TRAITEMENT D'IMAGES avec le kit Jetson Nano de Nvidia

Figure 1. Kit de développement Jetson Nano. (Source : developer.nvidia.com)

2e partie : reconnaissance d'images

Mathias Claußen (Elektor)

Si vous recherchez un moyen simple pour vous lancer dans le monde des réseaux neuronaux, vous devriez jeter un coup d'œil à Edge Impulse. Vous aurez ainsi accès à un large éventail de cartes et dispositifs de développement, par ex. pour capturer et construire des données d'images utilisables, ce qui vous permettra de créer des réseaux neuronaux dans le nuage. Vous pourrez ensuite les utiliser dans vos propres applications. Nous avons ainsi testé Edge Impulse, une carte de développement Jetson Nano et un certain nombre de cartes à jouer.

Après la présentation de Jetson Nano (**fig. 1**) et du kit JetBot AI de SparkFun (**fig. 2**) dans la première partie de cette série [1], il est maintenant temps de faire nos premiers pas dans la reconnaissance d'images. Pour reconnaître une image à l'aide du Jetson Nano, il faut entraîner un logiciel et un réseau neuronal. Comme nous l'avons déjà décrit dans l'article en ligne « Faire du café grâce au MAX78000 et à une pincée d'IA » [2], l'entraînement du réseau n'est pas une tâche facile. D'abord vous devrez vous familiariser avec l'application de type console qui s'appuie sur le *framework* PyTorch ou la bibliothèque TensorFlow. La création d'un réseau neuronal sur votre propre ordinateur prendra également un certain temps, en fonction de la carte graphique et du processeur de la machine.

Les défis ne s'arrêtent pas là : il faut avant tout entraîner le réseau neuronal à l'aide d'un ensemble de données. Des applications supplémentaires sont également nécessaires pour construire l'ensemble de données et le préparer pour l'entraînement. Un certain temps est donc nécessaire pour pouvoir entraîner notre premier réseau neuronal.

Edge Impulse

Si vous cherchez une introduction relativement simple au monde des réseaux neuronaux, n'hésitez pas à découvrir les capacités d'Edge Impulse [3]. Il s'agit d'une plateforme web destinée à traiter vos ensembles de données pour construire des réseaux neuronaux et les porter sur une grande variété de dispositifs. Ce portage peut aussi bien concerner un Arduino qu'un ordinateur portable doté d'un processeur x86/AMD64. Ici, Edge Impulse sert à créer un réseau neuronal, ensuite utilisé avec le kit JetBot pour reconnaître quelques objets. Les données peuvent être capturées à l'aide de différents appareils, par ex. un PC avec un navigateur web ou un smartphone. L'ensemble de données est envoyé directement aux serveurs Edge Impulse et y est stocké. Si vous disposez déjà d'un tel ensemble de données, vous pouvez simplement le télécharger via l'interface web.
Le tri et l'étiquetage du jeu de données s'effectuent ensuite via l'interface web. Les paramètres de création du réseau neuronal y sont également définis et configurés.
La création d'un réseau neuronal est très exigeante en puissance de calcul. Les données sont donc téléchargées sur le serveur Edge Impulse et y sont converties en réseau neuronal. L'entraînement a donc lieu dans le nuage et n'impose pas de charge à votre propre ordinateur. Le réseau neuronal créé peut ensuite être converti en bibliothèque de programmes, téléchargé puis utilisé dans d'autres applications.

Nous détaillons toutes ces étapes en construisant la modeste application de reconnaissance d'images proposée ci-dessous. Nous avons publié un article sur l'utilisation d'Edge Impulse pour la détection du bruit [4] dans le magazine Elektor Industry, autre publication d'Elektor.

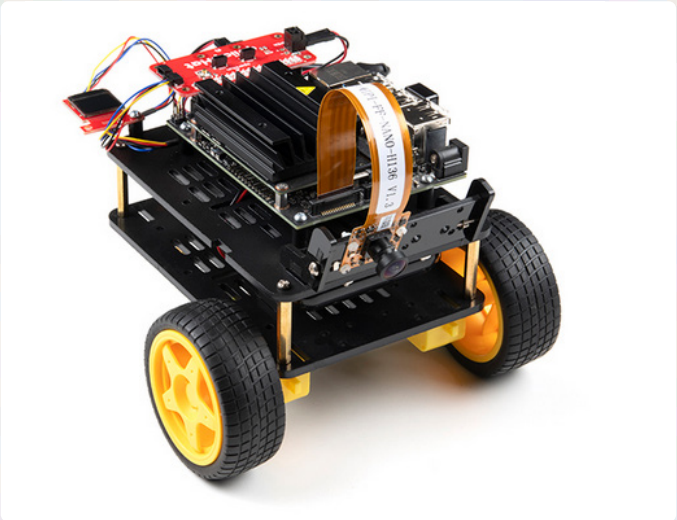

Figure 2. Kit JetBot AI de SparkFun. (Source : SparkFun)

Figure 3. Cartes UNO.

