure des dossiers de votre base d'images devrait être comme suit::

```
animaux/train/chien/chien-train-images animaux/train/chat/ chat -train-images animaux/train/lion // lion -train-images animaux/train/serpent// serpent -train-images
animaux/test/chien //chien-test-images animaux/test/chat // chat -test-images animaux/test/lion// lion -test-images animaux/test/serpent // serpent -test-images
```

Une fois que votre base de données est prête, vous pouvez créer une instance de la classe **ModelTraining**. Retrouver un exemple ci-dessous:

```
from imageai.Prediction.Custom import ModelTraining
model_trainer = ModelTraining()
```

Une fois que vous avez créé l'instance ci-dessous, vous pouvez utiliser les fonctions ci-dessous pour commencer le processus d'entraînement.

- `.setModelTypeAsSqueezeNet()` , Cette fonction ?tablit comme type de mod?le pour votre instance d?entraînement le mod?le **SqueezeNet**, ceci veut dire que l?algorithme **SqueezeNet** sera utilis? pour entraîner votre mod?le. Trouver un exemple de code ci-dessous?

```
model_trainer.setModelTypeAsSqueezeNet ()
```

- `.setModelTypeAsResNet()` , Cette fonction ?tablit comme type de mod?le pour votre instance d?entraînement le mod?le **ResNet**, ceci veut dire que l?algorithme **ResNet** sera utilis? pour entraîner votre mod?le. Trouver un exemple de code ci-dessous

```
model_trainer.setModelTypeAsResNet ()
```

- `.setModelTypeAsInceptionV3()` , Cette fonction ?tablit comme type de mod?le pour votre instance d?entraînement le mod?le **InceptionV3**, ceci veut dire que l?algorithme **InceptionV3** sera utilis? pour entraîner votre mod?le. Trouver un exemple de code ci-dessous?

```
model_trainer.setModelTypeAsInceptionV3 ()
```

- `.setModelTypeAsDenseNet()` , Cette fonction ?tablit comme type de mod?le pour votre instance d?entraînement le mod?le **DenseNet**, ceci veut dire que l?algorithme **DenseNet** sera utilis? pour entraîner votre mod?le. Trouver un exemple de code ci-dessous?

```
model_trainer.setModelTypeAsDenseNet ()
```

- `.setDataDirectory()` , Cette fonction prend en argument une chaine de caract?re qui doit ?tre le chemin vers le dossier qui contient les sous-dossiers **test** et **train** qui contiennent votre base d?images. Retrouver un exemple d?utilisation de la fonction, et de ses param?tres ci-dessous

```
prediction.setDataDirectory ("C:/Users/Moses/Documents/Moses/AI/Custom Datasets/  
↪ animaux")
```

– *parametre* **data\_directory** (obligatoire) : Il s?agit du chemin vers le dossier qui contient votre base d?images.

- `.trainModel()` , Il s?agit de la fonction qui commence le processus d?entraînement. Une fois commenc?, il cr?era un fichier JSON dans le dossier **dataset/json** (ex **animaux/json**) qui contient la correspondance de chaque classe dans la base d?images. Le fichier JSON sera utilis? pendant la d?tection personnalis?e pour produire les r?sultats. Trouvez un exemple de code ci-dessous

```
model_trainer.trainModel(num_objects=4, num_experiments=100, enhance_data=True, ↵  
↪ batch_size=32, show_network_summary=True)
```

– *param?tre* **num\_objects** (obligatoire) : Ceci fait r?f?rence au nombre de diff?rentes classes dans votre base d?images.

– *param?tre* **num\_experiments** (obligatoire) : Il repr?sente le nombre de fois que l?algorithme sera entrain? sur la base d?images. La pr?cision de votre entraînement augmente avec le nombre d?it?rations ou d?entraînement. Cependant la pr?cision atteint son maximum avec un certain nombre d?it?ration et nombre d?pend de la taille et de la nature de base de donn?es.

– *param?tre* **enhance\_data** (optionnel) : Ce param?tre est utilis? pour transformer votre base d?images en g?n?rant plus d?chantillons pour la phase d?entraînement. Par d?faut sa valeur est ?False?. N?anmoins, il est important de lui donner la valeur ?True? lorsque votre base d?images contient moins de 1000 images par classe.

– *param?tre* **batch\_size** (optionnel) : Pendant la phase d?entraînement, L?algorithme est entrain? sur un ensemble d?images en parall?le. A cause de cela, la valeur par d?faut est mise ? 32. Vous pouvez accroître ou d?croître cette valeur selon votre connaissance du syst?me que vous utilisez pour

l'apprentissage. Si vous envisagez de changer cette valeur, vous devrez utiliser des multiples de 8 pour optimiser le processus d'apprentissage.

