Projets

Classification des images avec Google Coral

Apprends comment réentraîner un modèle d'apprentissage automatique et l'utiliser pour l'identification d'images.

Étape 1 Introduction

Dans ce projet, tu vas réentraîner un modèle d'apprentissage automatique à l'aide d'une unité de traitement de tenseur (TPU, Tensor Processing Unit) Coral, puis l'utiliser pour reconnaître des images.

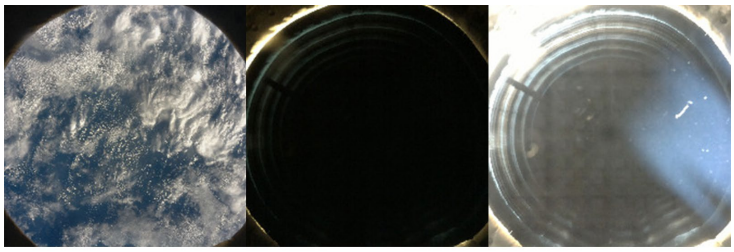
L'apprentissage automatique est une forme d'Intelligence artificielle. Dans la classification des images, un modèle est entraîné pour reconnaître différentes classes d'images.

Tu vas :

- Réentraîner un modèle d'apprentissage automatique pour reconnaître de nouvelles classes d'images
- Utiliser une unité de traitement de tenseur (TPU) Coral pour classer les images
- Avoir l'opportunité de participer au concours Mission Space Lab (https://astro-pi.org/mission-space-lab/) et d'exécuter des algorithmes d'apprentissage automatique sur la Station spatiale internationale.

Étape 2 Configurer ton Coral et rassembler tes images

De nombreux modèles d'apprentissage automatique ont été entraînés pour identifier différentes classes d'images. Ils peuvent facilement être réentraînés pour identifier de nouvelles classes.



Jour Nuit Crépusc

{:width="300px"}
Sur ton Raspberry Pi, tu devras installer le logiciel nécessaire pour utiliser ton TPU Coral avec Python.

Avec ta configuration Raspberry Pi (https://esero.fr/wp-content/uploads/2021/04/Premiers-pas-avec-le-Raspberry-Pi-_-Raspberry-Pi-Projects_2.pdf)
et ton TPU Coral déconnecté, tu peux ouvrir un terminal et exécuter la commande suivante.
Ceci permettra d'installer la bibliothèque Python de Coral et ses dépendances.

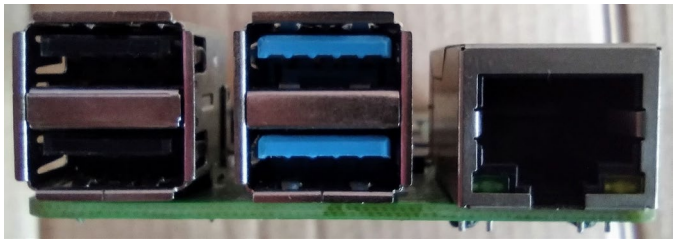
```
wget -O - https://raw.githubusercontent.com/raspberrypilearning/image-id-coral/master/en/resources/install_script.sh
| bash
```

Qu'y a-t-il dans ce script ?

Si tu souhaites installer manuellement le logiciel, tu peux le faire en entrant les commandes suivantes dans le terminal

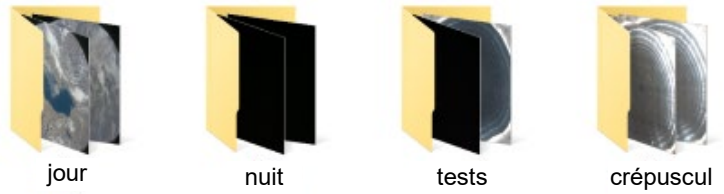
```
echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coraledgetpu.
list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-get update
sudo apt-get install libedgetpu1-std python3-pycoral -y
```

Une fois le logiciel installé, tu peux brancher ton périphérique Coral sur l'un des ports USB. Nous recommandons les ports USB 3 que tu peux identifier à leur couleur bleue.

