ImageAI Documentation

Version 2.0.2

"Moses Olafenwa" "John Olafenwa"

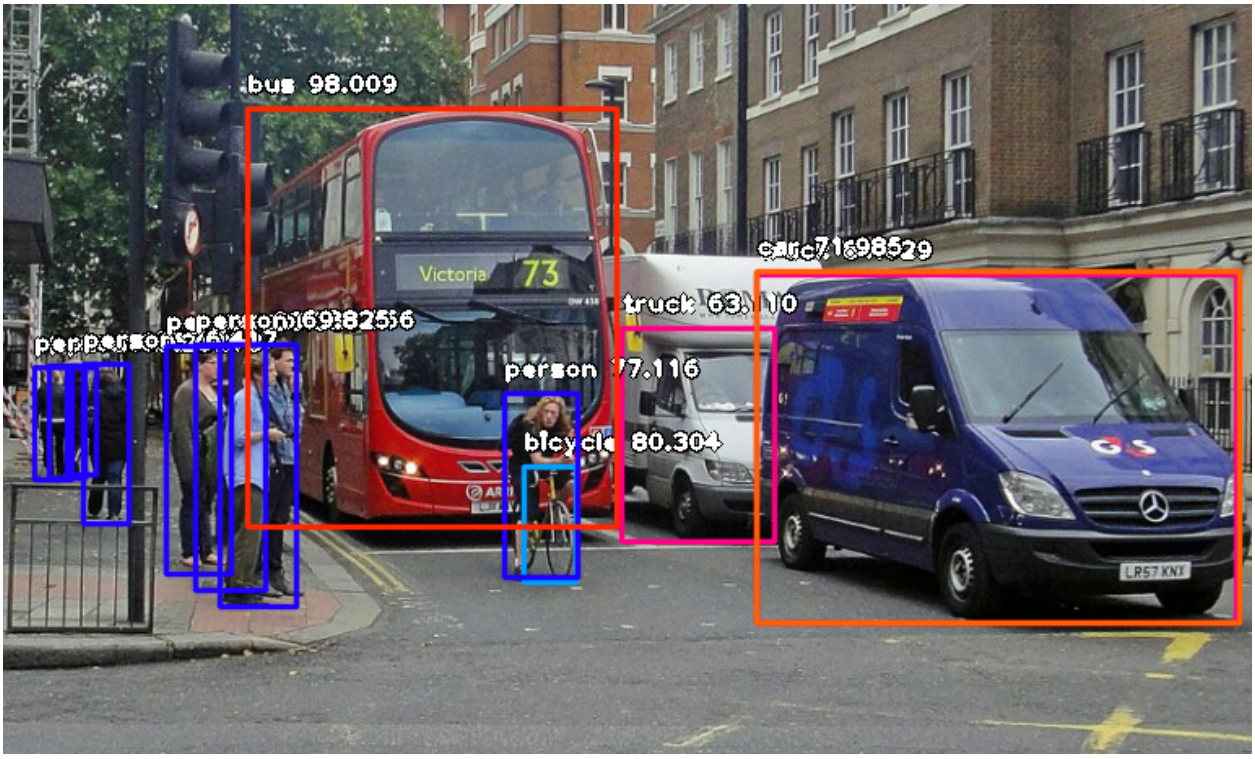
oct. 28, 2018

Contents:

ImageAI est une bibliothque python dŽveloppŽe pour permettre aux dŽveloppeurs, chercheurs, Žtudiants de construire des applications et des systmes qui intgrent lÕapprentissage profond et la vision assistŽe par ordinateur en utilisant simplement quelques lignes de code. Cette documentation est fournie pour donner assez de dŽtails sur toutes classes et fonctions disponibles dans **ImageAI**, couples a un certain nombre dÕexemples de code.

ImageAI est un projet dŽveloppŽ par Moses Olafenwa et John Olafenwa, the DeepQuest AI team.

Le dossier official GitHub de **ImageAI** est https://github.com/OlafenwaMoses/ImageAI



CHAPITRE 1

Installation de ImageAI

ImageAI nŽcessite que vous ayez Python 3.5.1 ou supŽrieur installe ainsi que dÕautres librairies et bibliothques python. Avant de procŽder a lÕinstallation de **ImageAI** vous devez installer les ŽlŽments suivantsÊ :

— **Python** 3.5.1 or superieur, telecharger Python
— **pip3** , ‘ telecharger PyPi <https://pypi.python.org/pypi/pip/>‘_
— **Tensorflow** 1.4.0 or superieur

