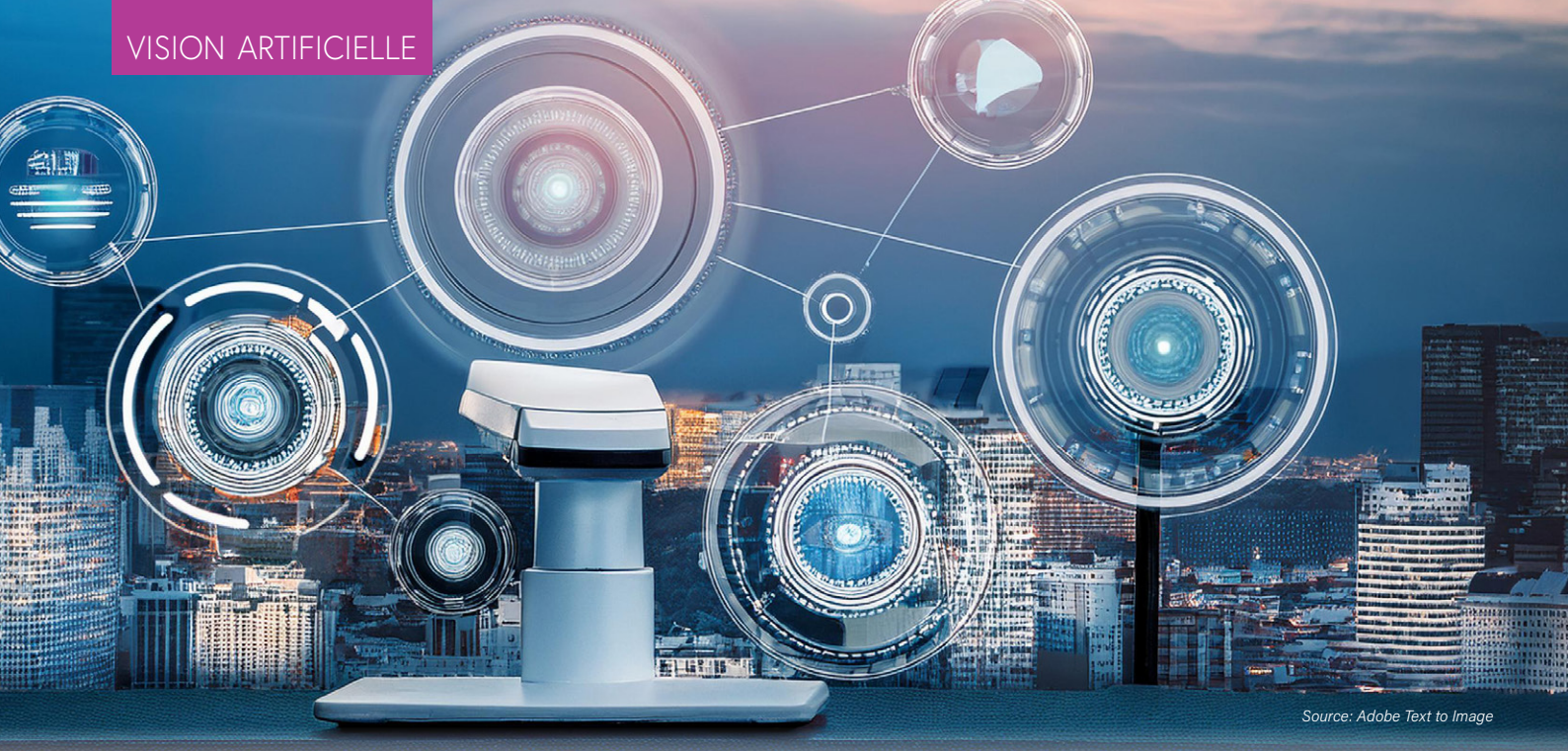


Source: Adobe Text to Image

vision artificielle avec OpenMV

Créer un détecteur de canettes de soda

Koen Vervloesem (Belgique)

Pour développer des applications de vision artificielle, OpenMV est une plateforme intéressante, aspirant à devenir « l'Arduino de la vision artificielle ». Elle combine un micrologiciel basé sur MicroPython avec des bibliothèques logicielles de vision artificielle simple à utiliser, un environnement de développement intégré (EDI) personnalisé et un matériel de caméra spécialisé. Dans ce tutoriel, nous utilisons OpenMV et une caméra Arduino Pro Nicla Vision pour détecter des objets tels que des canettes de soda.

Figure 1. Avec des dimensions de 22,86 mm × 22,86 mm × 7,26 mm, l'Arduino Nicla Vision est la plus petite caméra OpenMV disponible sur le marché.

Dans cet article, je démontrerai l'utilisation de la carte caméra Arduino Pro Nicla Vision [1] (**figure 1**) avec le micrologiciel OpenMV [2] pour la détection d'objets. Vous pouvez cependant exécuter ce projet sur n'importe quel matériel de caméra compatible avec OpenMV. La détection d'objets est un type particulier de classification d'images. Contrairement à un modèle de classification d'images classique qui se contente d'identifier la catégorie de l'objet présent dans l'image, un modèle de détection d'objets est également capable de localiser précisément l'objet détecté dans l'espace de l'image. Cela fonctionne même pour plusieurs objets dans une image. Ce processus est plus intensif en termes de calcul que la simple classification d'images, mais la carte Nicla Vision peut le gérer.

J'utiliserai l'approche FOMO (Faster Object, More Objects) pour exécuter un modèle de détection d'objets. FOMO est un algorithme performant qui permet la détection d'objets en temps réel avec des microcontrôleurs. Le projet commence par la constitution d'un ensemble d'images des objets que vous souhaitez détecter. Vous pouvez le faire en prenant des images avec la caméra de Nicla Vision. Une fois cet ensemble de données créé, vous pouvez entraîner le modèle FOMO avec la plateforme Edge Impulse [3]. Le modèle entraîné peut ensuite être flashé en tant que micrologiciel OpenMV personnalisé sur le Nicla Vision pour la détection d'objets. Comme OpenMV est basé sur le langage MicroPython [4], vous pouvez facilement personnaliser les actions de la carte lorsque des objets sont détectés.