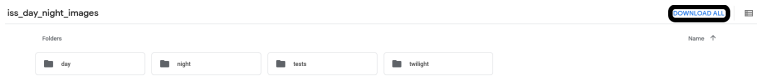


Tu peux choisir de réentraîner un modèle d'apprentissage automatique de deux manières différentes, soit sur ton ordinateur, à l'aide d'un script Python, soit en utilisant le site Teachable Machine de Google (https://teachablemachine.withgoogle.com/).

Dans les deux cas, tu auras besoin d'un ensemble d'images classées pour commencer. La manière la plus simple de classer les images est de les déplacer dans les répertoires désignés, où le nom du répertoire correspond au nom de la classe. Il est conseillé d'avoir quelques images dans un répertoire séparé pour les tests.



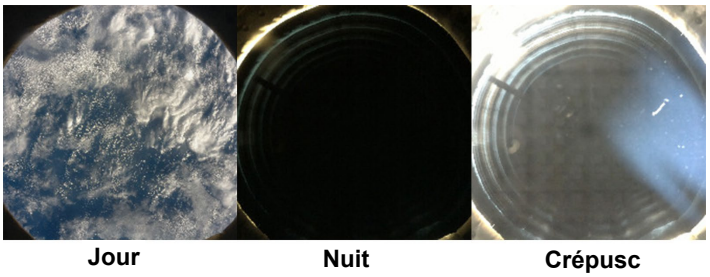
Tu peux créer des répertoires sur ton ordinateur pour toutes tes différentes classes d'images, et déplacer manuellement tes collections d'images dans les répertoires.
Tu peux également télécharger un ensemble d'images prises depuis l'ISS ici (https://drive.google.com/drive/folders/1owb4zoZSMld5qX0edCwZ1qZ6ypnJQ_5) qui ont déjà été classées.



Comment ces images ont-elles été prises ?

Les ensembles d'images que tu peux télécharger ont toutes été prises à l'aide d'un ordinateur Astro Pi sur l'ISS, que nous avons surnommé Izzy. Izzy dispose d'une caméra infrarouge proche, qui est dirigée vers la Terre à travers un hublot de l'ISS.

Les images prises ont été classées en 3 dossiers :**day (jour)**, **night (nuit)**, et **twilight (crépuscule)**. Certaines images ont également été conservées pour être utilisées dans les **tests**.



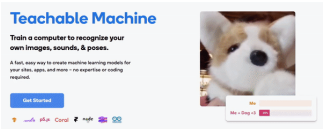
Il est parfois utile de savoir si l'ISS est actuellement en période de jour, de nuit ou de crépuscule, car cela peut avoir un effet sur d'autres mesures en cours. La classification de ces images pourrait donc être utile.

Si tu souhaites réentraîner un modèle à l'aide de l'outil Teachable Machine (https://teachablemachine.withgoogle.com/), tu peux passer dès maintenant à l'étape suivante.

Si tu souhaites réentraîner un modèle sur ton ordinateur, tu peux passer dès maintenant à l'étape Réentraîner un modèle sur ton Raspberry Pi (3).

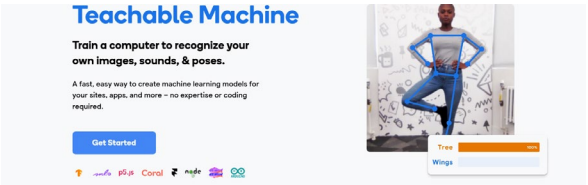
Étape 3 Réentraîner un modèle en ligne, avec Teachable Machine

Teachable Machine te permet de réentraîner facilement un modèle de reconnaissance d'images en téléchargeant des images qui ont été triées en classes, sur leur site et en utilisant leurs services Cloud pour créer un nouveau modèle.

