
Image générée par IA à la demande de l'auteur

# L'INTELLIGENCE *en périphérie*

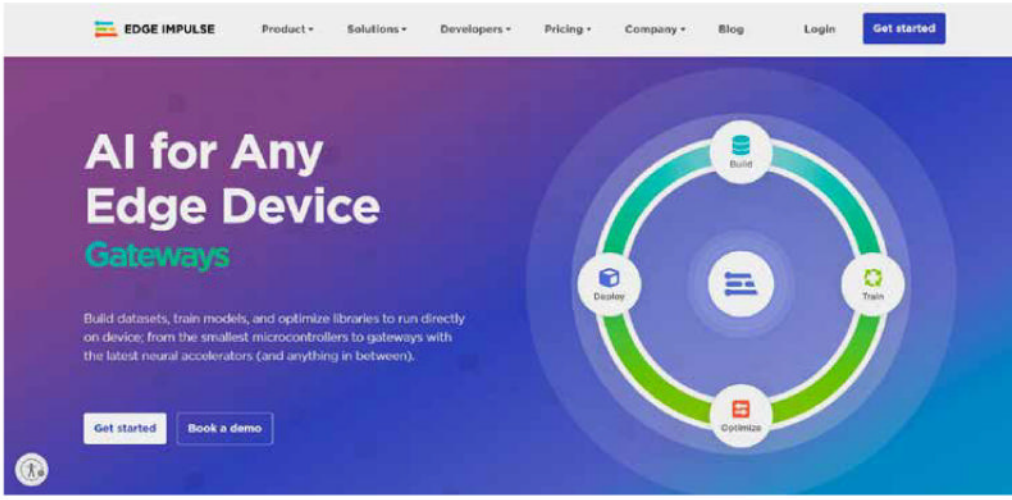
## Une IA à l'échelle de nos réseaux

Quand on parle d'IA (Intelligence Artificielle), on imagine d'énormes data centers et des ordinateurs surpuissants, extrêmement énergivores: c'est vrai pour faire tourner des IA telles que ChatGPT. Mais il est aussi possible de faire tourner des IA dont les capacités sont restreintes mais suffisantes pour ce qu'on veut faire en modélisme ferroviaire, sur des cartes à microcontrôleur actuelles. Ce n'est pas de la science-fiction, cela existe déjà!

Texte: Christian Bézanger, Illustrations: Edge Impulse (sauf mention contraire)

C'est ce qu'on appelle **l'Intelligence en périphérie** (**edge AI** en anglais) et cela présente bien des avantages. Cette IA n'a pas besoin de se connecter au cloud pour fonctionner, ce qui permet déjà d'économiser la planète. Elle est donc parfaitement adaptée pour des projets en zone faiblement couverte par internet et elle a une réponse plus rapide puisqu'on économise la latence de la connexion. De plus, elle permet une confidentialité accrue puisque justement les données de l'utilisateur ne transitent plus par ▸▸


1




2

▸▸ le cloud. L'Intelligence en périphérie est ainsi au plus près de l'action à réaliser, permettant d'économiser d'importantes ressources de bande passante.

En effet, l'edge AI combine à la fois l'Intelligence Artificielle et le calcul en périphérie, ce qui permet de développer des modèles d'IA directement sur des appareils en bout de réseau plutôt que dans des centres de données cloud spécialisés. Ces modèles d'IA s'exécutent localement, directement sur l'appareil (carte à microcontrôleur), et le temps d'inférence peut être réduit à quelques millisecondes, rendant les applications temps réel, tout en augmentant leur résilience et leur fiabilité. La tâche à faire réaliser par l'Intelligence en périphérie doit rester relativement simple, mais dans l'avenir les puces seront dotées de ressources encore plus puissantes pour accélérer les calculs complexes liés à l'IA (Processeur graphique GPU par exemple). Voici un exemple de ce qu'on peut imaginer faire dès à présent: inclure une IA capable de reconnaître une locomotive à vapeur d'une électrique. Si l'IA reconnaît une locomotive à vapeur, elle l'arrête en gare pour refaire le plein d'eau. Une belle animation en perspective qui constitue une application intéressante dans notre domaine ! D'autres applications existent comme commander son réseau à la voix. Tout cela va se développer dans l'avenir et sera sans doute inclus dans les centrales DCC de demain ou les logiciels de gestion de réseaux comme Train Controller.