Configuration de l'EDI OpenMV

Commencez par installer l'EDI OpenMV [5], qui fonctionne sous Windows, macOS et Linux (Ubuntu). Sur Ubuntu 23.04 et plus, vous devez modifier une ligne dans le script *setup.sh* de l'installateur *tar.gz* car les paquets installés en dehors d'un environnement virtuel Python ne sont plus supportés. Modifiez cette ligne :

```
sudo pip install pyusb
```

en

```
sudo apt install python3-usb
```

Exécutez ensuite le script *setup.sh* situé dans le répertoire décompressé. Après l'installation de l'EDI OpenMV, connectez votre carte Nicla Vision à votre ordinateur avec un câble micro-USB. Si la LED de la carte se met à clignoter en bleu, cela signifie que la carte exécute le script par défaut *main.py* d'OpenMV. Le stockage interne de la carte apparaît sur votre ordinateur comme un périphérique de stockage externe. Il contient un fichier *main.py* avec le code MicroPython du **listage 1**. Ce code fait clignoter la LED tant qu'il n'y a pas de connexion USB.

Listage 1. Clignotement de la LED intégrée pour vérifier si le matériel OpenMV fonctionne.

```
# main.py -- put your code here!
import pyb, time
led = pyb.LED(3)
usb = pyb.USB_VCP()
while (usb.isconnected()==False):
    led.on()
    time.sleep_ms(150)
    led.off()
    time.sleep_ms(100)
    led.on()
    time.sleep_ms(150)
    led.off()
    time.sleep_ms(600)
```

Connexion à votre carte OpenMV

Lancez l'EDI OpenMV et cliquez sur l'icône *Connect* en bas à gauche. La LED devrait s'arrêter de clignoter, et vous pourriez recevoir un message indiquant que le micrologiciel de la carte n'est pas à jour. Confirmez que vous souhaitez mettre à jour le micrologiciel et effacer le système de fichiers interne. Ce processus peut prendre du temps et le voyant de la carte clignotera en vert pendant un certain temps. Une fois que le firmware a été mis à jour et que l'EDI OpenMV a établi une connexion avec votre carte, la LED cesse de clignoter. La barre d'état en bas devrait afficher le nom de votre carte, le capteur, la version du micrologiciel, le port série et le lecteur sur lequel le stockage interne est connecté (**figure 2**).

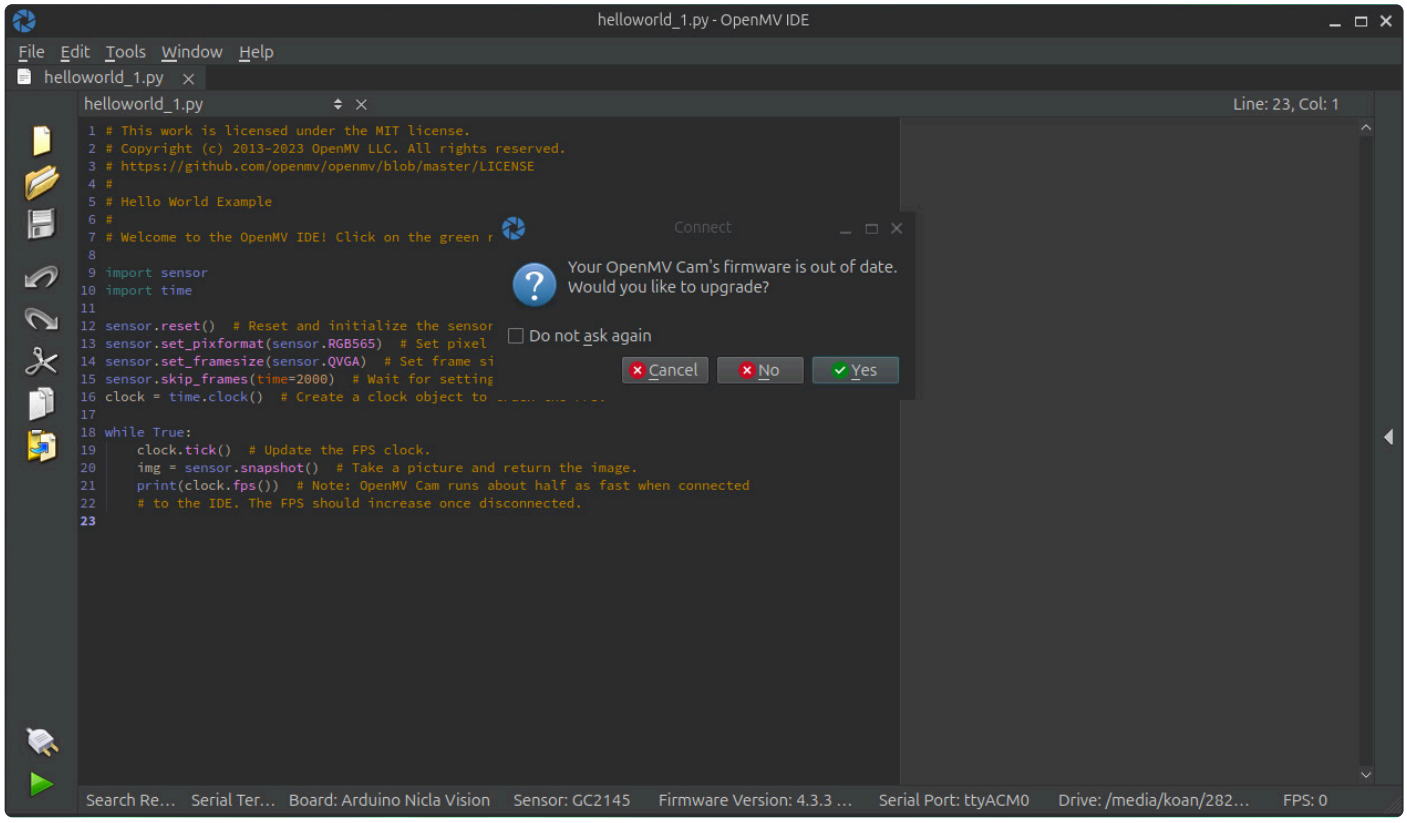


Figure 2. L'EDI OpenMV requiert la mise à jour du micrologiciel de la carte de la caméra.

Listage 2. Obtenir une vue en direct de la caméra de la carte OpenMV.

```
# This work is licensed under the MIT license.
# Copyright (c) 2013-2023 OpenMV LLC. All rights reserved.
# https://github.com/openmv/openmv/blob/master/LICENSE
#
# Hello World Example
#
# Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!

import sensor
import time

sensor.reset()  # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)  # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)  # Set frame size to QVGA (320x240)
sensor.skip_frames(time=2000)  # Wait for settings take effect.
clock = time.clock()  # Create a clock object to track the FPS.

while True:
   clock.tick()  # Update the FPS clock.
   img = sensor.snapshot()  # Take a picture and return the image.
   print(clock.fps())  # Note: OpenMV Cam runs about half as fast when connected
   # to the IDE. The FPS should increase once disconnected.
```

Listage 3. Le script de capture de données enregistre des images RGB de 240 × 240 pixels.