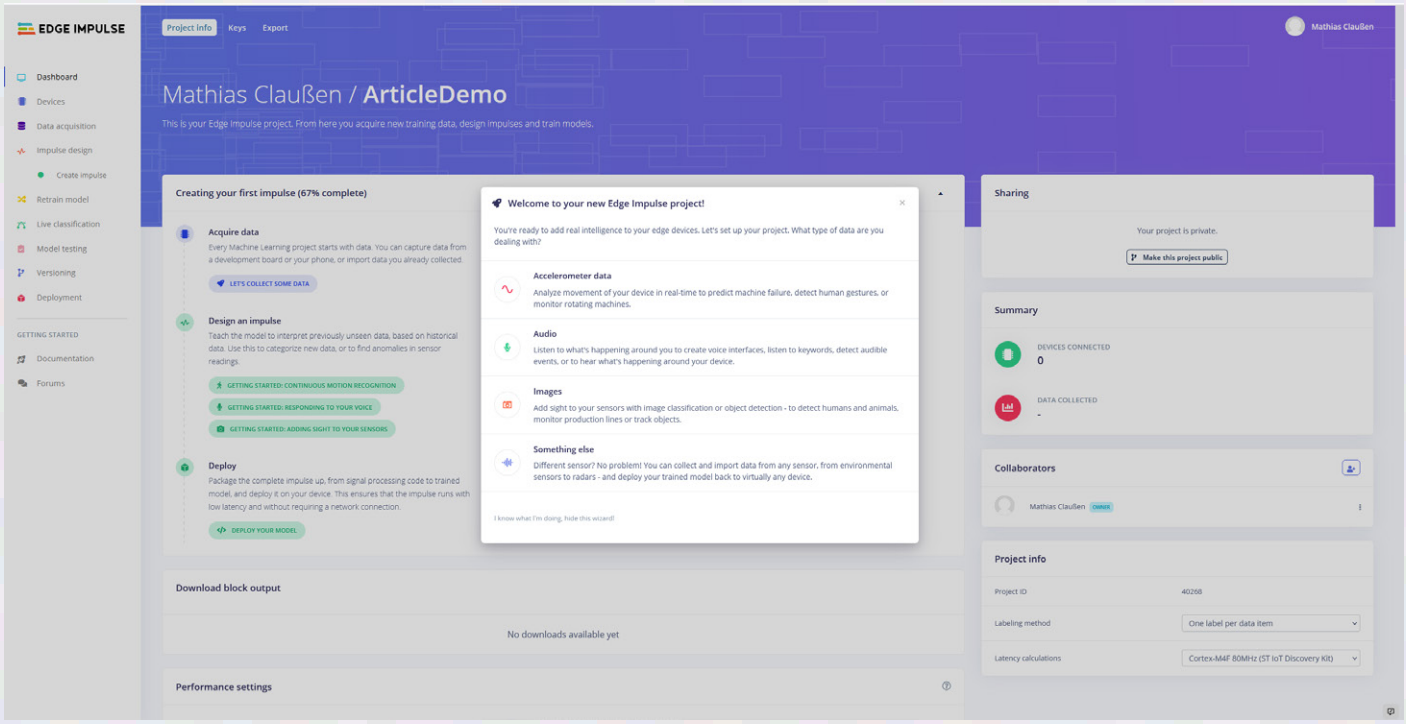

Figure 4. Assistant Edge Impulse.

Distribuer les cartes

Pour commencer, il est bon d'utiliser des objets visuellement très distincts comme éléments de test. Nous avons choisi différentes cartes à jouer du jeu UNO de Mattel (**fig. 3**), que nous avions à portée de main, mais vous pourriez utiliser d'autres objets, comme des panneaux de signalisation. Un Jetson Nano et un kit JetBot AI de SparkFun (fig. 2), qui contient une caméra à 8 mégapixels, forment la plateforme matérielle.

Notre premier objectif est de pouvoir distinguer l'image de la carte UNO n°1 de la carte n°2 et du verso d'une carte. Pour ce faire, comme nous l'avons déjà mentionné, nous devrons entraîner le réseau neuronal en utilisant des images des objets à reconnaître. Pour cela, découvrons l'acquisition de données et l'interface d'Edge Impulse.

Un projet Edge Impulse

Il faut d'abord s'inscrire pour pouvoir utiliser Edge Impulse. Nous utilisons ici la version gratuite de la plateforme, qui présente certaines limitations d'utilisation (nous y reviendrons). Après l'enregistrement, un nouveau projet peut être créé ; nous sommes alors accueillis par un assistant (**fig. 4**).

Lorsqu'on nous demande quel type de données nous souhaitons traiter, nous répondons par l'option *Images*, puisque ce sont les images des cartes UNO qui nous intéressent pour ce test. Dans ce cas, les cartes UNO doivent être classées (*Classify a single object*) car, dans un premier temps, nous ne voulons traiter qu'une seule carte et déterminer de laquelle il s'agit, comme le montre la **figure 5**. Le projet est maintenant configuré avec les paramètres de reconnaissance. La prochaine étape consiste à construire le jeu de données avec lequel nous pouvons commencer l'entraînement.

Acquisition de données

Dans ce projet, nous produirons nous-mêmes les données pour l'entraînement du réseau neuronal. Pour la reconnaissance d'images, il existe des jeux de données contenant des milliers d'images déjà triées dans certains groupes ; selon votre application, il peut être judicieux d'utiliser ce type de données. Nous n'avons pas trouvé de jeu de données de cartes UNO, nous devons donc en créer un. C'est un bon exercice pour passer en revue les étapes nécessaires pour créer soi-même un jeu de données d'images et montrer comment éviter certains pièges.

Pour capturer les images, l'idéal est d'utiliser la même caméra que celle qui sera utilisée ultérieurement pour la reconnaissance d'images. Ainsi le réseau neuronal apprend en même temps les particularités de la caméra. Afin de collecter les données à l'aide du Nvidia Jetson, nous pouvons installer un client pour Edge Impulse. Celui-ci peut alors envoyer les images (et l'audio si nécessaire) à Edge Impulse.

Pour installer le client, suivez les instructions [5] fournies par Edge Impulse. Pour le Nvidia Jetson, il est nécessaire de se connecter à la carte à l'aide du protocole SSH, puis d'entrer les commandes via une console. Autre solution : vous pouvez connecter une souris, un clavier et un moniteur et entrer les lignes de commande directement dans le système Linux qui fonctionne sur le Jetson Nano.

Lorsqu'une liaison SSH a été établie ou qu'une console est ouverte, il suffit de saisir les commandes figurant dans les instructions. Une connexion internet est nécessaire pour l'installation. Si le Jetson Nano n'est pas encore connecté à l'internet, commencez par cette étape.