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

— **Numpy** 1.13.1 or superieur

```
pip3 install numpy
```

— **SciPy** .19.1 or superieur

```
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 installŽ 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 installŽ, vous pouvez en quelques lignes de code accomplir les t‰ches de vision assistŽe par ordinateur les plus puissantes comme vous pouvez le voir ci-dessous. **Reconnaissance dÕImage**

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

DŽtection dÕobjets sur Image

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

DŽtection dÕobjets sur VidŽo

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

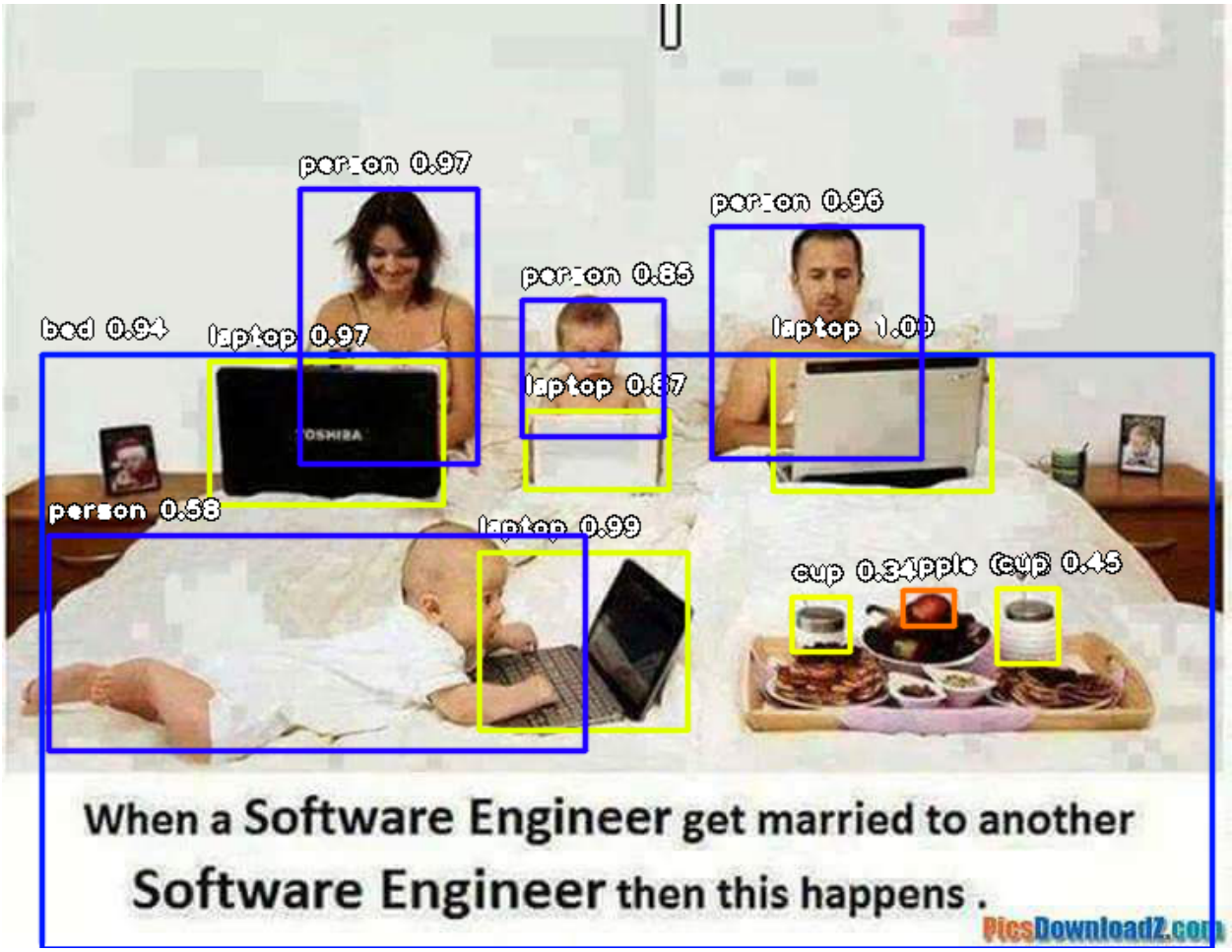
Analyse de dŽtection VidŽo

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

InfŽrence et entrainement personnalise pour reconnaissance dÕimages

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 complte des classes et fonctions disponibles.


person 0.97
person 0.96
person 0.86
bed 0.94
laptop 0.97
laptop 1.00
laptop 0.87
person 0.58
laptop 0.99
cup 0.34
apple
cup 0.45
When a Software Engineer get married to another
Software Engineer then this happens .
PicsDownloadZ.com

1.1 Classes de prŽdiction

ImageAI fournit un ensemble de classes puissantes et faciles ^ utiliser pour accomplir les t‰ches de *reconnaissance sur les images*. Vous pouvez pourrez accomplir toutes ces taches de pointe de vision assistŽe par ordinateur avec du code python allant de 5 ^ 12 lignes de code. Une fois que le python installe, dÕautres bibliothques et **ImageAI** installes dans votre ordinateur, il nÕy a aucune limite aux applications incroyables que vous pouvez crŽer. Trouvez ci-dessous les classes et leur fonction respective rendues disponibles pour votre utilisation. Ces classes peuvent tre intŽgrŽes dans nÕimporte quelle programme python traditionnel que vous dŽveloppez, que cela soit un site internet, une application Windows/Linux/MacOS ou un systme qui supporte ou fait partir dÕun rŽseau local.

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

La classe **ImagePrediction** vous fournit des fonctions pour utiliser les modles de reconnaissance dÕimages les plus pointus tel **SqueezeNet**, **ResNet**, **InceptionV3** et **DenseNet** qui ont ŽtŽ **prŽ-entraines** sur la base de donnŽes **ImageNet-1000**. Ceci pour dire que vous pouvez utiliser ces classes pour dŽtecter et reconnaitre plus de 1000 diffŽrents objets sur nÕimporte quelle image ou ensemble dÕimages. Pour initialiser la classe dans votre code, vous allez crŽer une instance dans votre code comme suitÊ :

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

Nous avons fourni les modles prŽ-entraines de reconnaissance dÕimages 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 tŽlŽcharger les modles. Vous pouvez tŽlŽcharger le modle que vous voulez utiliser.

Telechargez le modele SqueezeNet

TŽlŽchargez le modle ResNet

‘ TŽlŽchargez le modle InceptionV3 <https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0 />‘_

‘ TŽlŽchargez le modle DenseNet <https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0 />‘_

Aprs avoir crŽŽ une nouvelle instance de la classe **ImagePrediction**, Vous pouvez utiliser les fonctions ci-dessous pour dŽfinir les valeurs des propriŽtŽs et commencer la reconnaissance.

— **.setModelTypeAsSqueezeNet()** , cette fonction Žtablit comme modle pour votre instance de reconnaissance dÕimage que vous avez crŽŽ, le modle **SqueezeNet**Ê ; ce qui veut dire que vous accomplirez vos taches de prŽdiction en utilisant les modles prŽ-entrainŽs de **SqueezeNet** que vous avez tŽlŽchargŽ avec le lien ci-dessus. Trouvez le code ci-dessousÊ :

```
prediction.setModelTypeAsSqueezeNet()
```

— **.setModelTypeAsResNet()** , cette fonction Žtablit comme modle pour votre instance de reconnaissance dÕimage que vous avez crŽŽ, le modle **ResNet**Ê ; ce qui veut dire que vous accomplirez vos taches de prŽdiction en utilisant les modles prŽ-entrainŽs de **ResNet** que vous avez tŽlŽchargŽ avec le lien ci-dessus. Trouvez le code ci-dessousÊ :

```
prediction.setModelTypeAsResNet()
```

— **.setModelTypeAsInceptionV3()** , cette fonction Žtablit comme modle pour votre instance de reconnaissance dÕimage que vous avez crŽŽ, le modle **InceptionV3**Ê ; ce qui veut dire que vous accomplirez vos taches de prŽdiction en utilisant les modles prŽ-entrainŽs de **InceptionV3** que vous avez tŽlŽchargŽ avec le lien ci-dessus. Trouvez le code ci-dessousÊ :

```
prediction.setModelTypeAsInceptionV3()
```

— **.setModelTypeAsDenseNet()** , cette fonction Žtablit comme modle pour votre instance de reconnaissance dÕimage que vous avez crŽŽ, le modle **DenseNet**Ê ; ce qui veut dire que vous accomplirez vos taches

de prŽdiction en utilisant les modles prŽ-entrainŽs de **DenseNet que vous avez tŽlŽchargŽ avec le lien ci-dessus. Trouvez le code ci-dessousÊ :

```
prediction.setModelTypeAsDenseNet()
```

— **.setModelPath() , cette fonction accepte une chaine de caractre qui doit tre le chemin vers le fichier modle que vous avez t**
prediction.setModelPath(« resnet50_weights_tf_dim_ordering_tf_kernels.h5 »)

– *paramtre* **model_path** (requis) : Il sÕagit du chemin vers votre fichier modle tŽlŽchargŽ.

— **.loadModel()** , Cette fonction charge le modle ^ partir du chemin que vous avez spŽcifiŽ dans lÕappel de fonction ci-dessus dans votre instance de prŽdiction. Trouvez un exemple de code ci-dessousÊ :