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((240, 240))
sensor.skip_frames(time = 2000)

clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    print(clock.fps())
```

Pour tester le fonctionnement de la caméra, exécutez le script par défaut *helloworld_1.py*, en cliquant sur l'icône verte *run* en bas à gauche. Cela transformera la caméra en webcam, et affichera son flux dans l'EDI OpenMV. La partie *Frame Buffer* de la fenêtre montre l'image de la caméra en direct, tandis que la partie *Histogram* montre les composantes rouge, verte et bleue de l'image. Vous pouvez modifier cet espace colorimétrique en niveaux de gris, LAB ou YUV. En bas à droite, vous pouvez également voir le nombre d'images par seconde que la caméra diffuse en continu.

Examinons ce script *helloworld_1.py* par défaut (**listage 2**). Après avoir importé les modules Python nécessaires, le script réinitialise la caméra, puis définit le format des pixels et la taille des images. Après un délai de deux secondes, il entre dans une boucle pour capturer continuellement des images et obtenir le nombre actuel d'images par seconde.

Création d'un ensemble de données

Une fois que votre caméra est configurée, il est temps de construire un ensemble de données en capturant des images de divers objets. L'EDI OpenMV offre un support intégré pour cette tâche. Ouvrez le menu *Tools* / *Dataset Editor* / *New Dataset* et choisissez un répertoire pour enregistrer votre ensemble de données.

L'EDI affiche alors un éditeur de jeu de données sur la gauche, le script de capture de jeu de données (qui est identique au script *hello world* précédent) au milieu, et le *frame buffe*r et l'histogramme sur la droite. Cliquez sur l'icône *New Class Folder* dans la barre latérale gauche, ou appuyez sur *Ctrl+Shift+N* pour créer une nouvelle classe pour tout objet que vous souhaitez détecter.

Pour cet exemple, j'entraînerai le modèle à détecter les canettes de boissons gazeuses. Je vais donc créer une classe nommée *can*. Comme le modèle FOMO nécessite des images carrées et que la caméra offre une résolution de 320×240, il est indispensable d'ajuster le script de capture de données en conséquence (**listage 3**).

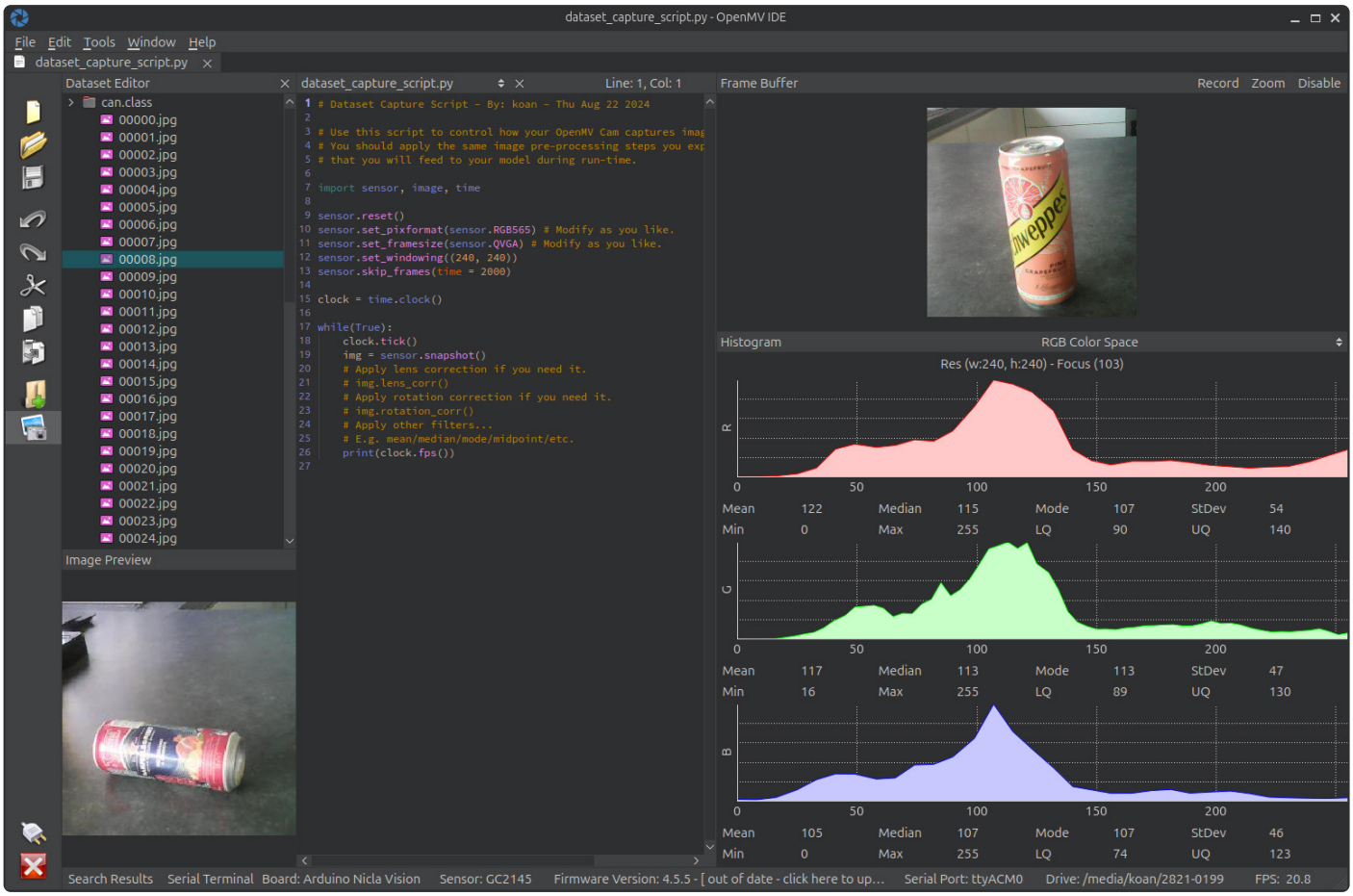


Figure 3. Créez votre nouvel ensemble de données composé d'images de différentes canettes de soda.

