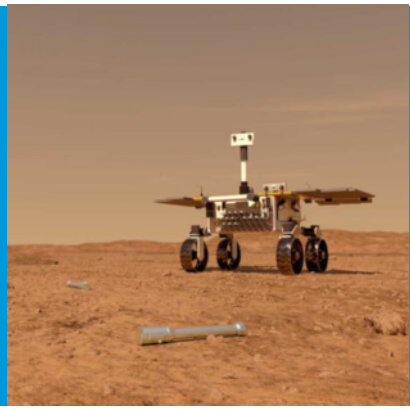
Belgium esero

Fiche pour l'enseignant

CONTRÔLE INTELLIGENT DU ROVER

Phase 5 / Déplacement du rover avec Python en utilisant la reconnaissance d'images.



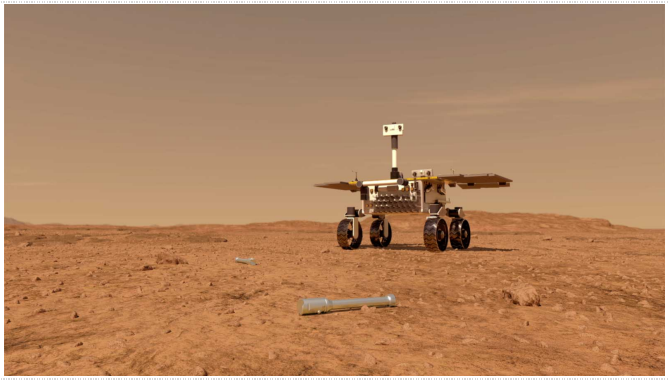
Après avoir construit et entraîné le FetchBot avec les élèves, cette activité montre comment le déplacer avec Python en utilisant la reconnaissance d'images.

Prérequis :

1. Reconnaissance d'images avec Python
(voir Activité 3 - Reconnaissance d'images avec Python)
2. Avoir pris en main le Raspberry Pi, et savoir s'y connecter avec VNC viewer
(voir Raspberry Pi: Prise en main et préparation)
3. Avoir construit le FetchBot
(voir Activité 4 - Construction du rover)
4. Avoir programmé le FetchBot pour le faire se déplacer
(voir Activité 6 - Contrôle du FetchBot avec Python)

esa belspo La Scientothèque L'ÉGALITÉ DES CHANCES PAR LES SCIENCES

Intelligence Artificielle



FetchBot : Rover martien intelligent

Age : 14-18 ans

Contrôle du FetchBot et reconnaissance d'images avec Python¶

Objectif :

- ✓ Faire déplacer le FetchBot avec Python en utilisant la reconnaissance d'images.

Notions abordées : Intelligence artificielle, classification d'images, robotique, programmation avec Scratch, condition, boucle.

- Phase 1 : Aperçu
- Phase 2 : Préparation
- Phase 3 : Création du modèle de reconnaissance d'images
- Phase 4 : Faire dire ce que le modèle reconnaît
- Phase 5 : Contrôle du rover avec la reconnaissance d'images

Durée : 4h00

Dispositif pédagogique : par groupe de deux.

Matériel

- Un FetchBot et un laptop/tablette par groupe de 2, avec connexion à Internet

Droits d'auteur

Description détaillée

Un protocole à destination des jeunes est disponible en ligne. L'activité peut donc être réalisée en autonomie par les élèves.

Phase 1 : Aperçu

Dans cette activité, tu vas programmer le FetchBot pour le faire explorer un terrain et retrouver un objet. On utilisera comme exemple ici la recherche d'un tube sur un sol martien, comme ce que devra faire le rover Fetch sur mars. Tu utiliseras la reconnaissance d'images pour contrôler ce que fais le rover: Faire avancer le robot lorsque le terrain est dégagé, le faire tourner lorsqu'il rencontre un obstacle, et le faire s'arrêter lorsqu'il trouve le tube.

L'activité se compose de trois parties principales:

- Créer un modèle de reconnaissance d'images capable de détecter 3 types de classes: le sol dégagé, les obstacles, et les tubes
- Utiliser le modèle de reconnaissance d'images dans un programme Scratch pour faire dire au sprite de rover la classe détectée (comme dans l'activité 2)
- Rajouter à ce programme Scratch l'envoi de commandes au rover en fonction de la classe détectée.

La vidéo de ce lien te donne un aperçu de la réalisation de ces activités.

Note : L'activité peut être adaptée pour la recherche d'autres objets (crayon, pièce de légo, …) et sur d'autres types de terrains, par exemple le sol de ta classe, ou même un terrain en extérieur.