```
prediction.loadModel()
```

– *paramtre* **prediction_speed** (optionnel) : Ce paramtre vous permet de rŽduire jusquÕ^ 80% le temps quÕil faut pour la tache de prŽdiction sur une image, ce qui conduit ^ une lŽgre rŽduction de la prŽcision. Ce paramtre accepte les chaines de caractres. Les valeurs disponibles sont « normal », « fast », « faster » et « fastest ». La valeur par dŽfaut est « normal ».

— **.predictImage()** , CÕest la fonction qui effectue la tache de prŽdiction a proprement parle sur une image. Elle peut tre appelŽe plusieurs fois sur plusieurs images une fois que le modle a ŽtŽ charge dans lÕinstance de prŽdiction. Trouvez un exemple de code, et paramtres de fonction ci-dessousÊ :

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

– *paramtre* **image_input** (requis) : Il fait rŽfŽrence au chemin vers votre fichier images, tableau Numpy de votre image ou le fichier flux de votre image, dŽpendamment du type que vous avez choisi.

—*paramtre* **result_count** (optionnel) : Il fait rŽfŽrence au nombre possible de prŽdictions qui doivent tre retourne. Le paramtre a une valeur par dŽfaut de 5.

– *paramtre* **input_type** (optionnel) : Il fait rŽfŽrence au type de la valeur dÕentrŽe dans le paramtre **image_input**. Il est ÔfileÕ par dŽfaut et accepte ÔarrayÕ et ÔstreamÕ aussi. Ê

—*valeur retournŽe* **prediction_results** (une liste python) : La premire valeur renvoyŽe par la fonction **predictImage** est une liste qui contient tous les rŽsultats possibles de prŽdiction. Les rŽsultats sont arrangŽs dans lÕordre descendant de probabilitŽ de pourcentage.

– *valeur retournŽe* **prediction_probabilities** (une liste python) :

La seconde valeur renvoyŽe par la fonction **predictImage** est une liste qui contient les pourcentages de probabilitŽ correspondant ^ toutes les prŽdictions possibles dans **prediction_results**

— **.predictMultipleImages()** , Cette fonction pour accomplir la prŽdictions sur 2 ou plusieurs images ^ la fois. Trouvez un exemple de code, et paramtres de fonction ci-dessousÊ :

```
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("----------------------")
```

– *paramtre* **sent_images_array** (requis) : Il fait rŽfŽrence a une liste qui contient le chemin vers les fichiers images, les tableaux Numpy de vos images ou les fichiers de flux de vos images, dŽpendamment de type spŽcifie pour la valeur dÕentrŽe.
– *paramtre* **result_count_per_image** (optionnel) : Il fait rŽfŽrence au nombre de possible de prŽdictions renvoyŽes pour chaque image. Ce paramtre a pour valeur par dŽfaut 2.
– *paramtre* **input_type** (optionnel) : Il fait rŽfŽrence au format dans lequel vos images sont reprŽsentŽes dans la liste contenu dans le paramtre **sent_images_array**. Il est par dŽfaut ÔfileÕ et accepte aussi ÔarrayÕ et ÔstreamÕ.
– *valeur retournŽe* **output_array** (une liste python) : La valeur retournŽe par la fonction **predictMultipleImages** est une liste qui contient des dictionnaires. Chaque dictionnaire correspond ^ une image contenue dans le tableau transmis a **sent_images_array**. Chaque dictionnaire a une propriŽtŽ « prediction_results » qui est la liste de tous les rŽsultats de prŽdictions sur lÕimage a cet indice ainsi que la Ôprediction_probabilitiesÕ qui est la liste correspondant au pourcentage de probabilitŽ de chaque rŽsultat.

Exemple de code

Trouver ci-dessous un Žchantillon de code pour la prŽdiction sur une imageÊ :

```
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 Žchantillon de code pour la dŽtection/prŽdiction sur plusieurs imagesÊ :

```
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:
```

(suite sur la page suivante)

(suite de la page précédente)

```
    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 **DŽtection et Extraction dÕobjets dans une image**. **ImageAI** vous permet dÕutiliser les algorithmes de pointe en apprentissage profond tel que **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Avec **ImageAI** vous pouvez accomplir des taches de dŽtection et dÕanalyse dÕimages. Trouvez ci-dessous les classes et leurs fonctions respectives mise ^ votre disposition pour votre utilisation. Ces classes peuvent tre intŽgrŽes dans tout programme Python traditionnel que vous dŽveloppez ; que ce soit un site internet, une application Windows/Linux/MacOS ou un systme qui supporte ou fait partir dÕun rŽseau local.

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

Cette classe **ObjectDetection** vous fournit les fonctions pour accomplir la dŽtection dÕobjets sur une image ou un ensemble dÕimages, utilisant les modles **prŽ-entraines** sur la base de donnŽes **COCO**. Les modles supportŽs sont **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Ceci veut dire que vous pouvez dŽtecter et reconnaitre 80 diffŽrents types communs dÕobjets de tous les jours. Pour commencer, tŽlŽcharger nÕimporte quel modle que vous voulez utiliser via les liens ci-dessousÊ :

TŽlŽcharger le modle RetinaNet - resnet50_coco_best_v2.0.1.h5

TŽlŽcharger le modle YOLOv3 - yolo.h5

TŽlŽcharger le modle TinyYOLOv3 - yolo-tiny.h5

Une fois que vous avez tŽlŽcharger le modle de votre choix, vous devez crŽer une instance de la classe **ObjectDetection** comme dans lÕexemple ci-dessousÊ :

```
from imageai.Detection import ObjectDetection

detector = ObjectDetection()
```

Une fois que vois vous avez crŽŽ une instance de la classe, vous pouvez utiliser les fonctions ci-dessous pour choisir convenablement les propriŽtŽs dÕinstance et de commencer la dŽtection dÕobjets sur les images.

— **.setModelTypeAsRetinaNet()** , cette fonction Žtablit comme type de modle pour lÕinstance de dŽtection dÕobjets que vous avez crŽŽ le modle **RetinaNet**, ceci veut dire que vous accomplirez la tache de dŽtection dÕobjets ^ lÕaide de modle prŽ-entrainŽ de **RetinaNet** que vous avez tŽlŽchargŽ par les liens ci-dessus. Trouvez un exemple de code ci-dessousÊ :

```
detector.setModelTypeAsRetinaNet()
```

— **.setModelTypeAsYOLOv3()** , cette fonction Žtablit comme type de modle pour lÕinstance de dŽtection dÕobjets que vous avez crŽŽ le modle **YOLOv3**, ceci veut dire que vous accomplirez la tache de dŽtection dÕobjets ^ lÕaide de modle prŽ-entrainŽ de **YOLOv3** que vous avez tŽlŽchargŽ par les liens ci-dessus. Trouvez un exemple de code ci-dessousÊ :

```
detector.setModelTypeAsYOLOv3()
```

— **.setModelTypeAsTinyYOLOv3()** , cette fonction Žtablit comme type de modle pour lÕinstance de dŽtection dÕobjets que vous avez crŽŽ le modle **TinyYOLOv3**, ceci veut dire que vous accomplirez la tache de dŽ-

tection dÕobjets ^ lÕaide de modle prŽ-entrainŽ de **TinyYOLOv3** que vous avez tŽlŽchargŽ par les liens ci-dessus. Trouvez un exemple de code ci-dessousÊ :

```
detector.setModelTypeAsTinyYOLOv3()
```

— **.setModelPath()** , Cette fonction prend en argument une chaine de caractres qui doit tre le chemin vers le fichier modle que vous avez tŽlŽchargŽ et doit correspondre au type de modle choisi pour votre instance de dŽtection dÕobjets. Trouvez un exemple de code et de paramtres de fonction ci-dessous :

detector.setModelPath(« yolo.h5 »)

– *paramtre* **model_path** (requis) : cÕest le chemin vers votre modle tŽlŽchargŽ.

— **.loadModel()** , Cette fonction charge le modle ^ partir du chemin que vous avez spŽcifiŽ dans lÕappel de fonction ci-dessus de votre instance de dŽtection dÕobjets. Trouver un exemple de code ci-dessousÊ :