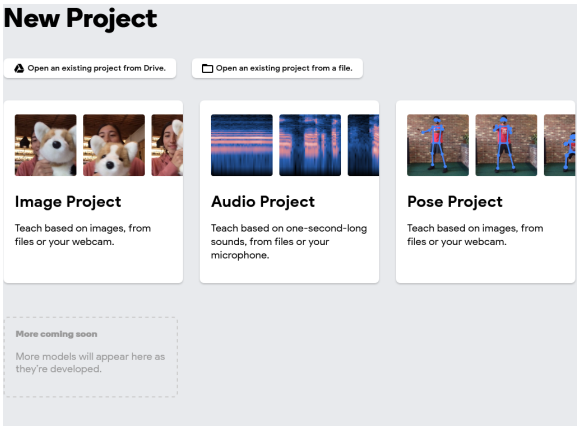


Dans une fenêtre de navigation, rends-toi sur le **site Web Teachable Machine** (https://teachablemachine.withgoogle.com/).{:target="_blank"}

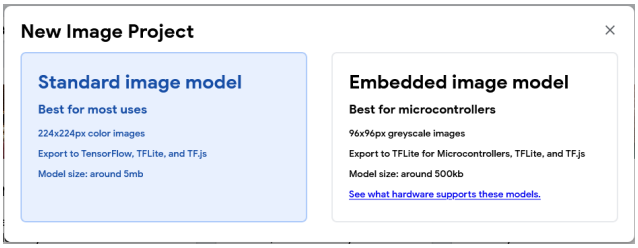
Clique sur le bouton **Commencer**.



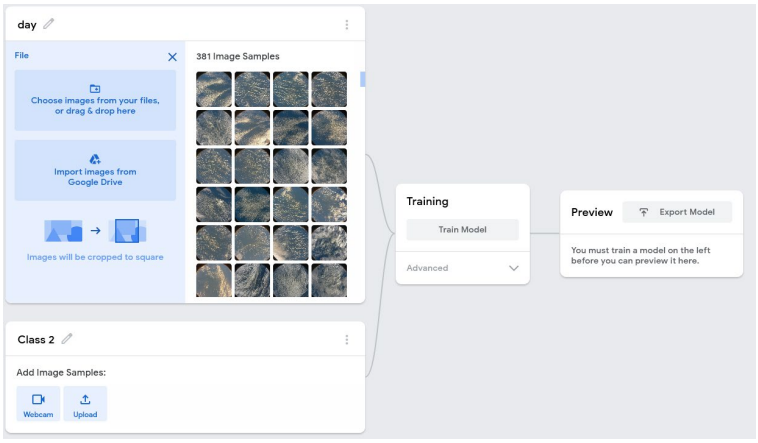
Choisis de créer un **Projet Images**.



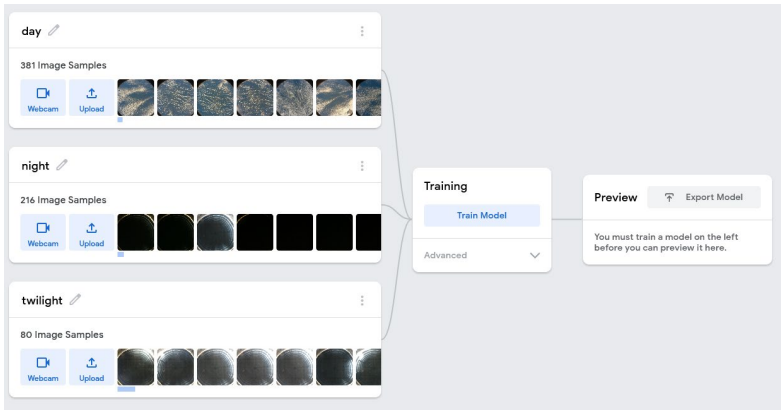
Sélectionne **Modèle d'image standard** lorsque l'on te le demande.