## Différence entre IA et algorithme

Un **algorithme** est une méthode pour résoudre un problème. Par exemple, l'algorithme ci-dessous est prévu pour déterminer si un nombre entier est pair ou impair :

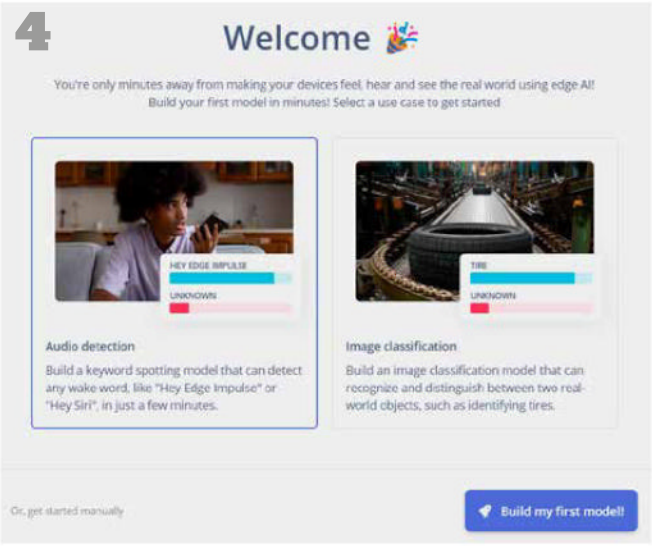
- *Prendre un nombre entier naturel*
- *Soit R le reste de la division entière (division Euclidienne) du nombre par 2*
- *Si R = 0, afficher « Le nombre est pair »*
- *Si R = 1, afficher « Le nombre est impair »*

On peut exprimer cet algorithme dans un langage informatique (Fortran, Basic, C, Python, etc.). Il fait appel à des lois mathématiques ou logiques et il est fiable à 100 % ; si vous entrez dix fois le même nombre, vous obtiendrez dix fois le même résultat. Hélas, il n'y a pas d'algorithme capable de reconnaître un chat dans une image ; pour cela, il faut faire intervenir une IA spécialement entraînée pour reconnaître un chat. C'est ce qu'on appelle le **Machine Learning** (ML) et des outils sont aujourd'hui accessibles à tous dans ce domaine. Le résultat n'est pas garanti à 100 %, certaines images de chat ne seront pas reconnues alors que des images qui ne montrent pas de chat seront interprétées comme des images de chats. On peut tout de même se rapprocher d'un taux de réussite élevé (plus de 90 %) si les données qui servent à entraîner l'IA sont suffisamment nombreuses et de qualité.

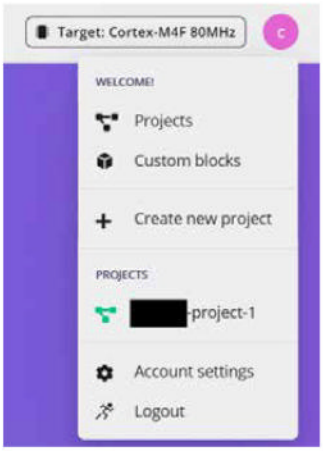
## Quelles sont les cartes à microcontrôleur concernées par l'IA en périphérie ?

Ce sont des cartes équipées de microcontrôleur 32 bits avec une mémoire suffisante pour recevoir le modèle d'IA embarquée, typiquement des cartes milieu de gamme dotée de 512 Ko de RAM. De telles cartes existent depuis quelques années et sont équipées d'ESP32 (Espressif), de RP2040 (Raspberry Pi) ou de STM32 (STMicroelectronics). La toute nouvelle carte Arduino **UNO-Q (figure 1)** est particulièrement adaptée à l'IA embarquée, car elle combine la puissance d'un microprocesseur quad-core QRB2210, compatible avec Linux Debian, et offrant une accélération IA et graphique (GPU pour **Graphic Processor Unit**), et un microcontrôleur STM32 assurant le contrôle en temps réel de capteurs et d'actionneurs. Pour la programmer, l'application **Arduino AppLab** prolonge cette philosophie de l'IA embarquée et donne aux utilisateurs la possibilité de créer leurs propres applications tout en codant dans le même environnement intégré. Linux et l'AppLab sont préinstallés sur la carte qui

3



4



5

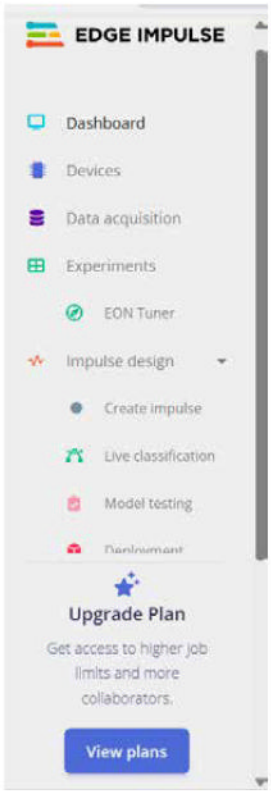
peut donc s'utiliser comme un ordinateur si on rajoute clavier, souris et un écran (mode SBC pour **Single Board Computer**).
Vous n'aurez donc pas à dépenser une fortune pour arriver à programmer ces cartes avec un modèle d'IA capable de réaliser des tâches relativement simples. Sachez qu'Arduino a passé un accord en octobre 2025 pour rejoindre Qualcomm Technologies qui possède aussi Edge Impulse (voir plus loin), créant ainsi une synergie importante pour les makers dans le monde de l'IA en périphérie.