```
detector.loadModel()
```

– *paramtre* **detection_speed** (optionnel) : Ce paramtre vous permet de rŽduire jusquÕ^ 80% le temps quÕil faut pour dŽtecter les objets sur une image ce qui conduit ^ une lŽgre rŽduction de la prŽcision. Ce paramtre accepte des valeurs de type chaines de caractres. Les valeurs disponibles sont **normal**, **fast**, **faster**, **fastest** et **flash**. La valeur par dŽfaut est **normal**

— **.detectObjectsFromImage()** ,CÕest la fonction qui accomplit la dŽtection dÕobjets aprs que le modle ait ŽtŽ chargŽ. Elle peut tre appelŽe plusieurs fois pour dŽtecter les objets dans plusieurs images. Trouvez un exemple de code ci-dessousÊ :

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

– *paramtre* **input_image** (requis) : Il fait rŽfŽrence au chemin vers le fichier image sur lequel vous voulez faire la dŽtection. Ce paramtre peut tre le tableau **Numpy** ou le fichier flux de lÕimage si vous donner la valeur « array » ou « stream » au paramtre **input_type**.

—*paramtre* **output_image_path** (requis seulement si **input_type** = « file ») : Il fait rŽfŽrence au chemin vers le lieu de sauvegarde de lÕimage dŽtectŽe ou dŽtection. Il nÕest requis que si **input_type** = « file ».

– *paramtre* **minimum_percentage_probability** (optionnel) : Ce paramtre est utilisŽ pour dŽterminer lÕintŽgritŽ des rŽsultats de dŽtections. RŽduire cette valeur permettra de dŽtecter plus dÕobjets alors que lÕaugmenter permet dÕavoir des objets dŽtectŽs avec la plus grande prŽcision. La valeur par dŽfaut est 50.

—*paramtre* **output_type** (optionnel) : ce paramtre permet de dŽfinir le format dans lequel lÕimage de dŽtections sera produit. Les valeurs disponibles sont ÔfileÕ(fichier) et ÔarrayÕ(tableau). La valeur par dŽfaut est ÔfileÕ. Si ce paramtre est dŽfinit comme ÔarrayÕ, la fonction va renvoyer un tableau Numpy pour lÕimage de dŽtection. Retrouvez un exemple ci-dessousÊ : :

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

—*paramtre* **display_percentage_probability** (optionnel) : Ce paramtre peut tre utilisŽ pour cacher le pourcentage de probabilitŽ de chaque objet dŽtectŽ dans lÕimage de dŽtectŽe si sa valeur est dŽfinie ^ ÔFalseÕ. La valeur par dŽfaut est ÔTrueÕ.

– *paramtre* **display_object_name** (optionnel) : Ce paramtre peut tre utilisŽ pour cacher le nom de chaque objet dŽtectŽ dans lÕimage de dŽtection si sa valeur est dŽfinie comme ÔFalseÕ. La valeur par dŽfaut est ÔTrueÕ.

—*paramtre* **extract_detected_objects** (optionnel) : ce paramtre peut tre utilisŽ pour extraire et sauvegarder/ retourner chaque objet dŽtectŽ dans une image dans une image sŽparŽe. Sa valeur par dŽfaut est ÔFalseÕ.

– *valeurs renvoyŽes* : Les valeurs renvoyŽes vont dŽpendre des paramtres envoyŽs dans la fonction **detectObjectsFromImage()**. Retrouvez les commentaires et le code ci-dessousÊ :

« »«

Si tous les paramtres nŽcessaires sont dŽfinis et “output_image_path” est dŽfini vers le chemin o le fichier de dŽtection sera sauvegardŽ, la fonction va renvoyerÊ : 1. un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets dŽtectŽs dans lÕimage. Chaque dictionnaire a les propriŽtŽs suivantesÊ :

***Nom (chaine de caractres – string)**
— **percentage_probability (float)**
— box_points (tuple de coordonnŽes x1,y1,x2 et y2)

« »«

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

« »«

Si tous les paramtres requis sont dŽfinis et output_type = “array”, la fonction va renvoyer 1. Un tableau Numpy de lÕimage de dŽtection. 2. Un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets dŽtectŽs dans lÕimage. Chaque dictionnaire contient les propriŽtŽs suivantesÊ : * Nom (string Ð chaine de caractres) * percentage_probability (float) * box_points (tuple de coordonnŽes 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 dŽfini par le chemin vers le lieu de sauvegarde de lÕimage de dŽtection, la fonction renvoieÊ :

1. Un tableau de dictionnaires, chaque dictionnaire correspond aux objets dŽtectŽs dans lÕimage. Chaque dictionnaire contient les propriŽtŽs suivantesÊ : * Nom (string) * percentage_probability (float) * box_points (tuple de coordonnŽes x1,y1,x2 et y2) 2. Un tableau de chaine de caractres reprŽsentant les chemins des images de chaque objet extrait de lÕimage de dŽpart.

« » » 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 renvoyerÊ :

1. Un tableau Numpy de lÕimage de dŽtection.

2. Un tableau de dictionnaires, avec chaque dictionnaire correspondant aux objets dŽtectŽs dans lÕimage. Chaque dictionnaire contient les propriŽtŽs suivantesÊ : * nom(string) * percentage_probability (float) * box_points (tuple de coordonnŽes x1,y1,x2 et y2)

3. Un tableau de tableaux Numpy de chaque objet dŽtectŽ dans lÕimage « » » returned_image, detections, extracted_objects = detector.detectObjectsFromImage(input_image= »image.jpg », output_type= »array », extract_detected_objects=True, minimum_percentage_probability=30)

— **.CustomObjects()** ,CÕest la fonction que vous utiliserez lorsque vous ne voulez faire la dŽtection que dÕun nombre limitŽ dÕobjets. Elle renvoie un dictionnaire dÕobjets et leur valeur ÔTrueÕ ou ÔFalseÕ. Pour dŽtecter les objets sŽlectionnŽs dans une image, vous devrez utiliser les dictionnaires renvoyŽs par cette fonction avec la fonction ** detectCustomObjectsFromImage()** . Retrouvez les dŽtails dans les commentaires et le code ci-dessousÊ :

```
"""
```

Il yÕa 80 possible objets que vous pouvez dŽtecter avec la classe ObjectDetection, vous pouvez les voir ci-dessous.

Person(personne), bicycle(vŽlo), car(voiture), motorcycle(moto), airplane(avion), Bus(bus), train(train), truck(camion), boat(bateau), traffic light(feu de signalisation), fire hydrant (bouche dÕincendie), stop_sign (panneau stop), parking meter (parc mtre), bench (banc), bird (oiseau), cat (chat), dog (chien), horse (cheval), sheep (brebis), cow (vache), elephant (elephant), bear (ours), zebra (zebre), giraffe (girafe), backpack (sac ^ dos), umbrella (parapluie), handbag (sac ^ 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 (gang 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 (cuillre), bowl (bolle), banana (banane), apple (pomme), sandwich (sandwich), orange (orange), broccoli (brocoli), carrot (carotte), hot dog (hot dog), pizza (pizza), donut (beignet), cake(g‰teau), chair (chaise), couch(canape), potted plant(plante a pot), bed(lit), dining table(table de diner), toilet(toilette), tv(tŽlŽvision), laptop(ordinateur), mouse(souris), remote(tŽlŽcommande), keyboard(clavier), cell phone(tŽlŽphone portable), microwave(micro-onde), oven (four), toaster(grille pain), sink(Žvier), refrigerator (rŽfrigŽrateur), book (cahier), clock (horloge), vase(vase) , scissors(ciseaux), teddy bear(ours en peluche), hair dryer(sche cheveux), toothbrush(brosse ^ dent).

Pour dŽtecter uniquement certains des objets ci-dessus, vous devrez instancier la fonction CustomObjects et dŽfinir le ou les noms des objets que vous voulez dŽtecter. Le reste sera dŽfini ^ ÔFalseÕ par dŽfaut. Dans lÕexemple ci-dessous, nous dŽtectons uniquement ÔpersonÕ(personne) et ÔdogÕ(chien).

« » » custom = detector.CustomObjects(person=True, dog=True)

— **.detectCustomObjectsFromImage()**, Cette fonction a tous les paramtres et renvoie toutes les valeurs de la fonction ** detectObjectsFromImage()**, avec une petite diffŽrence. Cette fonction ne fait la dŽtection sur une image que dÕobjets sŽlectionnŽs. Contrairement ^ la fonction ** detectObjectsFromImage()**, elle a besoin dÕun paramtre supplŽmentaire qui est **custom_objet** qui lui rŽcupre le dictionnaire renvoyŽ par la fonction ** CustomObjects()**. Dans lÕexemple ci-dessous, nous avons dŽfini la fonction de dŽtection pour quÕelle ne reconnaisse queÊles personnes et les chiensÊ :