– *param?tre* **show\_network\_summary** (optionnel) : Lorsque ce param?tre a la valeur ?True?, il affiche la structure de l'algorithme que vous utilisez pour l'apprentissage sur vos images dans une petite console avant de commencer l'apprentissage. Sa valeur par d?faut est ?False?.

– *param?tre* **initial\_learning\_rate** (optionnel) : Ce param?tre a une haute valeur technique. Il d?termine et contr?le le comportement de votre apprentissage, ce qui est critique pour la pr?cision ? r?aliser. Vous pouvez changer la valeur de ce param?tre si vous avez une pleine compr?hension de sa fonctionnalit?.

– *training\_image\_size* **initial\_learning\_rate** (optionnel) : Il repr?sente la taille que vos images prendront pendant le processus d'apprentissage, peu importe leur taille d'origine. La valeur par d?faut est de 224 et elle ne doit pas aller en dessous de 100. Augmenter sa valeur permettra de gagner en pr?cision mais augmentera aussi le temps d'apprentissage et vice-versa.

### Exemple de code pour un model apprentissage personnalis

Trouvez ci-dessous un exemple de code lors de l'apprentissage d'un mod?le personnalis? sur votre base d'images?:

```
from imageai.Prediction.Custom import ModelTraining

model_trainer = ModelTraining()
model_trainer.setModelTypeAsResNet()
model_trainer.setDataDirectory(r"C:/Users/Moses/Documents/Moses/AI/Custom Datasets/
↳animaux")
model_trainer.trainModel(num_objects=10, num_experiments=100, enhance_data=True,
↳batch_size=32, show_network_summary=True)
```

Ci-dessous est un aper?u de r?sultat lorsque l'apprentissage commence?:

```
Epoch 1/100
1/25 [>.....] - ETA: 52s - loss: 2.3026 - acc: 0.2500
2/25 [==>.....] - ETA: 41s - loss: 2.3027 - acc: 0.1250
3/25 [==>.....] - ETA: 37s - loss: 2.2961 - acc: 0.1667
4/25 [===>.....] - ETA: 36s - loss: 2.2980 - acc: 0.1250
5/25 [====>.....] - ETA: 33s - loss: 2.3178 - acc: 0.1000
6/25 [=====>.....] - ETA: 31s - loss: 2.3214 - acc: 0.0833
7/25 [=====>.....] - ETA: 30s - loss: 2.3202 - acc: 0.0714
8/25 [=====>.....] - ETA: 29s - loss: 2.3207 - acc: 0.0625
9/25 [=====>.....] - ETA: 27s - loss: 2.3191 - acc: 0.0556
10/25 [=====>.....] - ETA: 25s - loss: 2.3167 - acc: 0.0750
11/25 [=====>.....] - ETA: 23s - loss: 2.3162 - acc: 0.0682
12/25 [=====>.....] - ETA: 21s - loss: 2.3143 - acc: 0.0833
13/25 [=====>.....] - ETA: 20s - loss: 2.3135 - acc: 0.0769
14/25 [=====>.....] - ETA: 18s - loss: 2.3132 - acc: 0.0714
15/25 [=====>.....] - ETA: 16s - loss: 2.3128 - acc: 0.0667
16/25 [=====>.....] - ETA: 15s - loss: 2.3121 - acc: 0.0781
17/25 [=====>.....] - ETA: 13s - loss: 2.3116 - acc: 0.0735
18/25 [=====>.....] - ETA: 12s - loss: 2.3114 - acc: 0.0694
19/25 [=====>.....] - ETA: 10s - loss: 2.3112 - acc: 0.0658
20/25 [=====>.....] - ETA: 8s - loss: 2.3109 - acc: 0.0625
21/25 [=====>.....] - ETA: 7s - loss: 2.3107 - acc: 0.0595
22/25 [=====>.....] - ETA: 5s - loss: 2.3104 - acc: 0.0568
23/25 [=====>.....] - ETA: 3s - loss: 2.3101 - acc: 0.0543
24/25 [=====>.....] - ETA: 1s - loss: 2.3097 - acc: 0.0625Epoch
↳00000: saving model to C:\Users\Moses\Documents\Moses\W7\AI\Custom
↳Datasets\IDENPROF\idenprof-small-test\idenprof\models\model_ex-000_acc-0.100000.h5
```

(continues on next page)

(continued from previous page)

```
25/25 [=====] - 51s - loss: 2.3095 - acc: 0.0600 - val_loss: 2.3026 - val_acc: 0.1000
```

Expliquons les d?tails ci-dessus?:

1. La ligne Epoch 1/100 signifie que le r?seau fait le premier apprentissage sur les 100 voulus. 0
2. La ligne 1/25 [>.....] - ETA: 52s - loss: 2.3026 - acc: 0.2500 repr?sente le nombre de groupe qui ont ?t? entrain? dans la pr?sente phase d'apprentissage.
3. La ligne Epoch 00000: sauvegarde le mod?le ? l'emplacement C:\Users\User\PycharmProjects\ImageAITest\pets\models\model\_ex-000acc-0.100000.h5 ? la fin de la phase d'apprentissage pr?sente. ex\_000 repr?sente le niveau d'apprentissage tandis que acc0.100000 et valacc: 0.1000 repr?sente la pr?cision du mod?le sur l'ensemble d'images ?Test? apr?s le pr?sente apprentissage (La valeur maximale de la pr?cision est de 1.0). Ce r?sultat vous permet de connaitre le meilleur mod?le a utiliser pour la d?tection sur vos images.

Une fois que vous avez termin? l'apprentissage de votre mod?le termine, vous pouvez utiliser la classe **CustomImagePrediction** d?crite si dessous pour la d?tection avec votre mod?le.

```
===== imageai.Prediction.Custom.CustomImagePrediction =====
```

Cette classe peut ?tre consid?r?e comme une r?plique de **imageai.Prediction.ImagePrediction** puis qu'elle a les m?me fonctions, param?tres et r?sultats. La seule diff?rence est que cette classe fonctionne avec votre mod?le personnalis?. Vous aurez besoin de sp?cifier le chemin du fichier JSON g?n?r? pendant la phase d'apprentissage et aussi de sp?cifier le nombre de classe dans votre base d'image lors du chargement du mod?le. Ci-dessous est un exemple de cr?ation d'instance de la classe?:

```
from imageai.Prediction.Custom import CustomImagePrediction

prediction = CustomImagePrediction()
```

Une fois que vous avez cr?? l'instance, vous pouvez utiliser les fonctions ci-dessous pour configurer les propri?t?s de votre instance et commencer le processus de d?tection et reconnaissance sur des images.

- **.setModelTypeAsSqueezeNet()** , Cette fonction ?tablit comme type de mod?le pour votre instance de reconnaissance et d?tection, le mod?le **SqueezeNet**, ceci veut dire que l'algorithmme **SqueezeNet** g?n?r? pendant votre phase d'apprentissage personnalis?e sera utilis? pour la tache de pr?diction sur vos images. Trouver un exemple de code ci-dessous?:

```
prediction.setModelTypeAsSqueezeNet()
```

- **.setModelTypeAsResNet()** , Cette fonction ?tablit comme type de mod?le pour votre instance de reconnaissance et d?tection, le mod?le **ResNet**, ceci veut dire que l'algorithmme **ResNet** g?n?r? pendant votre phase d'apprentissage personnalis?e sera utilis? pour la tache de pr?diction sur vos images. Trouver un exemple de code ci-dessous?:

```
prediction.setModelTypeAsResNet()
```

- **.setModelTypeAsInceptionV3()** , Cette fonction ?tablit comme type de mod?le pour votre instance de reconnaissance et d?tection, le mod?le **InceptionV3**, ceci veut dire que l'algorithmme **InceptionV3** g?n?r? pendant votre phase d'apprentissage personnalis?e sera utilis? pour la tache de pr?diction sur vos images. Trouver un exemple de code ci-dessous?:

```
prediction.setModelTypeAsInceptionV3()
```

- **.setModelTypeAsDenseNet()** , Cette fonction ?tablit comme type de mod?le pour votre instance de reconnaissance et d?tection, le mod?le **DenseNet**, ceci veut dire que l'algorithmme **DenseNet** g?n?r? pendant votre

phase d'apprentissage personnalisée sera utilisée pour la tâche de prédiction sur vos images. Trouver un exemple de code ci-dessous:

```
prediction.setModelTypeAsDenseNet()
```

- **.setModelPath()**, cette fonction accepte une chaîne de caractères qui doit être le chemin vers le fichier modèle pendant votre phase d'apprentissage et doit correspondre au type de modèle que vous avez défini pour votre instance de reconnaissance d'images. Trouver un exemple de code, et de paramètres de fonction ci-dessous:

```
prediction.setModelPath("resnet_model_ex-020_acc-0.651714.h5")
```

– *paramètre* **model\_path** (requis) : Il s'agit du chemin vers le fichier modèle téléchargé.

- **.setJsonPath()**, cette fonction prend en argument une chaîne de caractères qui représente le chemin vers le fichier JSON pendant la phase d'apprentissage du modèle personnalisé. Trouvez ci-dessous un exemple de code et de paramètres de la fonction:

```
prediction.setJsonPath("model_class.json")
```

– *paramètre* **model\_path** (requis) : Il s'agit du chemin vers le fichier modèle téléchargé.