Exécutez ce script en appuyant sur l'icône de lecture, sélectionnez l'une des classes en cliquant sur son nom de dossier dans l'éditeur de jeu de données, et pointez la caméra sur un objet de cette classe. Capturez une image en cliquant sur l'icône de l'appareil photo (*Capture Data*) à gauche. Prenez 30 à 50 images de diverses canettes de soda sous différents angles, distances, conditions d'éclairage et arrière-plans (**figure 3**).

Étiquetez votre jeu de données dans Edge Impulse

L'EDI OpenMV est directement intégré à Edge Impulse, facilitant ainsi le chargement de votre jeu de données vers ce service cloud pour l'entraînement d'un modèle. Pour débuter, inscrivez-vous sur Edge Impulse ; l'inscription est gratuite pour les étudiants, les universités et les développeurs individuels dans le cadre du plan communautaire [6]. Dans votre tableau de bord Edge Impulse, cliquez sur *Create new project*, donnez un nom au projet, choisissez de le rendre public ou privé (**figure 4**).

Dans l'EDI OpenMV, naviguez vers *Tools* / *Dataset Editor* / *Export* / *Login to Edge Impulse Account* et *Upload to Project*. Connectez-vous avec vos identifiants Edge Impulse et choisissez le projet. Dans l'étape suivante, vous devez choisir la répartition entre les données d'entraînement et de test. Vous pouvez laisser la répartition par défaut 80%/20%.

Une fois les images téléchargées, l'ensemble de données apparaît dans

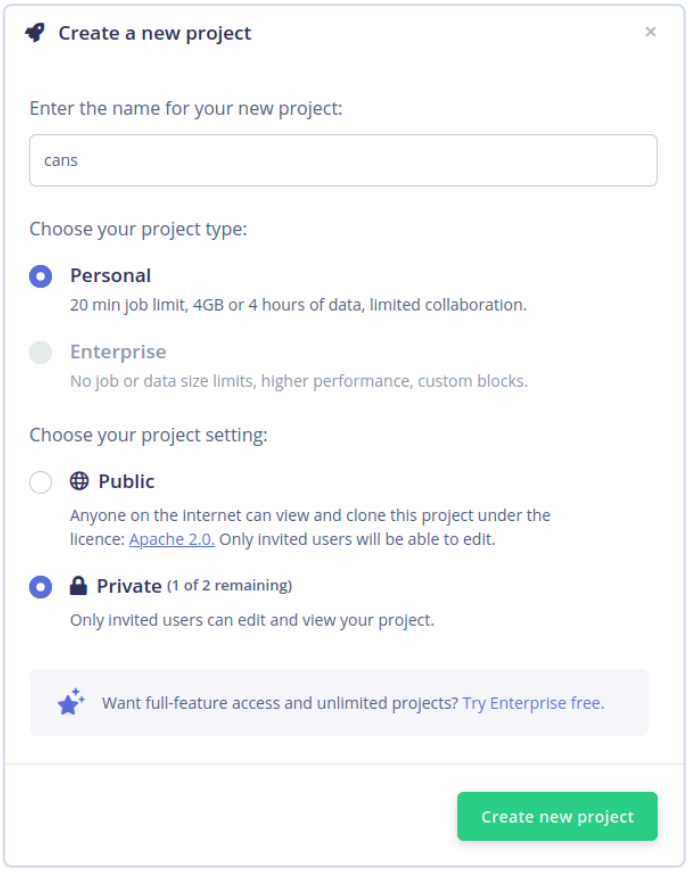


Figure 4. Créer un nouveau projet dans Edge Impulse.

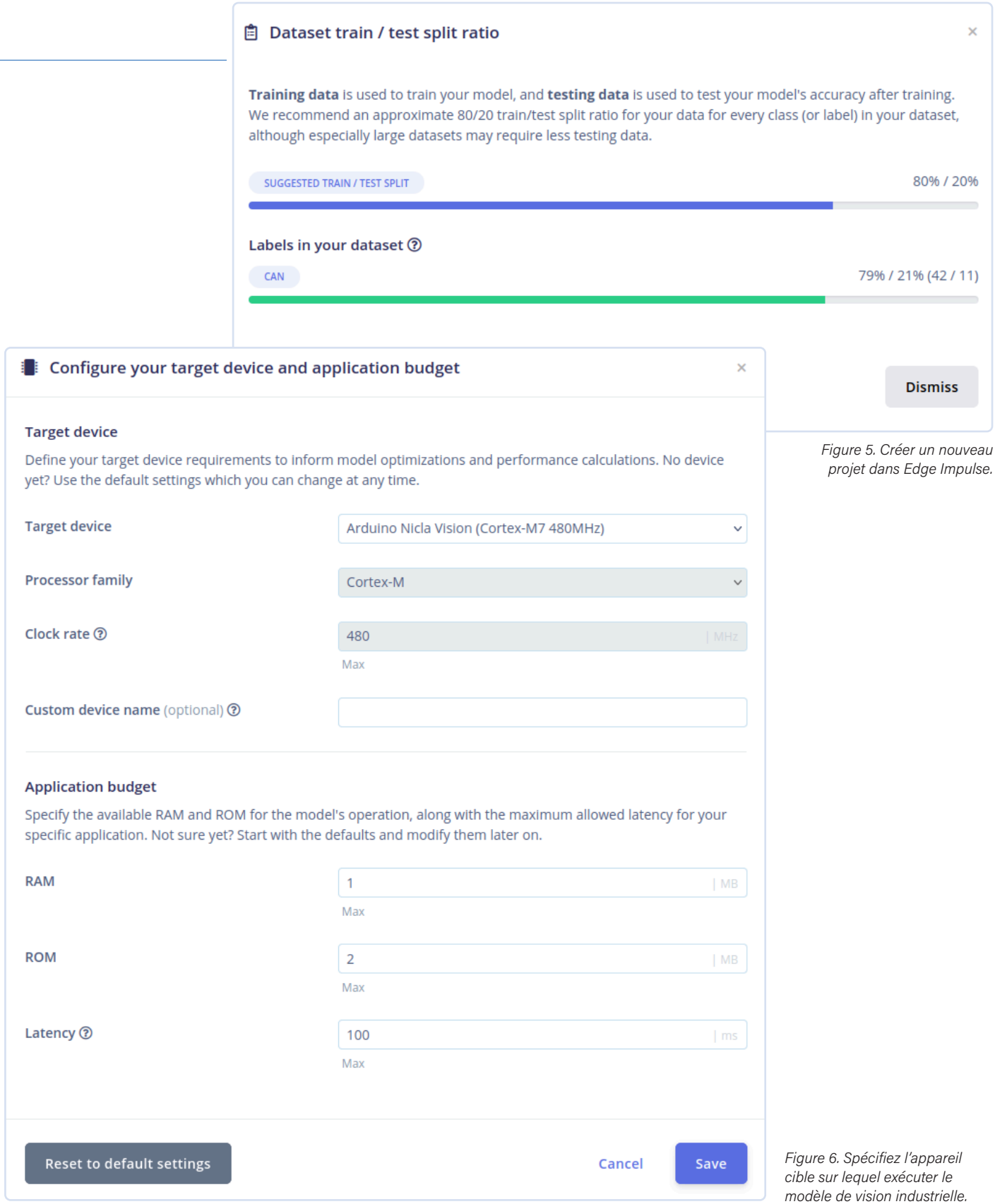


Figure 5. Créer un nouveau projet dans Edge Impulse.

Figure 6. Spécifiez l'appareil cible sur lequel exécuter le modèle de vision industrielle.

l'onglet *Data acquisition* de votre projet Edge Impulse. Le système détecte l'ajout de nouvelles données d'image et vous propose de créer un projet de détection d'objet. Confirmez pour poursuivre.
Cliquez ensuite sur *Labeling queue*. Pour chaque image, tracez un cadre autour de l'objet à identifier, ajoutez une étiquette et cliquez sur *Save labels* pour sauvegarder avant de passer à la suivante. Si une image contient plusieurs objets, dessinez une boîte autour de chacun d'entre eux et attribuez-lui une étiquette. Une fois que vous avez étiqueté tous les échantillons, revérifiez votre travail pour vous assurer de l'exactitude de l'étiquetage. L'étiquetage d'un grand nombre d'images est une tâche ennuyeuse et sujette à erreur, ce qui peut influer significativement l'apprentissage du modèle.