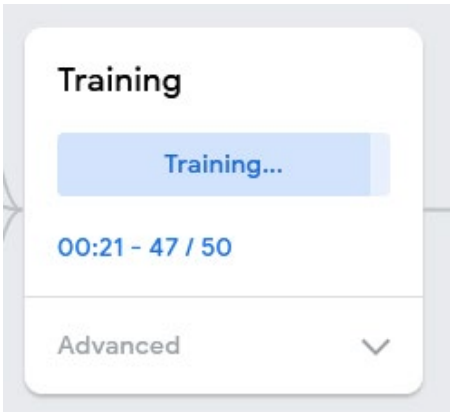
Renomme **Class 1** avec le nom de ton répertoire d'images, puis télécharge tes images dans ce répertoire.



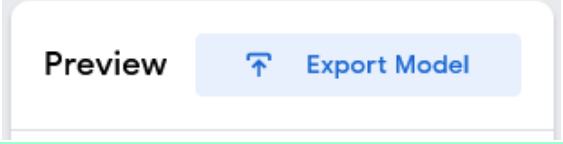
Répète cette étape pour tes autres répertoires d'images.



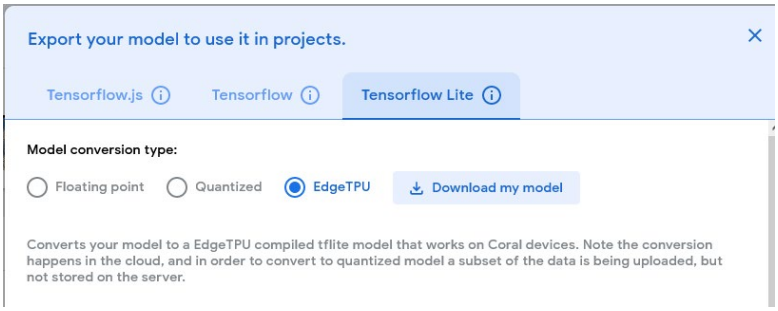
Clique sur le bouton **Entraîner le modèle**, pour réentraîner un modèle de reconnaissance d'images existant.



Clique sur le bouton **Exporter le modèle**.



Sélectionne les options **Tensorflow Lite** et **EdgeTPU**, puis clique sur **Télécharger mon modèle**.



Tu disposes maintenant d'un fichier que l'on appelle « modèle ». Il peut être utilisé par le TPU Coral pour classer de nouvelles images et t'indiquer si elles ont été prises de **jour (day)**, de **nuit (night)** ou au **crépuscule (twilight)**.

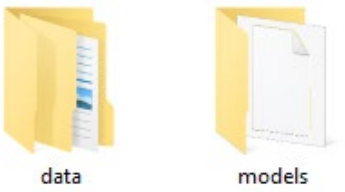
Tu peux maintenant passer à l'étape **Test de ton nouveau modèle** (5).

Étape 4 Réentraîner un modèle sur ton Raspberry Pi

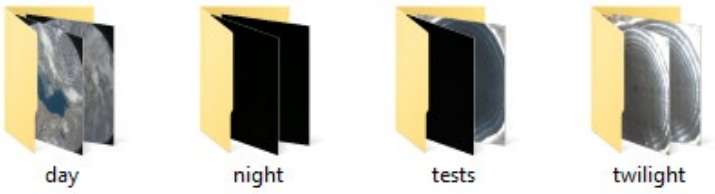
Tu peux également réentraîner un modèle existant à l'aide d'un script Python qui s'exécutera en utilisant tes répertoires d'images stockés localement.

Pour réentraîner un modèle sur ton ordinateur, tu devras connecter ton TPU Coral, organiser tes répertoires d'images et télécharger un modèle pré-entraîné.

Configure une structure de répertoire sur ton ordinateur afin d'avoir un répertoire pour les **données (data)** et un pour les **modèles (models)**.



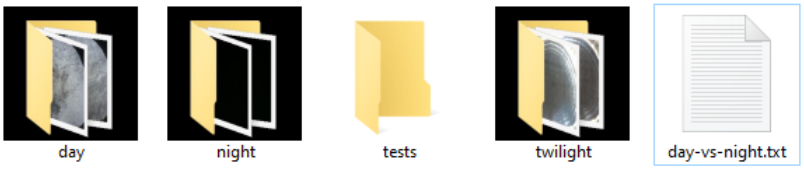
Dans le répertoire **data**, tu peux déplacer tes répertoires d'images classées et ton répertoire de tests.