```
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 dŽtection dÕobjets sur Image **

Trouvez ci-dessous un exemple de code pour dŽtecter les objets sur une imageÊ :

```
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)
```

(suite sur la page suivante)

(suite de la page précédente)

```
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 détection sur Vidéo et flux vidéo temp réel Vidéo

ImageAI fournit un ensemble de classes et de fonctions puissantes et facile à utiliser pour faire de la **Détection et du tracking d'objets dans une Vidéo** et L'**analyse vidéo**. **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 détection et analyser des vidéos et des flux vidéo temps réel à partir des cameras IP et de celle de vos appareils. Trouvez ci-dessous les différentes classes et les fonctions respectives mise à votre disposition pour votre utilisation. Ces classes peuvent être intégré 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 **pré-entrainé** à partir de la base de données **COCO**. Les modèles supportes sont **RetinaNet**, **YOLOv3** et **TinyYOLOv3**. Ceci veut dire que vous pouvez détecter et reconnaitre 80 différents types d'objets de la vie de tous les jours dans les vidéos. Pour commencer, télécharger un des modèles pré-entrainé 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 pré-entrainé **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 pré-entrainé **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 pré-entrainé **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 chaine de caractère 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 à partir du chemin que vous avez spécifié dans l'appel ci-dessus de fonction de votre instance de détection d'objets. Trouver un exemple de code ci-dessous :

```
detector.loadModel()
```

– *paramètre* **detection_speed** (optionnel) : Ce paramètre vous permet de réduire de 80% le temps qu'il faut pour détecter les objets sur une vidéo, ce qui conduira à une légère baisse de la précision. Ce paramètre accepte des valeurs de type chaine de caractères. Les valeurs disponibles sont « normal », « fast », « faster », « fastest » et « flash ». La valeur par défaut est 'normal'.

— **.detectObjectsFromVideo()** , Il s'agit de la fonction qui accomplit la détection d'objets sur un fichier vidéo ou un flux vidéo direct après que le modèle ait été chargé dans l'instance que vous avez créé. Retrouvez ci-après 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 référence au chemin vers le fichier vidéo sur lequel vous voulez faire la détection.

—*paramètre* **output_file_path** (requis si **save_detected_video** n'a pas la valeur False) : Il fait référence vers l'emplacement de l'enregistrement du fichier vidéo détecté. Par défaut, cette fonction enregistre les vidéos dans le format **.avi**.

– *paramètre* **frames_per_second** (optionnel, mais recommandé) :

Ce paramètre vous permet de définir le nombre de frames par seconde pour la vidéo détectée qui sera enregistrée. Sa valeur par défaut est 20 mais nous recommandons que vous définissiez la valeur qui sied pour votre vidéo ou vidéo-direct.

– *paramètre* **log_progress** (optionnel) : Définir ce paramètre a 'True' affiche le progrès de la vidéo ou flux direct pendant qu'il est détecter dans la console. Il fera un rapport sur chaque frame détecté pendant qu'il progresse. La valeur par défaut est 'False'

—*paramètre* **return_detected_frame** (optionnel) :

Ce paramètre vous permet de retourner le frame détecté comme tableau Numpy pour chaque frame, seconde et minute de la vidéo détectée. Le tableau Numpy retourné sera envoyé respectivement a **per_frame_function**, **per_second_function** et **per_minute_function** (détails ci-dessous)

– *paramètre* **camera_input** (optionnel) : Ce paramètre peut être assigne en remplacement de **input_file_path** si vous voulez détecter des objets dans le flux vidéo de la caméra. Vous devez instancier la fonction **VideoCapture()** d' OpenCV et de charger l'objet dans ce paramètre.

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)
```

—*paramètre* **minimum_percentage_probability** (optionnel) : Ce paramètre est utilisé pour déterminer l'intégrité des résultats de détection. Diminuer la valeur permet d'afficher plus d'objets alors que l'accroitre permet de s'assurer que les objets détectés ont la précision la plus élevée. La valeur par défaut est 50.

– *paramètre* **display_percentage_probability** (optionnel) : Ce paramètre peut être utilisé pour cacher le pourcentage de probabilité pour chaque objet détecté dans la vidéo détecté si sa valeur est 'False'. La valeur par défaut est 'True'.

—*paramètre* **display_object_name** (optionnel) : Ce paramètre peut être utilise pour cacher le nom de chaque objet détecte dans la vidéo s'il est défini à False. La valeur par défaut est True.

– *paramètre* **save_detected_video** (optionnel) : Ce paramètre peut être utilisé pour sauvegarder ou non la vidéo de détection. Sa valeur par défaut est True.

—*paramètre* **per_frame_function** (optionnel) : Ce paramètre pour permet de transmettre le nom d'une fonction que vous définissez. Puis, pour chaque frame de la vidéo qui est détectée, la fonction sera

définie dans le paramètre qui sera exécuté et la donnée analytique de la vidéo sera transmise à la fonction. Les données renvoyées peuvent être visualisées ou enregistrées dans une base de données NoSQL pour une utilisation et visualisation ultérieure.

Ci-dessous un exemple complet de code :

```
"""
```

Ce paramètre vous permet de définir dans une fonction ou vous voulez faire une exécution chaque fois qu'une image de vidéo est détectée. Si ce paramètre est défini par une fonction, après qu'une image de la vidéo soit détectée, la fonction sera exécutée avec les valeurs suivantes en entrée : * Numéro de position de la frame de la vidéo. * Un tableau de dictionnaires, avec chaque dictionnaire correspondant à chaque objet détecté. 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 présent. * Si return_detected_frame est défini a 'True', le tableau Numpy de la frame détectée sera transmis comme quatrième 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 vidéo est traitée et détectée, la fonction va recevoir et renvoyer les données analytiques pour chaque objet détecté dans la frame de la vidéo 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 récupère les données analytiques et les visualise ; ainsi que la frame détectée 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', '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': 'sandybrown
↪', 'cell phone': 'sienna', 'sink': 'coral', 'bench': 'salmon', 'bottle': 'brown',
↪'car': 'silver', 'bowl': 'maroon', 'tennis racket': 'palevilotered', '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': 'sapgreen',
↪'book': 'lawngreen', 'potted plant': 'greenyellow', 'toaster': 'ivory', 'stop sign
↪': 'beige', 'couch': 'khaki'}


    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
```

(suite sur la page suivante)

(suite de la page précédente)

```
            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)


-- *paramètre* **per_second_function** (optionnel) : Ce paramètre vous permet de␣
↪transmettre le nom d'une fonction que vous définissez. Puis, pour chaque seconde de␣
↪la vidéo qui est détectée, la fonction sera transmise dans le paramètre sera␣
↪exécutée et les données analytiques de la vidéo seront envoyées dans la fonction.␣
↪Les données renvoyées peuvent être visualisées ou sauvegardées dans une base de␣
↪données NoSQL pour une utilisation et visualisation future.