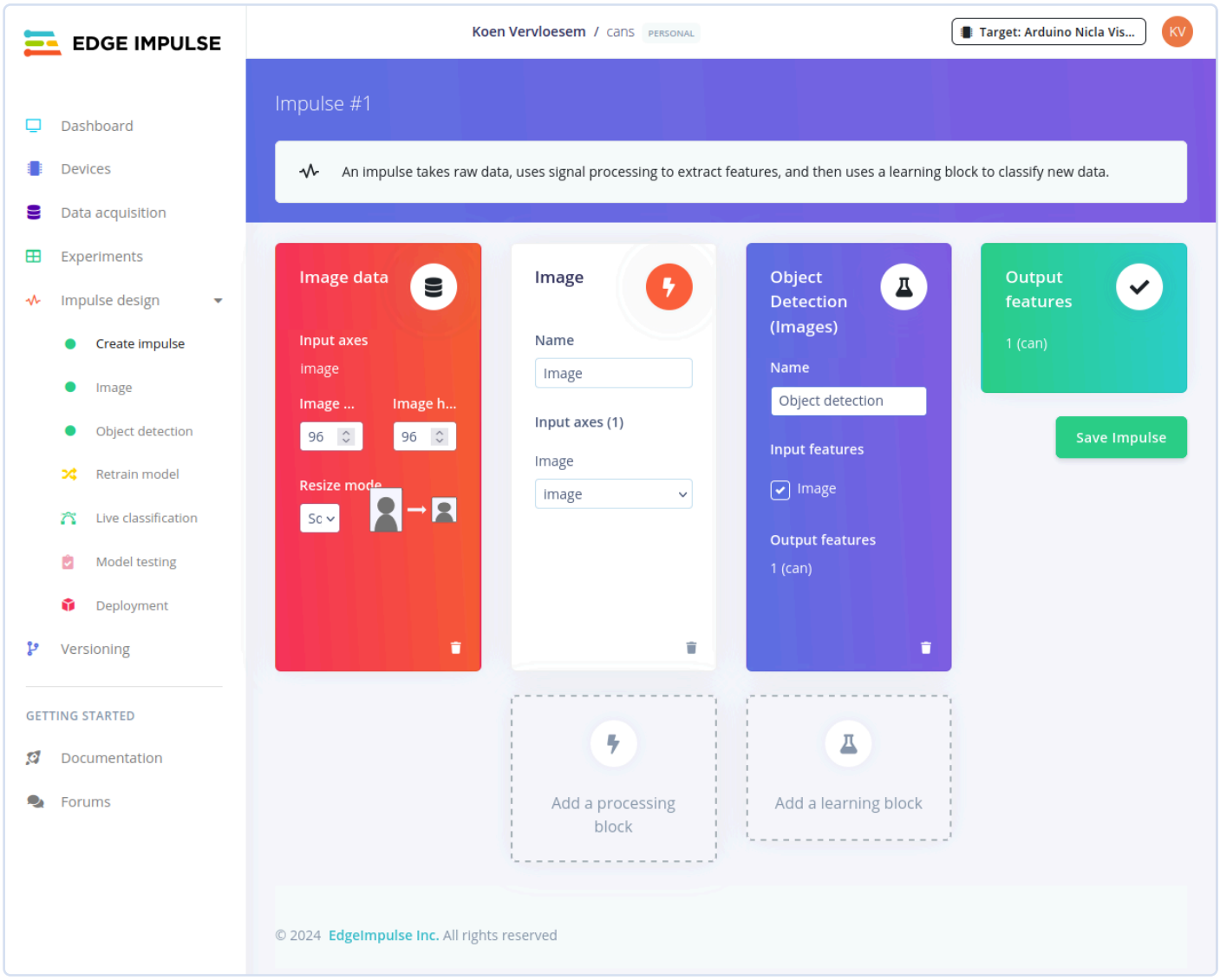


Figure 7. Créez une impulsion pour classer vos données.

S'il y a un problème avec le taux entraînement/test, un signe d'avertissement apparaîtra. Cliquez dessus pour obtenir plus de détails (**figure 5**). Par exemple, l'une des classes pourrait avoir un nombre insuffisant d'échantillons. Pour résoudre ce problème, ajoutez plus d'images dans l'EDI OpenMV et chargez à nouveau les données. La procédure de chargement évite les doublons et n'ajoute pas une même image deux fois dans votre projet de test. Si la répartition entre les données d'apprentissage et de test n'est pas équilibrée, cliquez sur les trois points à droite d'un échantillon dans les données d'apprentissage, puis sur *Move to test* jusqu'à ce que chaque étiquette de votre ensemble de données présente un rapport entraînement/test proche de 80 %/20 %.

Création du modèle

Avant de créer votre modèle, configurez l'appareil cible, qui sera utilisé pour optimiser le modèle et évaluer ses performances. Cliquez sur l'icône cible en haut à droite à côté des initiales de votre profil. Choisissez ensuite votre appareil cible (**figure 6**). L'Arduino Nicla Vision possède deux cœurs : un Cortex-M7 480 MHz et un Cortex-M4 240 MHz, sélectionnez l'un des deux. Sous *Application budget*, définissez la RAM et la ROM disponibles et la latence maximale autorisée. Vous pouvez commencer par les valeurs par défaut.

Ensuite, allez dans *Impulse design* et cliquez sur *Create impulse*. Dans le premier bloc, *Image data*, définissez la largeur et la hauteur de l'image à 96 pixels. Comme les images originales sont déjà carrées, le mode de redimensionnement n'est pas crucial ici. Cliquez ensuite sur *Add a processing block* et cliquez sur *Add* à côté du bloc *Image*. Ensuite, cliquez sur *Add a learning block* et cliquez sur *Add* à côté de *Object Detection (Images)*. Vous verrez que le bloc affiche automatiquement la classe *can* dans votre ensemble de données comme *output features*. Il est maintenant également affiché dans le bloc *Output features* à la fin. Cliquez sur *Save Impulse* (**figure 7**).

Une nouvelle partie de *Impulse design* apparaît maintenant : *Image*. Cliquez dessus pour afficher les images brutes (**figure 8**, vous pouvez choisir des images spécifiques dans le menu déroulant en haut à droite). Sous *Parameters*, réglez la profondeur de couleur sur *RGB* et cliquez sur *Save parameters*. Vous êtes maintenant dans l'onglet *Generate features*.