Phase 2 : Préparation

Ton rover est construit, et tu y es connecté avec VNC viewer depuis un ordinateur ou une tablette.

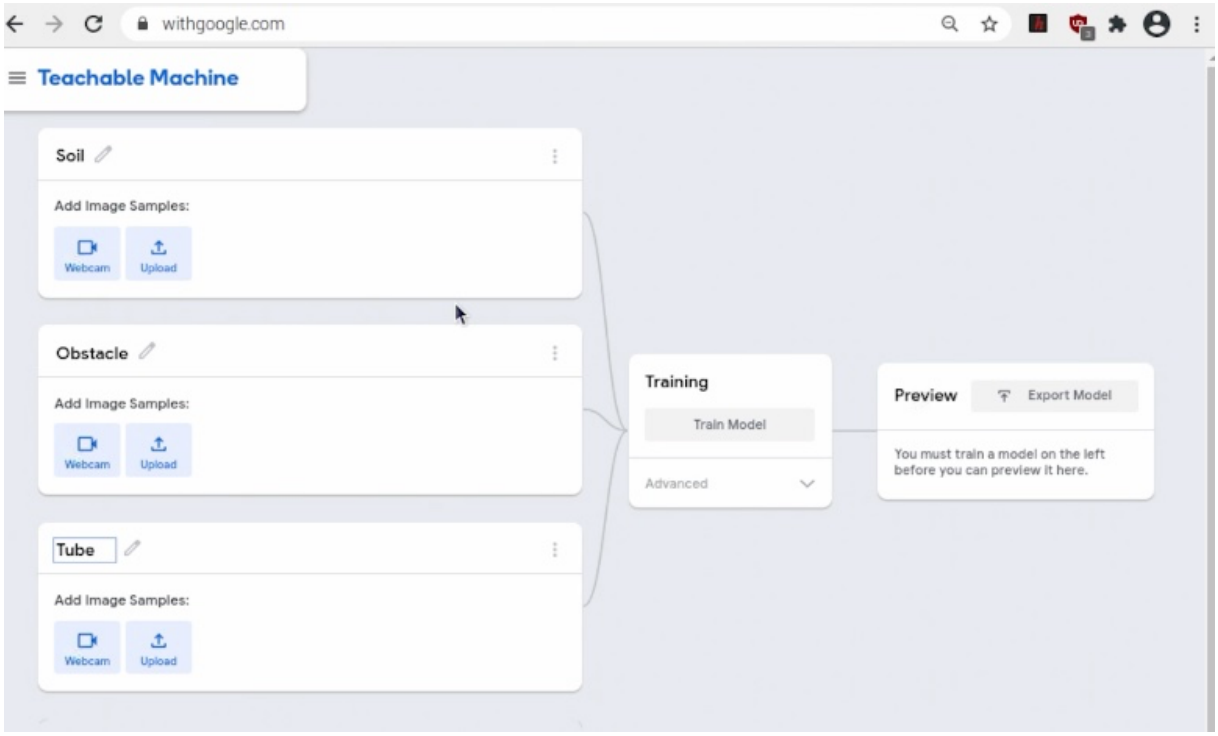
Il te faudra ensuite choisir un objet que le rover devra retrouver, et délimiter une zone dans laquelle le rover devra rechercher cet objet. Pour l'objet, tu peux choisir un tube, un crayon, une pièce de légo, ou autre. Pour la zone de recherche, tu peux imprimer un sol martien au format A0 ou sous forme de tapis, ou sinon délimiter une zone avec du ruban adhésif. Tu pourras ajouter dans la zone des obstacles à éviter (cailloux, boites, ou autre).

Phase 3 : Création du modèle de reconnaissance d'images

La création du modèle suit le même principe que celui décrit dans l'activité 1 - Reconnaissance d'images avec la Teachable Machine. Il te faudra ici prendre des exemples d'images pour trois classes:

- le sol dégagé: classe 'sol'. Les exemples peuvent inclure ici toute partie de la zone où le robot peut avancer, c'est à dire qui ne contient ni obstacle, ni tube.
- les obstacles: classe 'obstacle'. Les exemples peuvent inclure ici toute partie de la zone où le robot est face à un obstacle, comme un bord, un autre robot, ou des objets à éviter comme des cailloux.
- le tube à retrouver: classe 'tube'. Les exemples peuvent inlcure ici toute partie de la zone où le robot est devant un tube.