   Ci-dessous un exemple complet de code ::

      """
        Ce paramètre vous permet de transmettre dans une fonction où vous voulez␣
↪faire une exécuter après que chaque seconde de la vidéo soit détectée.  Si le␣
↪paramètre est défini comme une fonction, seconde après que le vidéo soit détecté,␣
↪la fonction sera exécutée avec les valeurs suivantes en argument :

      -- numéro de position du second
      -- un tableau de dictionnaire dont les clefs sont les numéros de position de␣
↪chaque frame présente à la dernière seconde, et la valeur de chaque clef est le␣
↪tableau de chaque frame qui contient les dictionnaires de chaque objet détecté dans␣
↪la frame.



      -- Un tableau de dictionnaires, avec chaque dictionnaire à chaque frame de la␣
↪seconde précédente, et les clés pour chaque dictionnaire sont les noms de numéros␣
↪d'objets uniques détectés dans chaque frame, et les clés sont le nombre d'instances␣
↪des objets trouvés dans le frame.



      -- Un dictionnaire avec pour clé le nom de chaque unique objet détecté dans␣
↪toutes les secondes passées, at les valeurs clés sont les moyennes d'instances␣
↪d'objets trouvés dans toutes les frames contenus dans les secondes passées.
```

(suite sur la page suivante)

(suite de la page précédente)

```
      -- Si return_detected_frame est défini aa 'True', le tableau Numpy du frame de␣
↪détection sera envoyé comme cinquième paramètre 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)
```

Dans l'exemple ci-dessus, chaque seconde dans la vidéo est traitée et détectée, la fonction va recevoir et renvoyer les données analytiques des objets détectés dans la vidéo 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}],
```

(suite sur la page suivante)

(suite de la page précédente)

```
    [{'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}]
    ]

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 détecté à 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”, “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” : “sandybrown”, “cell phone” : “sienna”, “sink” : “coral”, “bench” : “salmon”, “bottle” : “brown”, “car” : “silver”, “bowl” : “maroon”, “tennis racket” : “palevilotered”, “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” : “sapgreen”, “book” : “lawngreen”, “potted 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)

—*paramètre* **per_minute_function** (optionnel) : Ce paramètre peut être transmis en argument du nom de la fonction que vous définissez. Puis, pour chaque frame de la vidéo qui est détectée, le paramètre qui a été transmis dans la fonction sera interprété et les données analytiques de la vidéo seront transmis à la fonction. Les données retournées sont de même nature que **per_second_function** ; la différence est qu’elle ne tient compte que de tous les frames de la dernière minute de la vidéo.

Retrouvez une exemple de fonction pour ces paramètres 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 ———— »)

—*paramètre* **video_complete_function** (optionnel) : Ce paramètre peut être transmis en argument d'une fonction que vous définissez. Une fois que tous les frames de la vidéo sont totalement détectés, le paramètre transmis sera interprété et les données analytiques de la vidéo seront transmis à la fonction. Les données retournées ont la même nature que **per_second_function** et **per_minute_function** ; les différences sont qu'aucun index n'est renvoyé et ici tous les frames de la vidéo sont couvertes.

Retrouvez une exemple de fonction pour ces paramètres 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 personnalisŽ et prŽdiction des Classes

ImageAI fournit des classes puissante et nŽanmoins facile ^ utiliser pour entrainer des algorithmes a la pointe de la technologies tel que **SqueezeNet**, **ResNet** , **InceptionV3** et **DenseNet** sur votre propre base de donnŽes avec juste **5 lignes de code** pour gŽnŽrer votre propre modle personnalisŽ. Une fois que vous avez entrainŽ votre propre modle, vous pouvez utiliser la classe **CustomImagePrediction** fournit par **ImageAI** pour utiliser votre modle pour reconnaitre et faire la dŽtection sur une image ou un ensemble dÕimages.

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

La classe **ModelTraining** vous permet dÕentrainer un des quatre algorithme dÕapprentissage profond suivant (**SqueezeNet**, **ResNet** , **InceptionV3** et **DenseNet**) sur votre base de donnŽes dÕimages pour gŽnŽrer votre propre modle. Votre base de donnŽes dÕimages doit contenir au moins deux diffŽrentes classes/types dÕimages (chat et chien) et vous devez ressembler au moins 500 images de chaque classe pour obtenir le maximum de prŽcision possible.

Le processus dÕentrainement gŽnŽrŽ un fichier JSON qui fait une correspondance entre les types dÕobjets dans votre base dÕimages et crŽŽ des modles. Vous ferez le choix du modle avec la prŽcision la plus ŽlevŽe et qui puisse faire la prŽdiction en utilisant le modle et le fichier JSON gŽnŽrŽ.

Puisse la t‰che dÕapprentissage est gourmande en ressource, nous recommandons fortement de la faire ^ lÕaide dÕun ordinateur ŽquipŽ dÕun GPU NVIDIA et ayant le version GPU de Tensorflow installŽe. Faire lÕapprentissage sur un CPU va demander beaucoup dÕheures et de jours. Avec un systme informatique ŽquipŽ dÕun GPU NVIDIA cela ne devrait prendre que quelques heures. Vous pouvez utiliser Google Colab pour cette expŽrience, puisquÕil est ŽquipŽ dÕun GPU NVIDIA K80.

Pour entrainer votre modle de prŽdiction, vous devez prŽparer les images que vous voulez utiliser pour entrainer votre modle. Vous prŽparerez les images comme suitÊ :

– CrŽer un dossier avec le nom que vous aimeriez donner avec votre base de donnŽes (ex : Chats)

—Dans le dossier que vous avez prŽcŽdemment crŽe, crŽer un dossier que vous nommerez ÔtrainÕ

– A cote du dossier train, crŽer un autre dossier et nommez le ÔtestÕ

—Dans le dossier ÔtrainÕ, crŽez un dossier pour chaque type dÕobjets que vous aimeriez que votre modle reconnaisse et nomme le dossier selon la classe ^ prŽdire (exÊ : chien, chat, Žcureuil, serpents)

– Dans chaque dossier prŽsent dans votre dossier ÔtrainÕ, mettez-y les images de chaque objet ou classe. Ces images seront utilisŽes pour lÕapprentissage de votre modle.

—pour gŽnŽrer un modle qui puisse tre viable pour des applications robustes, Je vous recommande dÕavoir au moins 500 images ou de plus par objets. 1000 images per objets serait mieux.

– Dans le dossier ÔTestÕ, crŽer des dossiers et nommez les selon les noms que vous avez utilisŽ pour le dossier ÔTrainÕ. Mettez environ 100 ^ 200 images correspondantes dans chaque dossier. Ces images seront celle utilisŽes pour tester le modle aprs lÕavoir entrainŽ.

—Une fois que vous avez fait cela, la structure des dossiers de votre base dÕimages devrait tre 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 donnŽes est prte, vous pouvez crŽer une instance de la classe **ModelTraining**. Retrouver un exemple ci-dessousÊ :