Ensuite, nous entrons

```
wget -q -O - https://cdn.edgeimpulse.com/build-system/jetson.sh | bash
```

dans la console pour démarrer l'installation de *edge-impulse-linux*. L'installation prend un certain temps. Une fois qu'elle est terminée, entrez `edge-impulse-linux -clean` dans la console. L'outil requiert nos informations d'accès et le périphérique audio et vidéo que nous utiliserons pour l'acquisition des données. Une fois que tout est configuré, le Jetson Nano nouvellement configuré apparaît sur la page d'Edge Impulse sous la rubrique *Devices* (**fig. 6**). Grâce à cela, nous pouvons maintenant obtenir des images à partir de l'appareil photo.

Sur la page web d'Edge Impulse, nous pouvons les capturer à l'aide de l'option de menu *Data aquisition* (**fig. 7**). Comme une source audio a également été configurée ici, il faut d'abord commuter le capteur sur *Camera*

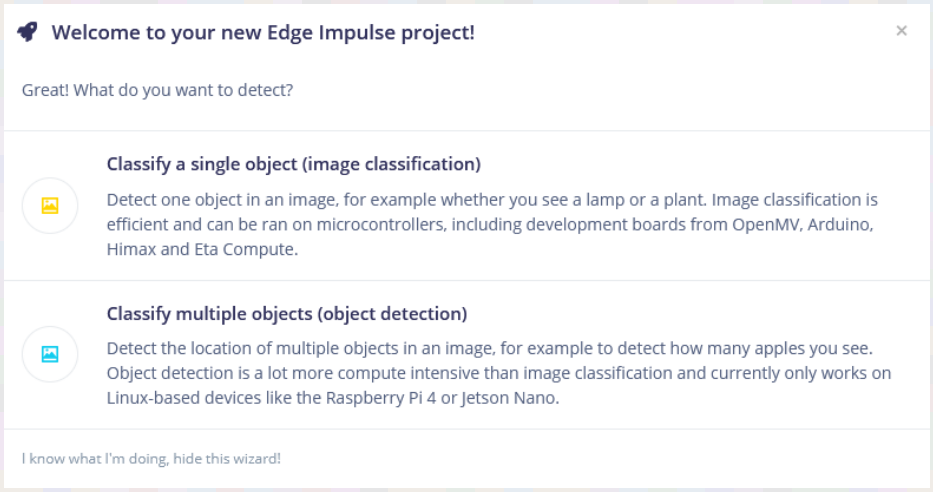

Figure 5. Sélectionner la classification des objets.

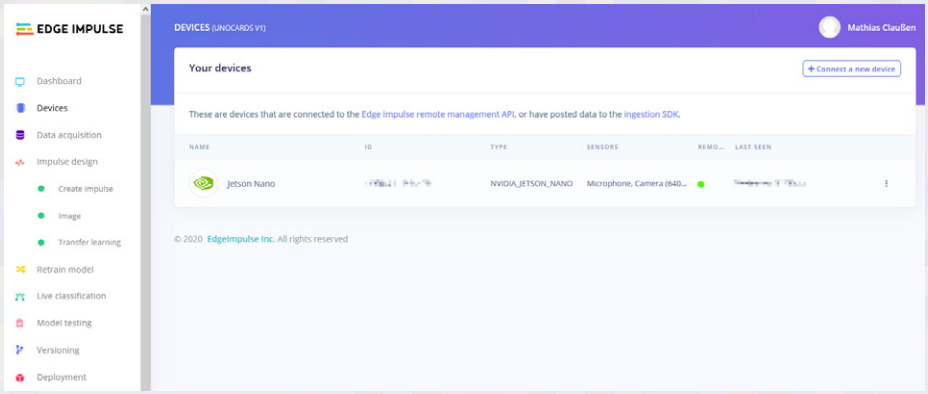

Figure 6. Définir JetBot comme dispositif destiné à acquérir les données.

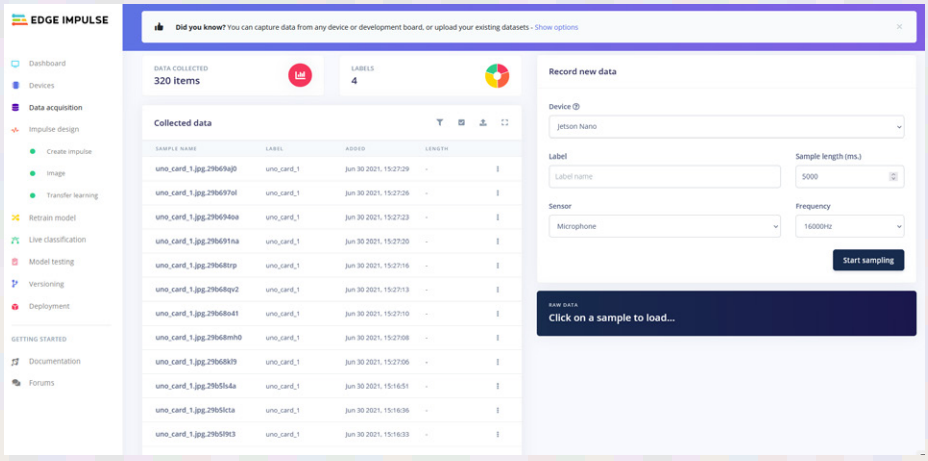

Figure 7. Acquisition des données avec Edge Impulse.

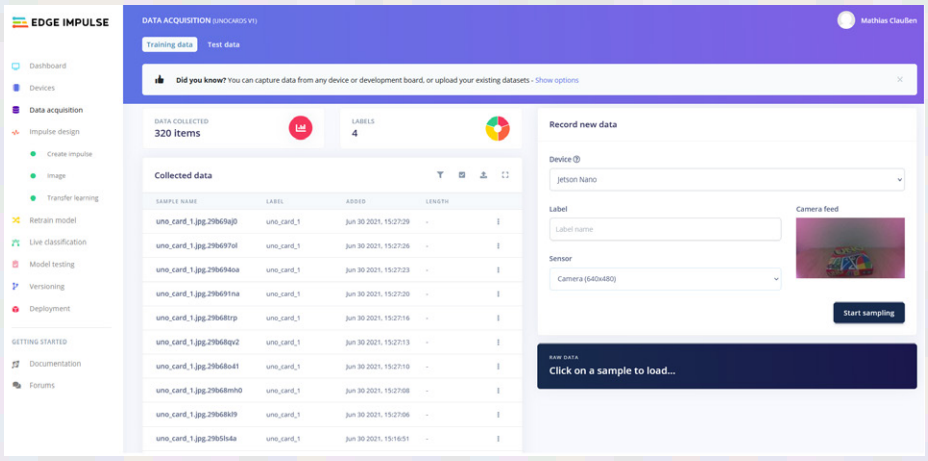

Figure 8. Configuration de la caméra.