Crée un fichier texte dans le répertoire **data** qui contient les étiquettes des répertoires d'images appropriés, ainsi qu'un ordre. Dans cet exemple, le fichier texte s'appelle **day-vs-night.txt**.

day-vs-night.txt

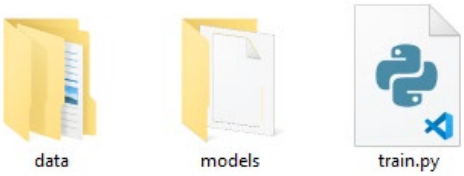
```
0 day
1 night
2 twilight
```



Télécharge un modèle de reconnaissance d'images **ici** (https://github.com/raspberrypilearning/image-id-coral/raw/draft/en/resources/mobilenet_v1_1.0_224_l2norm_quant_edgetpu.tflite).

Déplace le fichier de modèle dans ton répertoire **models**.

Télécharge le script d'entraînement **ici** (https://raw.githubusercontent.com/raspberrypilearning/imageid-coral/master/en/resources/train.py) et déplace-le dans le dossier parent.



Ouvre le script **train.py** à l'aide d'un éditeur de texte ou d'un IDE tel que **Thonny** (https://thonny.org/).

La ligne 30 définit le nom du modèle que tu vas générer à partir du réentraînement.
La ligne 33 demande le nom de ton fichier **label**.

Tu peux modifier ces lignes en fonction du nom de ton **modèle tflite** et de ton fichier **label**.

train.py

```
26 # the absolute path for the directory where this Python script is stored
27 script_dir = Path(__file__).parent.resolve()
28 # specify the input and output (retrained) model
29 model_path = script_dir/'models'/'mobilenet_v1_1.0_224_l2norm_quant_edgetpu.tflite'
30 out_model_path = script_dir/'models'/'astropi-day-vs-nite.tflite'
31 # specify where the labels and labelled training data are
32 # note: this is the simple version, could accept command-line arguments
33 data_dir = script_dir/'data'
34 labels_path = data_dir/'day-vs-night.txt'
```

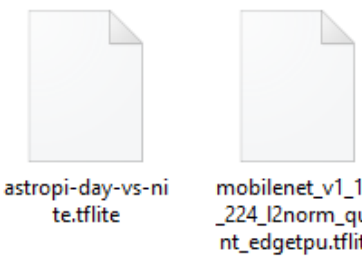
Lance le script à l'aide de ton IDE ou à partir de ton terminal.

```
python3 train.py
```

Tu devrais voir un résultat similaire à celui-ci :

```
26_St1_photo_193.jpg scores: [-0.0078125  0.        -0.0390625 ...  0.0390625  0.03125    0.0234375]
26_St1_photo_501.jpg scores: [-0.0390625 -0.0234375 -0.0390625 ...  0.046875   0.0390625 -0.0078125]
26_St1_photo_110.jpg scores: [ 0.015625  -0.046875  -0.0390625 ...  0.0234375  0.0234375  0.0625   ]
26_St1_photo_146.jpg scores: [ 0.        -0.03125   -0.046875  ...  0.0234375  0.046875   0.       ]
26_St1_photo_112.jpg scores: [ 0.015625  -0.03125   -0.046875  ...  0.0390625  0.0078125  0.0390625]
```

Ton nouveau modèle sera enregistré dans le répertoire **models**.



Quelle est la différence avec Teachable Machine ?

Lorsque tu réentraînes le modèle sur ton ordinateur, tu peux choisir le modèle que tu es en train de réentraîner. Avec Teachable Machine, tu utilises toujours un modèle qui a été choisi pour toi. En faisant l'entraînement sur ton ordinateur, tu as la possibilité de choisir parmi une grande variété de modèles différents, dont certains pourraient être mieux adaptés à la tâche que tu essayes d'accomplir.