```
from imageai.Prediction.Custom import ModelTraining

model_trainer = ModelTraining()
```

Une fois que vous avez crŽŽ lÕinstance ci-dessous, vous pouvez utiliser les fonctions ci-dessous pour commencer le processus dÕentrainement.

— **.setModelTypeAsSqueezeNet()** , Cette fonction Žtablit comme type de modle pour votre instance dÕentrainement le modle **SqueezeNet**, ceci veut dire que lÕalgorithme **SqueezeNet** sera utilisŽ pour entrainer votre modle. Trouver un exemple de code ci-dessousÊ :

```
model_trainer.setModelTypeAsSqueezeNet()
```

— **.setModelTypeAsResNet()** , Cette fonction Žtablit comme type de modle pour votre instance dÕentrainement le modle **ResNet**, ceci veut dire que lÕalgorithme **ResNet** sera utilisŽ pour entrainer votre modle. Trouver un exemple de code ci-dessous

```
model_trainer.setModelTypeAsResNet()
```

— **.setModelTypeAsInceptionV3()** , Cette fonction Žtablit comme type de modle pour votre instance dÕentrainement le modle ** InceptionV3**, ceci veut dire que lÕalgorithme ** InceptionV3** sera utilisŽ pour entrainer votre modle. Trouver un exemple de code ci-dessousÊ :

```
model_trainer.setModelTypeAsInceptionV3()
```

— **.setModelTypeAsDenseNet()** , Cette fonction Žtablit comme type de modle pour votre instance dÕentrainement le modle **DenseNet**, ceci veut dire que lÕalgorithme **DenseNet** sera utilisŽ pour entrainer votre modle. Trouver un exemple de code ci-dessousÊ :

```
model_trainer.setModelTypeAsDenseNet()
```

— **.setDataDirectory()** , Cette fonction prend en argument une chaine de caractre qui doit tre le chemin vers le dossier qui contient les sous-dossiers **test** et **train** qui contiennent votre base dÕimages. Retrouver un exemple dÕutilisation de la fonction, et de ses paramtres ci-dessous

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

– *parametre* **data_directory** (obligatoire) : Il sÕagit du chemin vers le dossier qui contient votre base dÕimages.

— **.trainModel()** , Il sÕagit de la fonction qui commence le processus dÕentrainement. Une fois commencŽ, il crŽera un fichier JSON dans le dossier **dataset/json** (ex **animaux/json**) qui contient la correspondance de chaque classe dans la base dÕimages. Le fichier JSON sera utilisŽ pendant la dŽtection personnalisŽe pour produire les rŽsultats. 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)
```

– *paramtre* **num_objects** (obligatoire) : Ceci fait rŽfŽrence au nombre de diffŽrentes classes dans votre base dÕimages.

—*paramtre* **num_experiments** (obligatoire) : Il reprŽsente le nombre de fois que lÕalgorithme sera entrainŽ sur la base dÕimages. La prŽcision de votre entrainement augmente avec le nombre dÕitŽrations ou dÕentrainement. Cependant la prŽcision atteint son maximum avec un certain nombre dÕitŽration et nombre dŽpend de la taille et de la nature de base de donnŽes.

– *paramtre* **enhance_data** (optionnel) : Ce paramtre est utilisŽ pour transformer votre base dÕimages en gŽnŽrant plus dÕŽchantillons pour la phase dÕentrainement. Par dŽfaut sa valeur est ÔFalseÕ. NŽanmoins, il est important de lui donner la valeur ÔTrueÕ lorsque votre base dÕimages contient moins de 1000 images par classe.

—*paramtre* **batch_size** (optionnel) : Pendant la phase dÕentrainement, LÕalgorithme est entrainŽ sur un ensemble dÕimages en parallle. A cause de cela, la valeur par dŽfaut est mise ^ 32. Vous pouvez accroitre ou dŽcroitre cette valeur selon votre connaissance du systme que vous utilisez pour lÕapprentissage. Si vous envisagez de changer cette valeur, vous devrez utiliser des multiples de 8 pour optimiser le processus dÕapprentissage.

– *paramtre* **show_network_summary** (optionnel) : Lorsque ce paramtre a la valeur ÔTrueÕ, il affiche la structure de lÕalgorithme que vous utilisez pour lÕapprentissage sur vos images dans une petite console avant de commencer lÕapprentissage. Sa valeur par dŽfaut est ÔFalseÕ.

—*paramtre* **initial_learning_rate** (optionnel) : Ce paramtre a une haute valeur technique. Il dŽtermine et contr™le le comportement de votre apprentissage, ce qui est critique pour la prŽcision ^ rŽaliser. Vous pouvez changer la valeur de ce paramtre si vous avez une pleine comprŽhension de sa fonctionnalitŽ.

– *training_image_size* **initial_learning_rate** (optionnel) : Il reprŽsente la taille que vos images prendront pendant le processus dÕapprentissage, peu importe leur taille dÕorigine. La valeur par dŽfaut est de 224 et elle ne doit pas aller en dessous de 100. Augmenter sa valeur permettra de gagner en prŽcision mais augmentera aussi le temps dÕapprentissage et vice-versa.

Exemple de code pour un model apprentissage personnalise

Trouvez ci-dessous un exemple de code lors de lÕapprentissage dÕun modle personnalisŽ sur votre base dÕimagesÊ :

```
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 aperu de rŽsultat lorsque lÕapprentissage commenceÊ :

```
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

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

Expliquons les dŽtails ci-dessusÊ :

1. La ligne Epoch 1/100 signifie que le rŽseau fait le premier apprentissage sur les 100 voulus. 0
2. La ligne 1/25 [>.....] - ETA : 52s - loss : 2.3026 - acc : 0.2500 reprŽsente le nombre de groupe qui ont ŽtŽ entrainŽ dans la prŽsente phase dÕapprentissage.
3. La ligne Epoch 00000 : sauvegarde le modle ^ lÕemplacement C :UsersUserPycharmProjectsImageAITestpetsmodelsmodelex-000acc-0.100000.h5 ^ la fin de la phase dÕapprentissage prŽsente. ex_000 reprŽsente le niveau dÕapprentissage tandis que acc0.100000 et valacc : 0.1000 reprŽsente la prŽcision du modle sur lÕensemble dÕimages ÔTestÕ aprs le prŽsent apprentissage(La valeur maximale de la prŽcision est de 1.0). Ce rŽsultat vous permet de connaitre le meilleur modle a utiliser pour la dŽtection sur vos images.

Une fois que vous avez terminŽ lÕapprentissage de votre modle termine, vous pouvez utiliser la classe **CustomImagePrediction** dŽcrite si dessous pour la dŽtection avec votre modle.

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

Cette classe peut tre considŽrŽe comme une rŽplique de **imageai.Prediction.ImagePrediction** puis quÕelle a les mme fonctions, paramtres et rŽsultats. La seule diffŽrence est que cette classe fonctionne avec votre modle personnalisŽ. Vous aurez besoin de spŽcifier le chemin du fichier JSON gŽnŽrŽ pendant la phase dÕapprentissage et aussi de spŽcifier le nombre de classe dans votre base dÕimage lors du chargement du modle. Ci-dessous est un exemple de crŽation dÕinstance de la classeÊ :

```
from imageai.Prediction.Custom import CustomImagePrediction

prediction = CustomImagePrediction()
```

Une fois que vous avez crŽŽ lÕinstance, vous pouvez utiliser les fonctions ci-dessous pour configurer les propriŽtŽs de votre instance et commencer le processus de dŽtection et reconnaissance sur des images.