(640x480). Le flux de la caméra (*Camera feed*) apparaît alors, montrant les images en direct (**fig. 8**). À l'aide de la mention *Label* (libellé), nous pouvons maintenant indiquer le type de chaque image (« class ») (c.-à-d. aucune carte, carte n°1, carte n°2 ou le verso de la carte).
Passons maintenant à la partie la plus ennuyeuse – l'acquisition des données de chaque carte selon différentes orientations. Au début, environ 80 à 100 images doivent être prises pour chaque carte que nous reconnaîtrons plus tard. Le libellé associé aux images enregistrées doit également être attribué en même temps.

Entraînement initial

Une fois toutes les images prises, il est temps de procéder à la première session d'entraînement. Avec *Impulse design* (**fig. 9**), nous pouvons maintenant définir l'impulsion (c'est-à-dire le mode opérationnel du réseau neuronal). Pour la taille de l'image, nous utilisons 160 × 160 pixels et l'option *Fit shortest axis* (adapter l'axe le plus court) dans le champ déroulant *Resize mode* (mode de redimensionnement). Comme aucune couleur ne sera évaluée, nous pouvons sélectionner *Grayscale* (niveaux de gris) (**fig. 10**). Après avoir cliqué sur *Save parameters* (enregistrer les paramètres), les classes de reconnaissance peuvent être créées sur la page suivante.
Le nuage de points affiché dans l'explorateur de caractéristiques montre la distribution de nos images sur la base d'un premier entraînement approximatif. Il permet d'identifier les données incorrectement étiquetées. La proximité des points individuels (*cluster*) donne une indication de la difficulté qu'a le réseau neuronal à distinguer les cartes les unes des autres (**fig. 11**).
Les paramètres d'entraînement peuvent maintenant être définis dans *Transfer learning* (**fig. 12**). Les limites de la version gratuite ont été mentionnées au début et nous les rencontrons maintenant. Les serveurs d'Edge Impulse allouent 20 min de temps de calcul pour le processus d'entraînement. Pour nos quatre classes, ce temps est suffisant pour obtenir des résultats utiles. Si le processus d'entraînement nécessite plus de 20 min, le processus est simplement annulé.
Le nombre de cycles d'entraînement affecte sa qualité, ainsi que le temps nécessaire. Dans le cas présent, la valeur est fixée à 20 pour l'apprentissage relatif aux données ; l'augmentation du nombre de cycles de 20 à 50 améliore le résultat. Comme

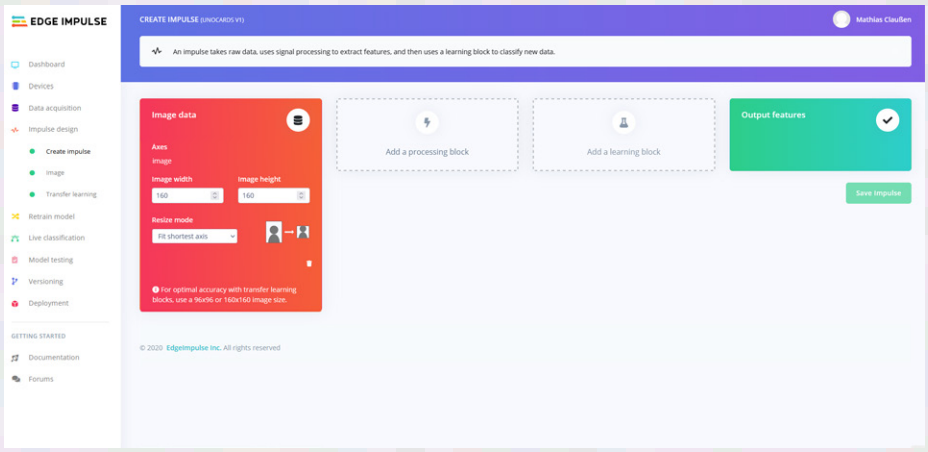

Figure 9. Créer une nouvelle impulsion.

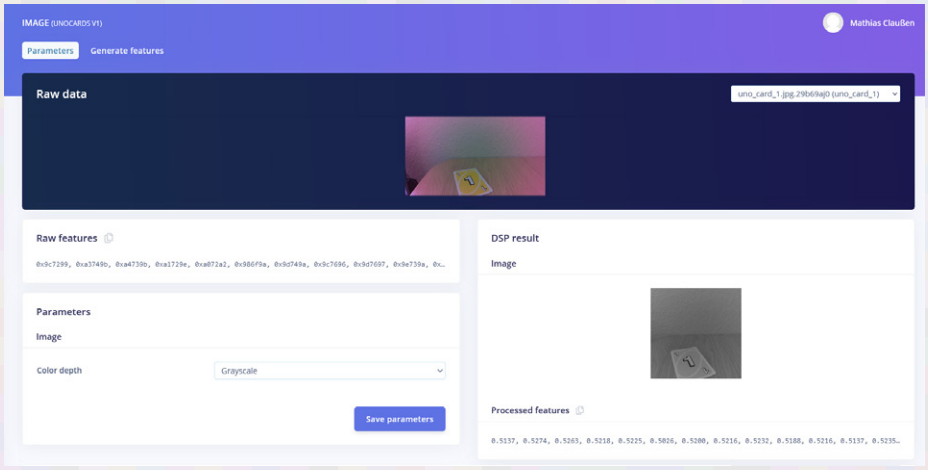

Figure 10. Travailler en niveaux de gris.