- **.loadModel()**, Cette fonction charge le modèle à partir du chemin spécifié dans votre appel de fonction ci-dessus pour votre instance de prédiction d'images. Au paramètre **num\_objects** vous devrez donner la valeur correspondant au nombre de classes dans votre base d'images. Trouvez ci-dessous un exemple de code et de paramètres de la fonction:

```
prediction.loadModel(num_objects=4)
```

– *paramètre* **num\_objects** (requis) : La valeur de ce paramètre doit correspondre au nombre de classe dans votre base d'images.

– *paramètre* **prediction\_speed** (optionnel) : Ce paramètre vous permet de réduire le temps de prédiction sur une image d'environ 80% ce qui conduit à une légère réduction de la précision. Ce paramètre prend des valeurs de type chaîne de caractères. Les valeurs disponibles sont: "normal", "fast", "faster" et "fastest". La valeur par défaut est "normal"

- **.predictImage()**, C'est la fonction qui accomplit proprement parler la prédiction sur une image. Elle peut être appelée plusieurs fois sur plusieurs images une fois que le modèle a été chargé dans l'instance de prédiction. Trouver ci-dessous un exemple de code, de paramètres de fonction ainsi que les valeurs renvoyées:

```
predictions, probabilities = prediction.predictImage("image1.jpg", result_count=2)
```

– *paramètre* **image\_input** (requis) : Il fait référence au chemin vers votre image, le tableau de type Numpy de votre image ou le flux de votre image, indépendamment de type de valeur d'entrée spécifiée.

— *paramètre* **result\_count** (optionnel) : il fait référence au nombre possible de prédiction qui peuvent être données. Ce paramètre a pour valeur par défaut 5.

– *paramètre* **input\_type** (optionnel) : Il fait référence au type de la valeur d'entrée que vous passez au paramètre **image\_input**. Il est de type `file` par défaut et accepte `stream` et `array` aussi.

— *valeur retournée* **prediction\_results** (une liste python) :

La première valeur retournée par la fonction **predictImage** est une liste qui contient tous les résultats possibles de prédiction. Les résultats sont ordonnés en ordre descendant de pourcentage de probabilité.

– *valeur retournée* **prediction\_probabilities** (une liste python) :

La première valeur retournée par la fonction **predictImage** est une liste qui contient les pourcentages de probabilité correspondantes à toutes les prédictions possibles dans **prediction\_results**

- **.predictMultipleImages()** , Cette fonction peut être utilisée pour effectuer la tâche de prédiction sur 2 ou plusieurs images en une seule fois. Trouvez ci-dessous un exemple de code, paramètres de fonction et de valeurs renvoyées:

```

results_array = multiple_prediction.predictMultipleImages(all_images_array,
↳result_count_per_image=2)

for each_result in results_array:
    predictions, percentage_probabilities = each_result["predictions"], each_
↳result["percentage_probabilities"]
    for index in range(len(predictions)):
        print(predictions[index] , " : " , percentage_probabilities[index])
    print("-----")

```

– *paramètre sent\_images\_array* (requis) : Il fait référence à une liste qui contient le chemin vers vos fichiers image, vos tableau Numpy de vos images ou vos fichiers de flux d'images, indépendamment du type de valeur d'entrée spécifiée.

– *paramètre result\_count\_per\_image* (optionnel) : Il fait référence au nombre possible de prédictions qui doivent être données pour chaque image. Ce paramètre a pour valeur par défaut 2.

– *paramètre input\_type* (optionnel) : Il fait référence au format de vos images ont dans la liste du paramètre **sent\_images\_array**. Il est de type `file` par défaut et accepte `stream` et `array` aussi.

– *valeur retournée output\_array* (une liste python) :

La valeur retournée par la fonction **predictMultipleImages** est une liste qui contient des dictionnaires. Chaque dictionnaire correspond à une image contenue dans le tableau envoyé par **sent\_images\_array**. Chaque dictionnaire a une propriété "prediction\_results" qui est une liste de tous les résultats de prédiction pour l'image à cet indice aussi bien que la probabilité de prédiction "prediction\_probabilities" qui est une liste de pourcentage de probabilité correspondant à chaque résultat.

### Exemple de code

Trouvez ci-dessous un échantillon de code pour la prédiction personnalisée:

```

from imageai.Prediction.Custom import CustomImagePrediction
import os

execution_path = os.getcwd()

prediction = CustomImagePrediction()
prediction.setModelTypeAsResNet()
prediction.setModelPath(os.path.join(execution_path, "resnet_model_ex-020_acc-0.
↳651714.h5"))
prediction.setJsonPath(os.path.join(execution_path, "model_class.json"))
prediction.loadModel(num_objects=4)

predictions, probabilities = prediction.predictImage(os.path.join(execution_path, "4.
↳jpg"), result_count=5)

for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , " : " , eachProbability)

```

## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`