— **.setModelTypeAsSqueezeNet()** , Cette fonction Ždtablit comme type de modle pour votre instance de reconnaissance et dŽtection, le modle **SqueezeNet**, ceci veut dire que lÕalgorithme **SqueezeNet** gŽnŽrŽ pendant votre phase dÕapprentissage personnalisŽe sera utilisŽ pour la tache de prŽdiction sur vos images. Trouver un exemple de code ci-dessousÊ :

```
prediction.setModelTypeAsSqueezeNet()
```

— **. setModelTypeAsResNet()** , Cette fonction Ždtablit comme type de modle pour votre instance de reconnaissance et dŽtection, le modle ** ResNet**, ceci veut dire que lÕalgorithme **ResNet** gŽnŽrŽ pendant votre phase dÕapprentissage personnalisŽe sera utilisŽ pour la tache de prŽdiction sur vos images. Trouver un exemple de code ci-dessousÊ :

```
prediction.setModelTypeAsResNet()
```

— **. setModelTypeAsInceptionV3 ()** , Cette fonction Ždtablit comme type de modle pour votre instance de reconnaissance et dŽtection, le modle **InceptionV3**, ceci veut dire que lÕalgorithme **InceptionV3** gŽnŽrŽ pendant votre phase dÕapprentissage personnalisŽe sera utilisŽ pour la tache de prŽdiction sur vos images. Trouver un exemple de code ci-dessousÊ :

```
prediction.setModelTypeAsInceptionV3()
```

— **. setModelTypeAsDenseNet()** , Cette fonction Ždtablit comme type de modle pour votre instance de reconnaissance et dŽtection, le modle ** DenseNet**, ceci veut dire que lÕalgorithme **DenseNet** gŽnŽrŽ pendant votre phase dÕapprentissage personnalisŽe sera utilisŽ pour la tache de prŽdiction sur vos images. Trouver un exemple de code ci-dessousÊ :

```
prediction.setModelTypeAsDenseNet()
```

— **.setModelPath()** , cette fonction accepte une chaine de caractre qui doit tre le chemin vers le fichier modle gŽnŽrŽ pendant votre phase dÕapprentissage et doit correspondre au type de modle que vous avez dŽfini pour votre instance de reconnaissance dÕimages. Trouver un exemple de code, et de paramtres de fonction ci-dessousÊ :

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

– *paramtre* **model_path** (requis) : Il sÕagit du chemin vers le fichier modle tŽlŽchargŽ.

— **.setJsonPath()** , cette fonction prend en argument une chaine de caractre qui reprŽsente le chemin vers le fichier JSON gŽnŽrŽ pendant la phase dÕapprentissage du modle personnalisŽ. Trouvez ci-dessous un exemple de code et de paramtres de la fonctionÊ :

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

– *paramtre* **model_path** (requis) : Il sÕagit du chemin vers le fichier modle tŽlŽchargŽ.

— **.loadModel()** , Cette fonction charge le modle ^ partir du chemin spŽcifiŽ dans votre appel de fonction ci-dessus pour votre instance de prŽdiction dÕimages. Au paramtre **num_objects** vous devrez donner la valeur correspondant au nombre de classes dans votre base dÕimages. Trouvez ci-dessous un exemple de code et de paramtres de la fonctionÊ :

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

– *paramtre* **num_objects** (requis) : La valeur de ce paramtre doit correspondre au nombre de classe dans votre base dÕimages.

– *paramtre* **prediction_speed** (optionnel) : Ce paramtre vous permet de rŽduire le temps de prŽdiction sur une image dÕenviron 80% ce qui conduit ^ une lŽgre rŽduction de la prŽcision. Ce paramtre prend des valeurs de type chaine de caractre. Les valeurs disponibles sontÊ : « normal », « fast », « faster » et « fastest ». La valeur par dŽfaut est « normal »

— **.predictImage()** , CÕest la fonction qui accomplit ^ proprement parler la prŽdiction sur une image. Elle peut tre appeler plusieurs fois sur plusieurs images une fois que le modle a ŽtŽ charge dans lÕinstance de prŽdiction. Trouver ci-dessous un exemple de code, de paramtres de fonction ainsi que les valeurs renvoyŽesÊ :

```
predictions, probabilities = prediction.predictImage("image1.jpg", result_
↪count=2)
```

– *paramtre* **image_input** (requis) : Il fait rŽfŽrence au chemin vers votre image, le tableau de type Numpy de votre image ou le flux de votre image, dŽpendamment de type de valeur dÕentrŽe spŽcifiŽe.

—*paramtre* **result_count** (optionnel) : il fait rŽfŽrence au nombre possible de prŽdiction qui peuvent tre donne. Ce paramtre a pour valeur par dŽfaut 5.

– *paramtre* **input_type** (optionnel) : Il fait rŽfŽrence au type de la valeur dÕentrŽe que vous passez au paramtre **image_input**. Il est de type ÔfileÕ par dŽfaut et accepte Ôstream Õet Ôarray Õaussi.

—*valeur retournŽe* **prediction_results** (une liste python) :

La premire valeur retournŽe par la fonction **predictImage** est une liste qui contient tous les rŽsultats possibles de prŽdiction. Les rŽsultats sont ordonnŽs en ordre descendant de pourcentage de probabilitŽ.

– *valeur retournŽe* **prediction_probabilities** (une liste python) :

La premire valeur retournŽe par la fonction **predictImage** est une liste qui contient les pourcentages de probabilitŽ correspondantes a toutes les prŽdictions possibles dans **prediction_results**

— **.predictMultipleImages()** , Cette fonction peut tre utilisŽe pour effectuer la tache de prŽdiction sur 2 ou plusieurs images en une seule fois. Trouvez ci-dessous un exemple de code, paramtres de fonction et de valeurs renvoyŽesÊ :

```
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("----------------------")
```

– *paramtre* **sent_images_array** (requis) : Il fait rŽfŽrence a une liste qui contient le chemin vers vos fichiers image, vos tableau Numpy de vos images ou vos fichiers de flux dÕimages, dŽpendamment du type de valeur dÕentrŽe spŽcifiŽe.

– *paramtre* **result_count_per_image** (optionnel) : Il fait rŽfŽrence au nombre possible de prŽdictions qui doivent tre donnŽes pour chaque image. Ce paramtre a pour valeur par dŽfaut 2.

– *paramtre* **input_type** (optionnel) : Il fait rŽfŽrence au format de vos images ont dans la liste du paramtre **sent_images_array**. Il est de type ÔfileÕ par dŽfaut et accepte Ôstream Õet Ôarray Õaussi.

– *valeur retournŽe* **output_array** (une liste python) :

La valeur retournŽe par la fonction **predictMultipleImages** est une liste qui contient des dictionnaires. Chaque dictionnaire correspond ^ une image contenue dans le tableau envoyŽe ^ **sent_images_array**. Chaque dictionnaire a une propriete « prediction_results » qui est une liste de tous les resultats de prediction pour lÕimage a cet indice aussi bien que la probabilite de prediction « prediction_probabilities » qui est une liste de pourcentage de probabilite correspondant a chaque resultat.

Exemple de code

Trouvez ci-dessous un Žchantillon de code pour la prŽdiction personnalisŽÊ :

```
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)
```

CHAPITRE 2

Indices and tables

— genindex
— modindex
— search