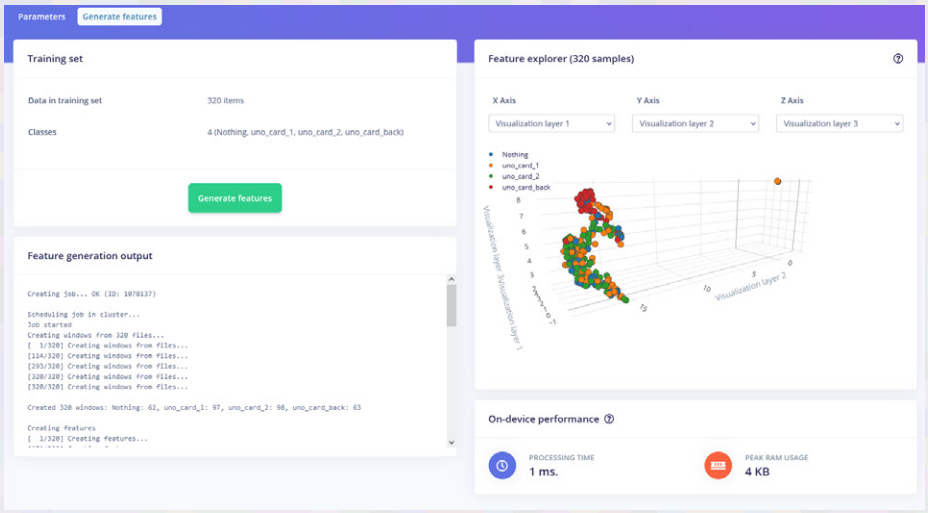

Figure 11. Génération des caractéristiques du réseau neuronal.

l'ensemble de données utilisé ici est très facile à gérer, l'augmentation des données peut servir à améliorer la reconnaissance. Cette augmentation des données déforme, étire et fait pivoter les images afin d'élargir artificiellement l'ensemble de données. Cela peut améliorer la confiance dans la reconnaissance.

Une fois l'entraînement terminé, des statistiques sont calculées (**fig. 13**) qui montrent le bon fonctionnement de notre réseau neuronal. Le réseau qui vient d'être créé permet d'effectuer un test sur le Jetson.

Premier test à l'aide du Jetson

Une fois le réseau neuronal entraîné, nous devrons ouvrir à nouveau une console sur le Jetson Nano. Le moyen le plus simple est de se connecter au Nano à l'aide de SSH. La commande `edge-impulse-linux-runner --clean` permet de lancer un assistant qui demande nos données d'accès. Si plusieurs projets ont été créés, il faut sélectionner celui qui doit être testé.

Une fois que tout a été configuré, la console produit une liste de classification pendant que la caméra fonctionne (**fig. 14**). La valeur numérique indiquée après chaque classe individuelle indique le niveau de confiance avec lequel la classe apparaît dans l'image qui vient d'être enregistrée, avec 1,0 comme maximum (c'est-à-dire 100 % de certitude) et 0,00 à titre de minimum.

Si nous plaçons maintenant la carte n°1 face à la caméra, nous pouvons voir que le niveau de confiance change et qu'un 1 est reconnu avec un niveau de confiance de 0,9420 (c.-à-d. un très haut degré de certitude).

Se fier entièrement au résultat donné par la console n'est pas très pratique. Il se peut qu'une carte ne soit pas reconnue correctement et que vous vouliez savoir ce que le Jetbot voit réellement. Pour cela, l'outil *edge-impulse-linux-runner* ne se contente pas d'exécuter le réseau neuronal, il fournit également un serveur web accessible via le port 4912 (**fig. 15**). Sur la page web, vous pouvez suivre en direct quelle carte est actuellement reconnue ou, comme le montre la figure 15, si elle n'est pas reconnue correctement. Il est également possible de produire de nouvelles images, avec lesquelles le réseau neuronal peut être réentraîné. Pour ce faire, utilisez CTRL+C pour arrêter *impulse-linux-runner*. Avec `edge-impulse-linux`, Jetbot apparaît à nouveau comme une source de données dans le portail web Edge

Impulse et de nouvelles images peuvent être enregistrées.

Deuxième cycle d'entraînement

Avec de nouvelles données valides supplémentaires, vous pouvez lancer une nouvelle procédure d'entraînement. Plus il y a de données ajoutées à la catégorie avec laquelle le réseau neuronal actuel a des difficultés, plus le résultat sera satisfaisant. Avec `edge-impulse-linux`, le JetBot redeviendra le collecteur de données ; nous pouvons prendre des images des cartes à jouer et les étiqueter de manière appropriée. Lorsque l'acquisition de nouvelles données est terminée, le processus d'entraînement peut être de nouveau lancé avec *Retrain Model* pour réutiliser les paramètres précédents avec les nouvelles données (**fig. 16**). Une fois cela fait, nous pouvons tester la nouvelle version du réseau sur le Jetson. En utilisant `edge-impulse-linux-runner`, le nouveau réseau neuronal sera téléchargé et exécuté sur JetBot. Nous pouvons maintenant utiliser un navigateur web pour voir en temps réel comment les images des cartes sont classées avec une plus grande confiance (**fig. 17**).

De l'entraînement à l'application

Une fois les données enregistrées et le réseau neuronal entraîné, la question se pose de savoir comment l'utiliser dans une application spécifique. Edge Impulse propose des kits de développement logiciel pour prendre en charge différents langages de programmation. Si on veut coder en langage C / C++, le JetBot rencontre alors un problème concernant la commande de la caméra et la bibliothèque OpenMV utilisée. Aucune image ne peut être capturée à l'aide de la caméra avec cette bibliothèque. (La capture d'images via Edge Impulse décrite ci-dessus n'est pas affectée ; elle ne semble pas utiliser OpenMV.)