Cliquez sur *Generate features* pour appliquer le bloc de traitement à toutes les images de l'ensemble de données. Au final, cela crée une visualisation 3D de toutes les données d'apprentissage, regroupées en fonction de leur similarité. Cette opération peut prendre un certain temps, en fonction de la taille de vos données d'apprentissage. Si vous voyez tous les points de données de la même classe clairement regroupés, cela signifie que le modèle a un moyen facile d'apprendre à distinguer les classes. Si ce n'est pas le cas, vous devriez essayer de créer plus d'images, ou des images plus claires.

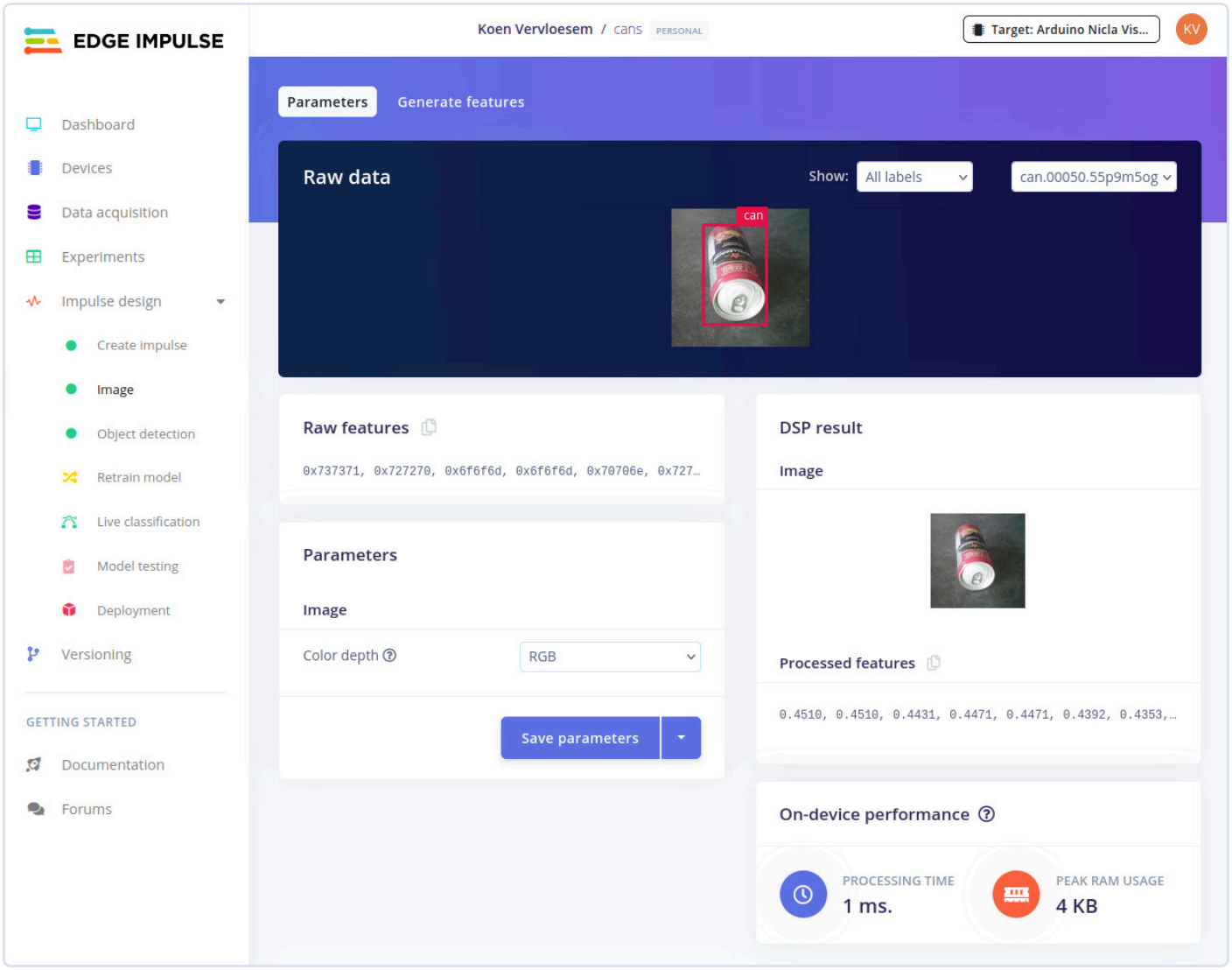


Figure 8. Edge Impulse vous montre les données brutes dont il dispose pour chaque image.

L'étape suivante consiste à configurer le modèle de détection d'objets. Cliquez sur *Object detection*. Le modèle affiché est *FOMO (Faster Objects, More Objects) MobileNetV2 0.35.* Vous pouvez toujours choisir un autre modèle, mais le modèle et les paramètres par défaut sont parfaits. Cliquez sur *Save & train*. Cela divisera les données d'entraînement en un ensemble d'entraînement et de validation, et l'ensemble du processus prendra un certain temps.

Une fois le modèle entrainé, des statistiques sur la précision apparaîtront sous la sortie de formation. Ces chiffres indiquent la qualité des performances du modèle. S'ils ne sont pas satisfaisants, mais montrent une amélioration au cours de l'entraînement, envisagez de réentrainer le modèle avec plus de cycles.

Les chiffres figurant sous *On-device performance* sont également intéressants : ils indiquent le temps d'inférence du modèle, le pic d'utilisation de la RAM et l'utilisation de la mémoire flash. Si le temps d'inférence semble trop élevé, vous devez réduire la résolution de l'image et recréer le modèle. L'utilisation maximale de la mémoire vive et l'utilisation de la mémoire flash sont également des facteurs importants. Vous devrez les comparer aux spécifications de votre carte. Par exemple, la Nicla Vision a 1 MB RAM. Ma première tentative de création d'un modèle avec 240 × 240 images en niveaux de gris avait 1,3 MB de RAM en pic d'utilisation, ce qui aurait été impossible à exécuter sur la carte, ayant 1 MB de RAM.