Étape 5 Tester ton nouveau modèle

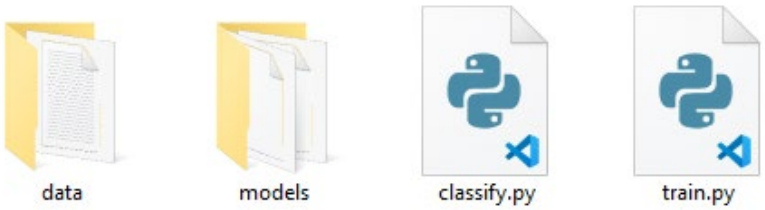
Maintenant que tu as un modèle entraîné, il est temps de l'utiliser pour classer les images.

```
pi@raspberrypi:~/pycoral $  cd /home/pi/pycoral ; /usr/bin/env /bin/python3
/home/pi/.vscode-server/extensions/ms-python.python-2021.8.1105858891/python
Files/lib/python/debugpy/launcher 40409 -- /home/pi/pycoral/classify.py
day: 0.99609
```

Le script que tu es sur le point de créer prendra une image de test, puis la fera passer par le modèle que tu as réentraîné, afin de tenter de classer l'image, et fournira un score indiquant le degré de confiance du modèle dans l'identification correcte de l'image.

Là encore, dans les exemples présentés, nous utilisons les images prises depuis l'ISS pour déterminer si elles ont été prises de jour, de nuit ou au crépuscule.

Ouvre ton IDE Python et crée un nouveau fichier appelé **classify.py**. Sauvegarde le fichier dans le même répertoire qui contient tes répertoires **data** et **models**.



Ajoute les éléments importés ci-dessous au début de ton fichier. Pathlib te permettra de naviguer dans la structure du répertoire, PIL est utilisé pour redimensionner les images, et pycoral est utilisé pour travailler avec le TPU Edge.

classify.py

```
1 from pathlib import Path
2 from PIL import Image
3 from pycoral.adapters import common
4 from pycoral.adapters import classify
5 from pycoral.utils.edgetpu import make_interpreter
6 from pycoral.utils.dataset import read_label_file
```

L'étape suivante consiste à indiquer à ton script où se trouvent tous les fichiers et répertoires. Pour cela, tu peux utiliser pathlib pour obtenir le nom du répertoire où le script a été enregistré.

Ajoute la ligne de code suivante, pour déterminer le répertoire où le script est enregistré.

classify.py

```
8 | script_dir = Path(__file__).parent.resolve()
```

Ensuite, les répertoires et les fichiers requis par le classificateur doivent être stockés sous forme de chemins, renvoyant à l'endroit où est enregistré ton script Python. Il s'agit du fichier model, du répertoire data, du fichier label et de l'image que tu utiliseras pour tester ton classificateur.

Ajoute ces quatre lignes et modifie les noms en fonction de la façon dont tu as nommé ton fichier model, ton fichier label et l'image de test que tu souhaites utiliser.

classify.py

```
10 model_file = script_dir/'models/astropi-day-vs-nite.tflite' # name of model
11 data_dir = script_dir/'data'
12 label_file = data_dir/'day-vs-night.txt' # Name of your label file
13 image_file = data_dir/'tests'/'day_3.jpg' # Name of image for classification
```

Il faut maintenant configurer l'interprète TensorFlow Lite pour utiliser le TPU Edge de Coral.

Ajoute les deux lignes suivantes pour initialiser l'interprète.

classify.py

```
15 interpreter = make_interpreter(f"{model_file}")
16 interpreter.allocate_tensors()
```

Les modèles d'apprentissage automatique préfèrent que toutes les images aient une taille définie, car cela rend la classification beaucoup plus simple. Si un modèle a été entraîné sur des images de 224 pixels par 224 pixels, il serait alors logique que toute image intégrée au modèle ait les mêmes dimensions.