Le processeur graphique du Jetson Nano n'est pas non plus utilisé pour le traitement des images, tous les calculs sont donc effectués par l'unité centrale. Il n'y a peut-être pas de perte majeure pour les petites cartes comme le Raspberry Pi, mais avec le Jetson Nano, une grande partie de la capacité de calcul de l'IA est ainsi perdue.

Le travail d'entraînement effectué pour le réseau n'a pas été vain. Il est possible d'exporter le réseau neuronal entraîné sous la forme d'une bibliothèque indépendante de la plateforme, ce qui lui permet de fonctionner sur d'autres systèmes (voir

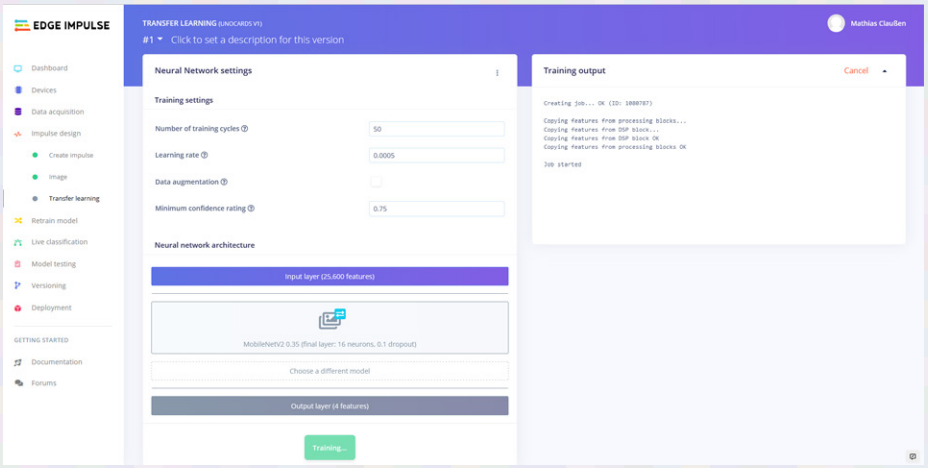

Figure 12. Définir les paramètres pour l'entraînement.

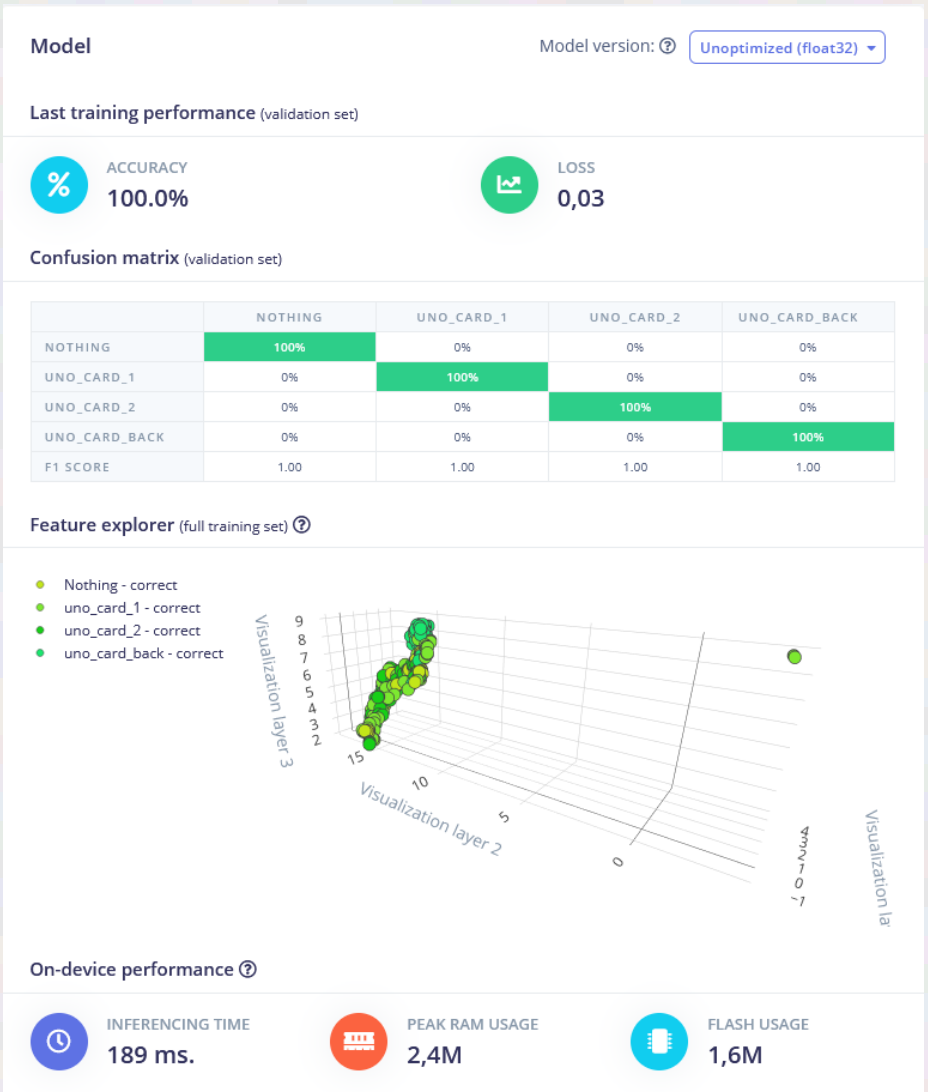

Figure 13. Résultats de l'entraînement.