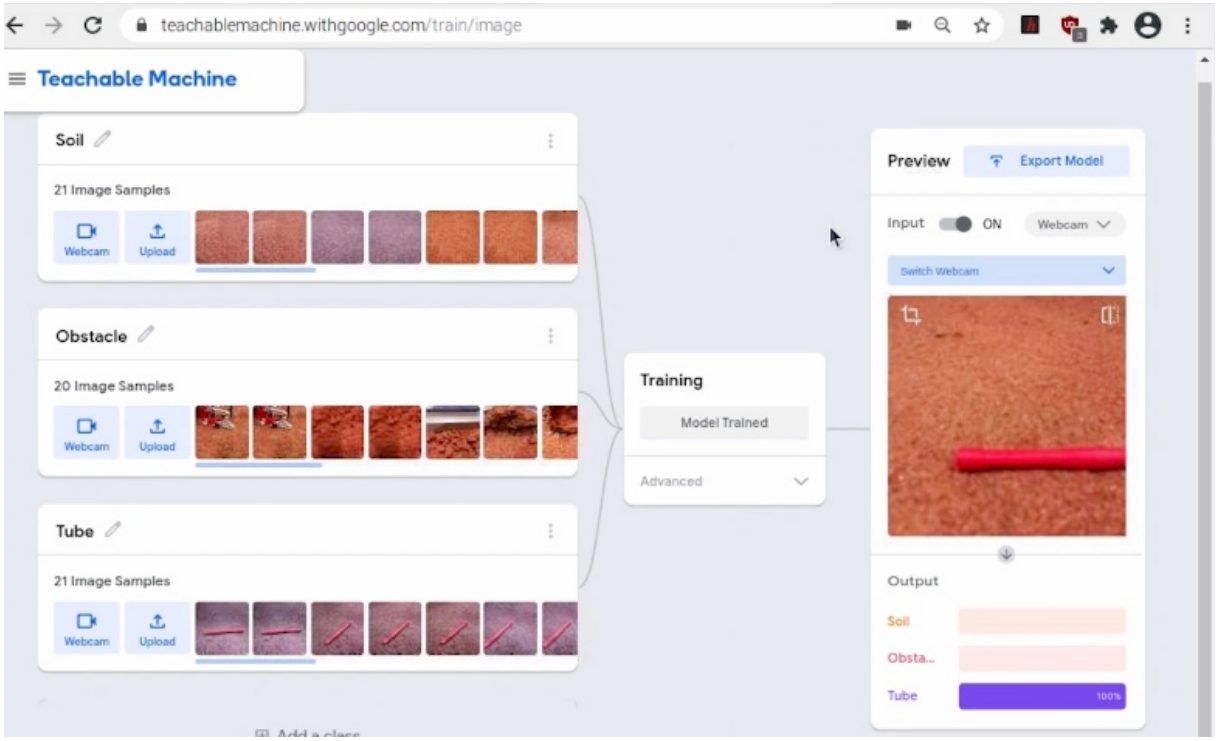
Ouvre un projet d'images dans la Teachable Machine, et crée trois catégories 'Sol', 'Obstacle', et 'Tube'.



Pour chaque classe, il est en général suffisant de prendre une vingtaine d'exemples. Essaye de varier au maximum les exemples que tu prends (angle ou distance par rapport à un obstacle ou un tube, luminosité, reflet sur le sol, etc...).

La vidéo ici (partie 'Training', de 1'30 à 4'50) te montre comment faire cela en pratique.

Après avoir pris une vingtaine d'exemples pour chaque classe, entraîne le modèle. Tu peux ensuite tester si le modèle reconnaît correctement les trois classes. L'interface devrait de la Teachable Machine devrait ressembler à ceci :