Dans mon cas, j'ai atteint une performance d'entraînement où le modèle a été capable de détecter 75 % des canettes correctement, avec seulement 50 images, ce qui est un bon début. Le temps d'inférence est de 52 ms et le pic d'utilisation de la RAM de 239,5 K, ce qui rend le modèle adapté pour fonctionner sur le Nicla Vision avec une détection presque immédiate.

Cependant, Edge Impulse optimise par défaut le modèle avec une quantification int8, qui est plus rapide et utilise moins de RAM. Si vous sélectionnez la version *Unoptimized (float32)* du modèle, les performances de l'entraînement peuvent être bien meilleures. Dans mon cas, les performances de détection ont atteint 90.9 %. Cependant, le temps d'inférence a augmenté à 125 ms, et le pic d'utilisation de la RAM était de 887.1 K (**figure 9**).

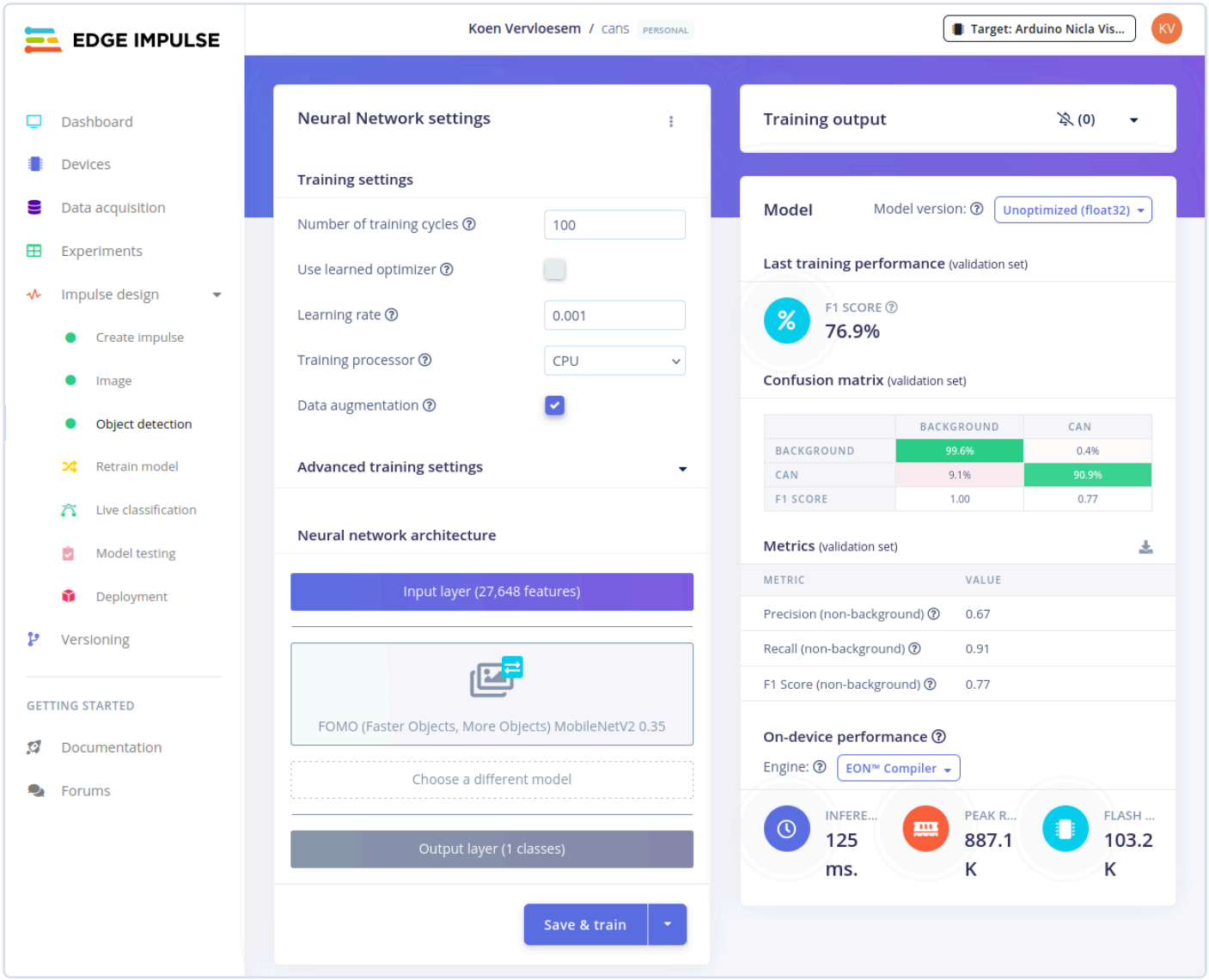


Figure 9. Le réseau neuronal permettant de détecter les canettes de soda a été entraîné.

Test du modèle

Maintenant que vous avez entraîné le modèle, il est temps de le tester. Edge Impulse a entraîné le modèle uniquement sur les données d'apprentissage, vous pouvez donc utiliser les images de l'ensemble de données de test pour tester le modèle. Rendez-vous dans la section *Model testing* et cliquez sur *Classify all*. Cela exécutera le modèle sur toutes les images des données de test et comparera la sortie du modèle à l'étiquette que vous avez attribuée à l'image.

Si la précision de cette validation est faible, cela signifie que votre modèle est trop adapté aux données de l'ensemble de données d'apprentissage. Une solution consiste à réentraîner le modèle avec un taux d'apprentissage plus faible. Vous pouvez également élargir votre ensemble de données avec des exemples plus variés de l'objet que vous souhaitez détecter, les étiqueter et recréer le modèle. Il est nécessaire de procéder à quelques tests pour affiner votre modèle.

Déploiement sur votre appareil

Il est maintenant temps de déployer le modèle sur votre appareil OpenMV. Edge Impulse dispose d'un support direct pour créer un micrologiciel OpenMV que vous pouvez flasher sur votre appareil. Rendez-vous sur *Deployment*, sélectionnez *OpenMV firmware* dans le champ de recherche, choisissez la version du modèle, puis cliquez sur *Build*. Une fois la compilation terminée, votre navigateur télécharge le fichier zip. Si le téléchargement ne se lance pas, cliquez sur *Latest build* en haut à droite pour télécharger le fichier.