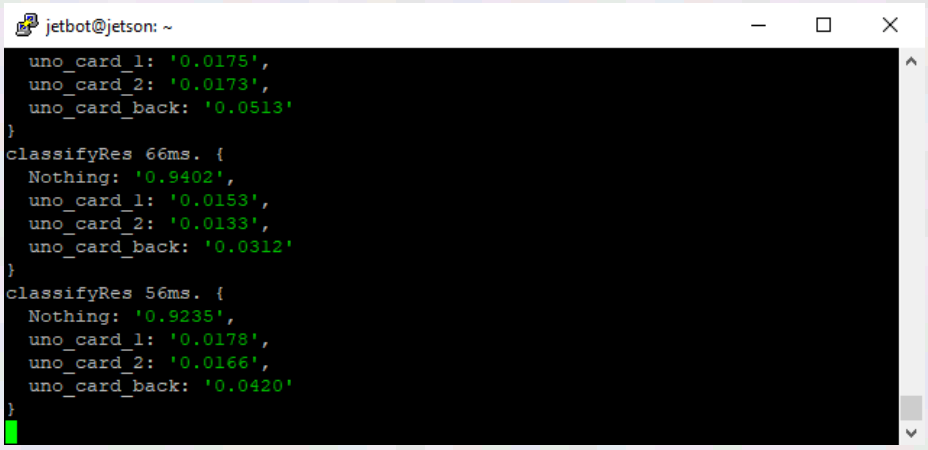

Figure 14. Affichage de la reconnaissance d'images sur la console.

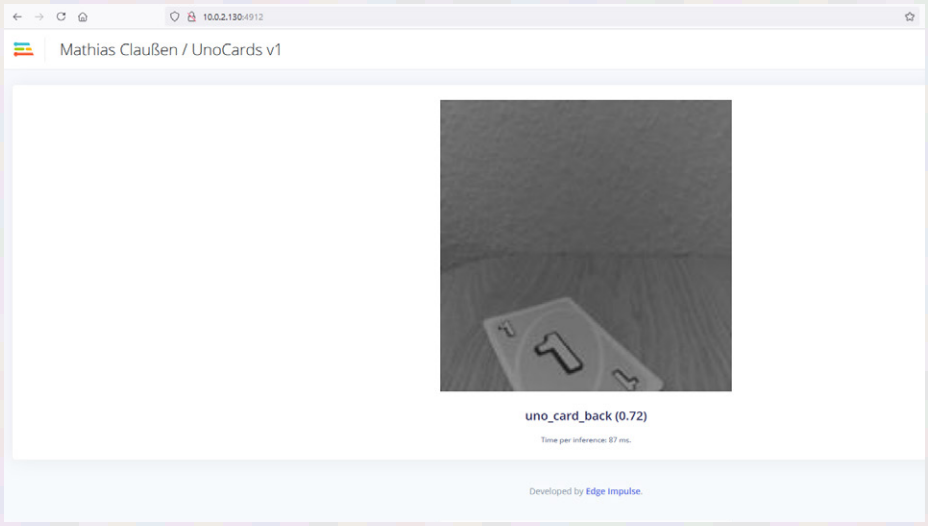

Figure 15. Visualisation en temps réel de la reconnaissance d'images à l'aide d'un navigateur.

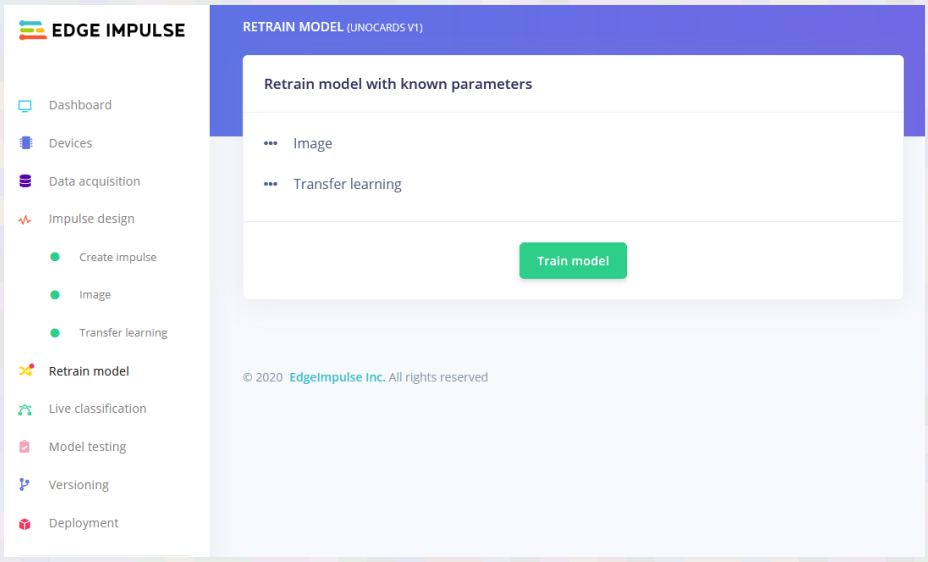

Figure 16. Nouvel entraînement avec davantage de données.

téléchargement [6]). Il n'est pas nécessaire qu'il s'agisse d'un Raspberry Pi ; un ESP32 avec une caméra est tout aussi approprié pour utiliser le réseau que nous avons entraîné.

Malheureusement, les données brutes enregistrées avec Edge Impulse ne sont disponibles que là. Les images que nous avons étiquetées ne peuvent pas être exportées vers d'autres plateformes ou vers un système local pour être traitées. C'est un compromis que nous devrons accepter pour utiliser la plateforme Edge Impulse.

Ceci n'est pas un filigrane, c'est un cheval…

L'évolution nous a dotés d'une capacité sophistiquée à reconnaître des formes. Nous pouvons apprendre par ex. à quoi ressemble un cheval. Si l'on nous montre un certain nombre de photos de ces animaux, nous apprenons que s'il a quatre pattes, des sabots, une certaine taille et des proportions spécifiques, il s'agit probablement d'un cheval. Au cours de notre vie, notre propre définition du cheval s'affine (par ex. un âne n'est pas un cheval, même s'il en partage de nombreux attributs).

Avec un traitement d'images classique, on apprend à un algorithme les caractéristiques à rechercher dans les images pour reconnaître les chevaux (par ex. sabots, yeux, crinière, couleur, proportions). Toutefois, cette méthode a ses limites lorsqu'il s'agit de généraliser la détection des chevaux. C'est ici qu'entrent en jeu les réseaux neuronaux. Il suffit de leur donner un grand nombre d'images (plusieurs milliers) et de laisser aux algorithmes qui forment le réseau neuronal le soin d'identifier un cheval. Cette approche semble plausible. Qu'est-ce qui peut bien se passer ?