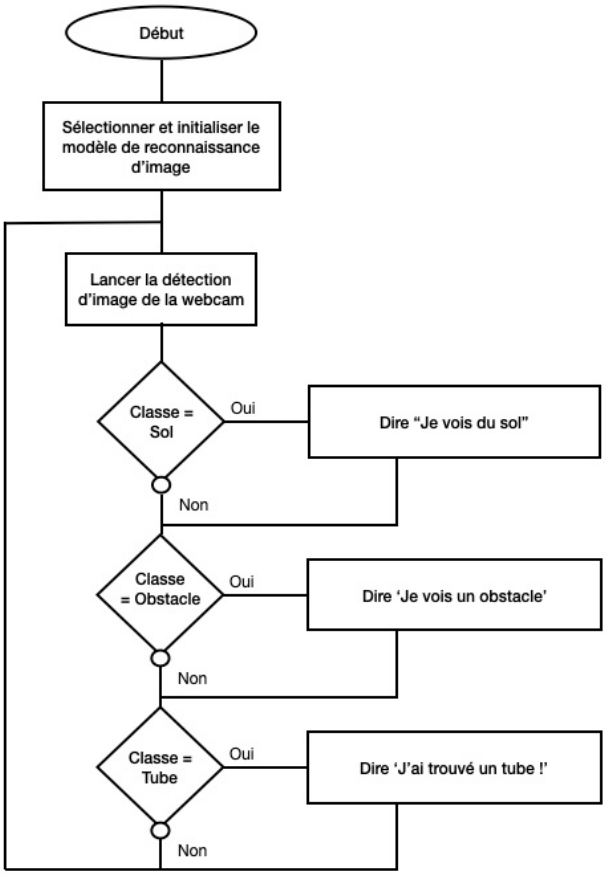


Ici, un tube est devant la caméra, et le modèle reconnaît correctement (classe 'Tube' détectée à 100%).

Note : Il est plus facile d'être deux pour prendre les exemples d'images des trois classes: un pour déplacer le rover dans la zone, et l'autre pour prendre les photos sur la Teachable Machine.

Phase 4 : Faire dire ce que le modèle reconnaît¶

Tu peux maintenant utiliser le modèle dans Python et faire dire ce qu'il reconnaît. Le principe est le même que celui décrit dans l'activité 3. Tu as ici trois classes, donc une troisième condition à ajouter dans le logigramme:



Et voici la traduction en langage Python:

```
# Necessary import
import cv2 # cv2 is used to take image from the camera
import myfunctions # myfunctions helps for taking picture and making
predictions
import matplotlib.pyplot as plt # matplotlib is used to deal with images
from PIL import Image # PIL is used to deal with images
import time # time is used for making the computer wait

# Get camera object
camera_object = cv2.VideoCapture(0)
camera_object.set(cv2.CAP_PROP_BUFFERSIZE, 3)

# Initialize model
interpreter = myfunctions.initialize_model(model_path='model.tflite')

# Infinite loop
while True:

    # Wait for one second
    time.sleep(1)

    # Take image from the camera
    picture_rgb = myfunctions.take_picture(camera_object)
```

```
    # Predict image class
    prediction, probability = myfunctions.model_prediction(interpreter,
picture_rgb)

    # If prediction is class 0, class is 'Soil'
    if prediction == 0:

        print("Je vois du sol")

    # If prediction is class 1, class is 'Obstacle'
    if prediction == 1:

        print("Je vois un obstacle")

        # If prediction is class 2, class is 'Tube'
    if prediction == 2:

        print("J'ai trouvé un tube!")
```

Note : Le fichier model.tflite est celui que tu as exporté de la Teachable Machine comme dans l'Activité 3 - Reconnaissance d'images avec Python

Le programme se trouve dans le répertoire 3_Rover_Camera_Control/FetchBot_Control_IR_Python/ et s'appelle 1-predict-image.py.

Phase 5 : Contrôle du rover avec la reconnaissance d'images

Il ne te reste plus qu'à programmer comment contrôler rover en fonction de que la caméra détecte. Modifie le programme pour :

- Faire avancer le rover si la classe détectée est le sol
- Faire tourner le rover à gauche si la classe détectée est un obstacle.

Quel est le logigramme de ce programme?

```
# Necessary import
import cv2 # cv2 is used to take image from the camera
import myfunctions # myfunctions helps for taking picture and making
predictions
import matplotlib.pyplot as plt # matplotlib is used to deal with images
from PIL import Image # PIL is used to deal with images
import time # time is used for making the computer wait
import motor_control

# Get camera object
camera_object = cv2.VideoCapture(0)
camera_object.set(cv2.CAP_PROP_BUFFERSIZE, 3)

# Initialize model
interpreter = myfunctions.initialize_model(model_path='model.tflite')

# Infinite loop
while True:

    # Wait for one second
    time.sleep(1)

    # Take image from the camera
    picture_rgb = myfunctions.take_picture(camera_object)

    # Predict image class
    prediction, probability = myfunctions.model_prediction(interpreter,
picture_rgb)

    # If prediction is class 0, class is 'Soil'
    if prediction == 0:

        print("Je vois du sol")
        motor_control.turn_left(speed=0.5, duration=1)

    # If prediction is class 1, class is 'Obstacle'
    if prediction == 1:

        print("Je vois un obstacle")
        # Turn left
        motor_control.turn_left(speed=0.5, duration=1)

        # If prediction is class 2, class is 'Tube'
    if prediction == 2:

        print("J'ai trouvé un tube!")
```

Le programme se trouve dans le répertoire
`3_Rover_Camera_Control/FetchBot_Control_IR_Python/`
et s'appelle `2-predict-image-control.py`.