Le fichier zip contient des fichiers *.bin* adaptés à tous les appareils OpenMV pris en charge. Pour le Nicla Vision, vous avez besoin du fichier *edge_impulse_firmware_arduino_nicla_vision.bin*. Extrayez ce fichier. Ensuite, dans l'EDI OpenMV, naviguez vers *Tools* / *Run Bootloader (Load Firmware)* et sélectionnez le fichier bin que vous avez extrait. Sélectionnez *Erase internal file system* et cliquez sur *Run*. Le micrologiciel sera alors flashé sur votre appareil.

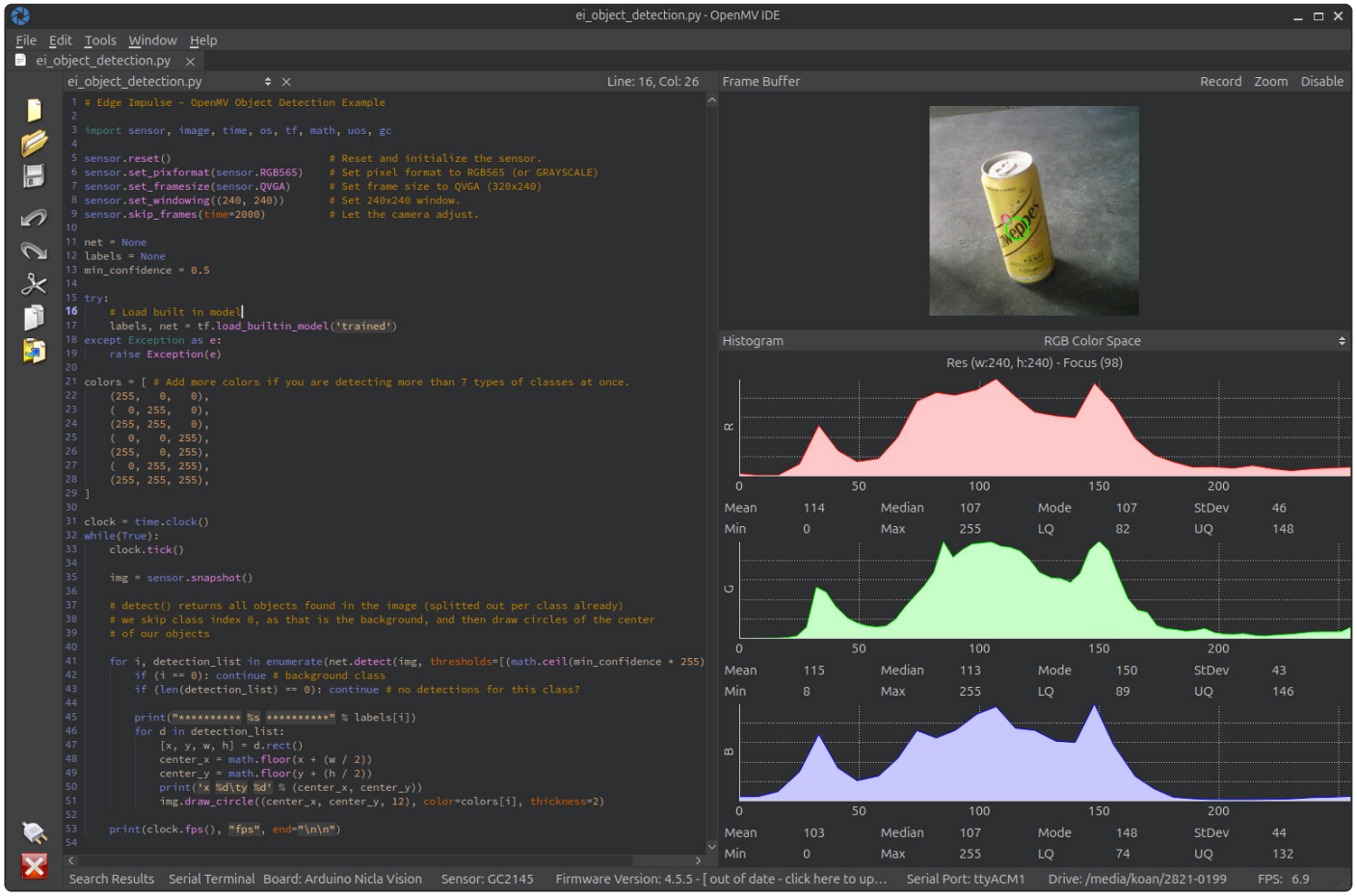


Figure 10. Le script OpenMV peut maintenant détecter les canettes de soda dans les images de la caméra.

Extrayez ensuite le fichier *ei_object_detection.py* du fichier zip et ouvrez-le dans l'EDI OpenMV. Cliquez sur l'icône de lecture pour l'exécuter sur votre appareil OpenMV. Ce projet exécute le modèle TensorFlow Lite sur l'image de la caméra en direct et dessine un cercle au centre des objets détectés (**figure 10**).

Facilement personnalisable

L'entraînement d'un modèle de détection d'objets prend du temps, et la qualité des données est un facteur important à prendre en compte. Dans cet article, je vous ai présenté un exemple simple où le modèle est configuré pour reconnaître une seule classe d'objets. Vous pouvez ajouter d'autres classes, par exemple des boîtes de conserve, des bouteilles, etc. L'EDI OpenMV s'intègre parfaitement au service en ligne Edge Impulse, ce qui simplifie du processus. De plus, le micrologiciel généré exécute du code MicroPython, ce qui vous permet de personnaliser facilement le comportement de l'appareil lorsqu'il détecte un objet. Par exemple, ajoutez du code MicroPython pour changer la couleur de la LED en fonction de la classe de l'objet détecté.

240450-04

LIENS

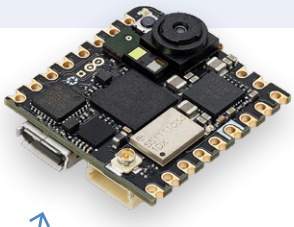
[1] Arduino Pro Nicla Vision : https://docs.arduino.cc/hardware/nicla-vision/
[2] OpenMV : https://openmv.io
[3] Edge Impulse : https://edgeimpulse.com
[4] MicroPython: https://micropython.org
[5] EDI OpenMV : https://openmv.io/pages/download
[6] Edge Impulse plans : https://edgeimpulse.com/pricing

Questions ou commentaires ?

Envoyez un courriel à l'auteur (koen@vervloesem.eu), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

KoKoen Vervloesem est auteur d'articles sur Linux et l'open source, la sécurité informatique, la vie privée, la programmation, l'IA et l'Internet des objets depuis plus de 20 ans. Il est titulaire d'un master en ingénierie informatique et en philosophie et donne des cours sur Linux, Python et l'IdO.



Product

Arduino Pro Nicla Vision
www.elektor.fr/20152