Comme l'a mentionné Stuart Cording lors du webinaire « Get to know the NVIDIA Jetson Nano Developer Kit » [7], alors qu'il travaillait sur un projet d'IA, il a téléchargé une série d'images trouvées via un moteur de recherche afin de les utiliser comme données pour identifier un cheval sur une photo. Malheureusement, les images contenaient un filigrane. Cela signifie que le réseau neuronal identifiait l'image comme étant un cheval s'il détectait un filigrane dans l'image traitée.

Quelque chose de similaire s'est produit lorsque nous avons tenté pour la première fois de construire un ensemble de données avec les cartes UNO. Une partie d'une main a

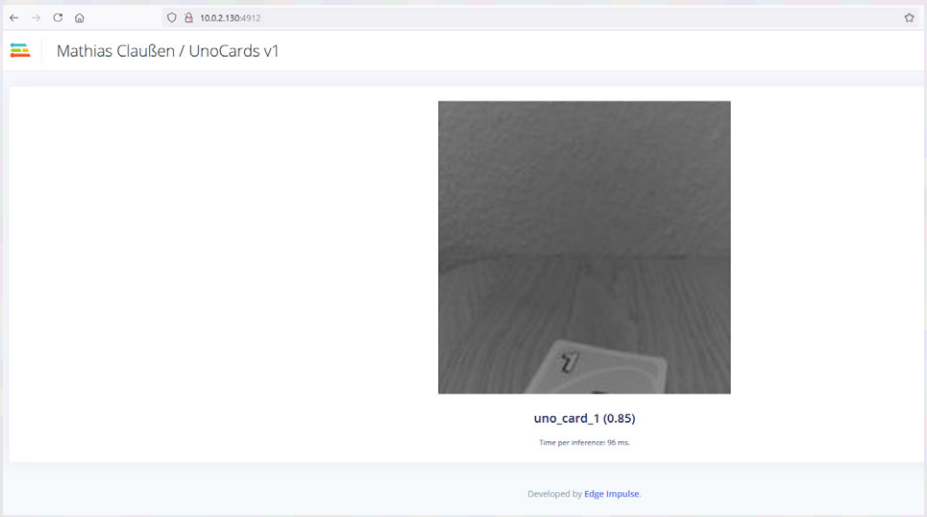

Figure 17. Image en temps réel indiquant une plus grande confiance dans la reconnaissance.

été capturée dans certaines des images. Ainsi, la main est devenue l'une des caractéristiques les plus déterminantes pour distinguer les cartes numérotées 1 et 2. Nous avons dû jeter l'ensemble de données original et en créer un nouveau en prenant soin de supprimer toute information non pertinente. Il est important de faire attention lors de l'enregistrement des images à l'ensemble de données. En effet, un filigrane ou toute autre caractéristique non pertinente capturée par inadvertance dans les clichés faussera le processus de reconnaissance.

Résumé et suites possibles

L'entraînement de notre propre réseau neuronal à l'aide d'Edge Impulse et du Jetson Nano est une procédure assez simple. L'étape suivante consiste à intégrer ce réseau dans une application pratique. Le Jetson Nano n'est pas la plateforme matérielle la mieux adaptée pour cette application ; certaines de ses ressources sont inutilisées, et il y a quelques problèmes logiciels qui empêchent la caméra de fonctionner directement avec le SDK Edge Impulse dans certaines applications.

Si vous voulez explorer l'IA et la reconnaissance d'images de manière simple, vous devriez tout de même essayer Edge Impulse. Il n'est pas nécessaire qu'il tourne sur un Jetson Nano. Un Raspberry Pi 4 fera tout aussi bien l'affaire.

Et ensuite ? Le Jetson Nano dispose d'une puissance suffisante pour reprendre ce qu'Edge Impulse a effectué ici. L'étape suivante pourrait consister à créer un réseau neuronal capable de reconnaître les cartes UNO et pouvant fonctionner de manière indépendante sur le Jetson Nano.

210318-B-01

PRODUITS

- Module Jetson Nano de Nvidia
  www.elektor.fr/nvidia-jetson-nano-developer-kit
- Kit AI JetBot v2.1 de SparkFun (sans kit de développement NVIDIA Jetson Nano inclus)
  www.elektor.fr/sparkfun-jetbot-ai-kit-v2-1-without-nvidia-jetson-nano-developer-kit
- Carte de développement ESP32-Cam-CH340
  www.elektor.fr/19333

Des questions ? Des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Conception et texte : **Mathias Claußen**
Rédaction : **Jens Nickel**
Mise en page : **Harmen Heida**
Traduction : **Pascal Godart**

LIENS

[1] M. Claussen, « Traitement d'images avec le kit Jetson Nano de Nvidia (1ère partie) », Elektor, 09-10/2021 : www.elektormagazine.fr/210318-04
[2] M. Claussen, « Faire du café grâce au MAX78000 et à une pincée d'IA », site elektormagazine.fr, 04/2021 : www.elektormagazine.fr/articles/faire-du-cafe-grace-au-max78000-et-a-une-pincee-ia-2e-partie
[3] Edge Impulse : www.edgeimpulse.com
[4] A. Garrapa, « MCU Machine Learning With Edge Impulse », Elektor Industry, 02/2021 : www.elektormagazine.com/magazine/elektor-238/59631
[5] Edge Impulse – Nvidia Jetson Nano : https://docs.edgeimpulse.com/docs/nvidia-jetson-nano
[6] Téléchargement du réseau entraîné : www.elektormagazine.fr/210318-B-04
[7] Webinaire d'Elektor, « Meet the Engineers (Part 2): Get to Know the NVIDIA Jetson Nano Developper Kit », 05/2021 : https://youtu.be/KGazuosH0xw