Si tu as besoin que ton modèle détecte des détails plus fins dans une image, il faudra l'entraîner en utilisant des images avec une plus haute résolution. Cela peut être important si tu essaies de classer les images présentant des caractéristiques spécifiques, telles que des côtes dans une image Astro Pi, ou la couleur des yeux d'une personne dans un portrait.

Ajoute les lignes suivantes pour connaître les dimensions utilisées pour le modèle que tu as réentraîné, puis définis la taille de ton image de test aux mêmes dimensions, en utilisant PIL.

classify.py

```
18 size = common.input_size(interpreter)
19 image = Image.open(image_file).convert('RGB').resize(size, Image.ANTIALIAS)
```

Il faut maintenant soumettre l'image au modèle pour voir dans quelle mesure elle correspond à l'une des classes avec lesquelles le modèle a été entraîné.

Ajoute ces trois lignes pour classer ton image de test.

classify.py

```
21 common.set_input(interpreter, image)
22 interpreter.invoke()
23 classes = classify.get_classes(interpreter, top_k=1)
```

Enfin, le fichier label peut être utilisé pour fournir une classe lisible par les humains pour l'image. Un score est également fourni pour t'indiquer le degré de confiance du modèle dans son identification de l'image.

classify.py

```
labels = read_label_file(label_file)
for c in classes:
print(f'{labels.get(c.id, c.id)} {c.score:.5f}')
```

Exécute ton code et tu devrais voir un résultat t'indiquant la classe de ton image, ainsi qu'un pourcentage de confiance.

```
day: 0.99609
```

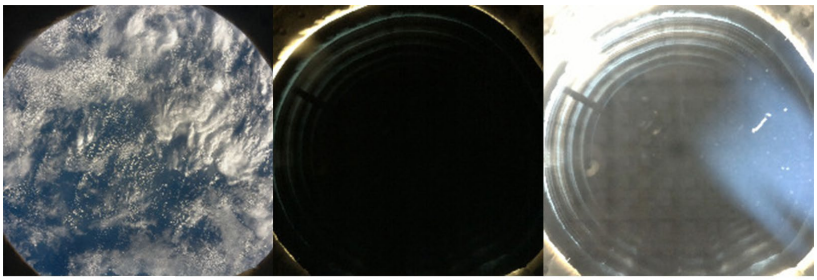
Enregistre ton projet

Étape 6 Mission Space Lab

Le **Mission Space Lab** (https://astro-pi.org/mission-spacelab/ ou https://esero.fr/projets/astro-pi/) fait partie du European Astro Pi Challenge, un projet éducatif de l'ESA mené en collaboration avec la Raspberry Pi Foundation, destiné aux jeunes de 19 ans et moins des **États membres de l'ESA** (https://astro-pi.org/mission-space-lab/eligibility), de Slovénie, de Lettonie, de Lituanie, du Canada et de Malte.

Le Mission Space Lab consiste à écrire un programme informatique qui sera exécuté sur l'une des deux unités Astro Pi de la Station spatiale internationale, afin de mener une expérience scientifique sur la vie sur Terre ou dans l'espace. Chaque unité Astro Pi sera équipée d'un accélérateur d'apprentissage automatique Coral.

Si tu souhaites intégrer le Mission Space Lab, tu peux expérimenter avec les images de jour, de nuit et de crépuscule (https://drive.google.com/drive/folders/1owb4zoZzSMld5qX0edCwZ1qZ6ypnJQ_5?usp=partage) fournies.



Jour Nuit Crépusc

Il existe cependant d'autres façons d'utiliser Coral et l'apprentissage automatique pour tes expériences de Mission Space Lab. Avec la reconnaissance d'images, tu pourrais essayer de détecter les océans, les terres ou les côtes. Tu pourrais chercher des chaînes de montagnes, des lacs ou des déserts.

Avec l'apprentissage automatique, tu peux également consulter d'autres sources de données. Tu peux analyser puis prédire les changements de température sur l'ISS, ou essayer de prédire des corrections orbitales.

Si tu souhaites en savoir plus sur l'utilisation de Coral, jette un œil aux exemples et à la documentation sur https://coral.ai (https://coral.ai).