## Comment entraîner une IA en périphérie ?

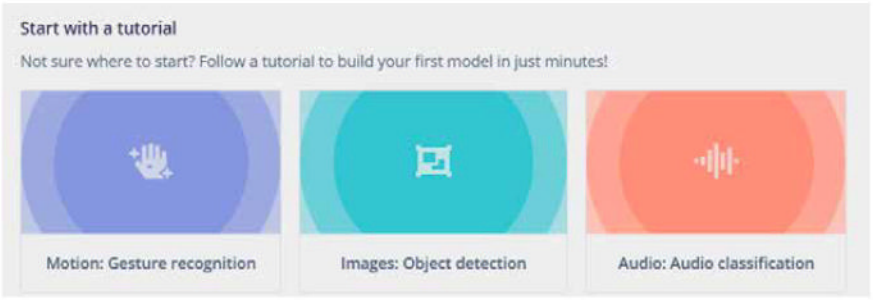
Le processus d'entraînement d'une IA est relativement complexe : acquisition et filtrage des données, répartitions de celles-ci en groupe d'entraînement et de test, réseau neuronal, etc. Ce n'est pas forcément simple mais il existe une plate-forme en ligne qui s'occupe de tout et produit des bibliothèques pour cartes Arduino, qui vous serviront à programmer vos cartes. Le site s'appelle **Edge Impulse** (une filiale de Qualcomm) et propose tous les outils nécessaires pour produire l'edge AI, regroupés et accessibles en ligne sous le nom d'**Edge Impulse Studio (EIS)**. La **figure 2** montre le site Edge Impulse (en Février 2026) ; l'ouverture d'un compte est gratuite et une fois le compte ouvert, vous accédez à tous les outils et tutoriels proposés pour faire de l'Intelligence en périphérie. Vous pouvez choisir entre la **reconnaissance audio** ou la **reconnaissance d'images** ou bien encore le **suivi de signaux générés par des capteurs**. Edge Impulse permet donc une maîtrise de la chaîne complète de développement de l'IA en périphérie et ce qui semblait extrêmement complexe autrefois devient aujourd'hui plus accessible. D'autres tutoriels existent sur des sites comme YouTube par exemple et c'est toute une large communauté d'utilisateurs que vous rejoindrez et qui pourront vous aider le cas échéant.

## Que peut réaliser l'IA en périphérie en modélisme ?

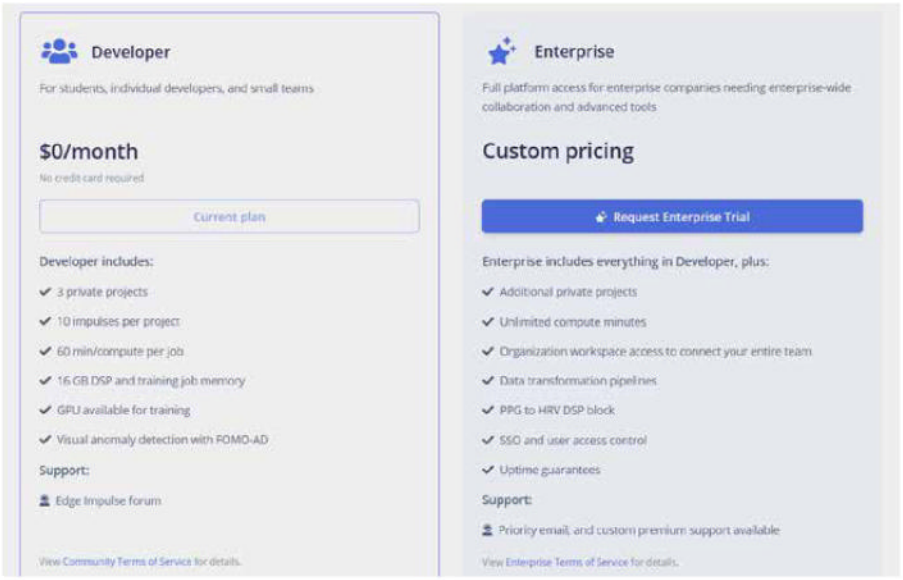
Dans notre domaine, on peut entraîner une IA en périphérie pour reconnaître la voix humaine (reconnaissance audio) et ainsi commander le réseau à la voix. L'IA peut aussi être entraînée à la reconnaissance d'images, comme l'a suggéré l'exemple un peu plus haut ; c'est de nombreuses possibilités qui s'ouvrent alors pour créer des applications de **Vision** ▸▸




6




7




8

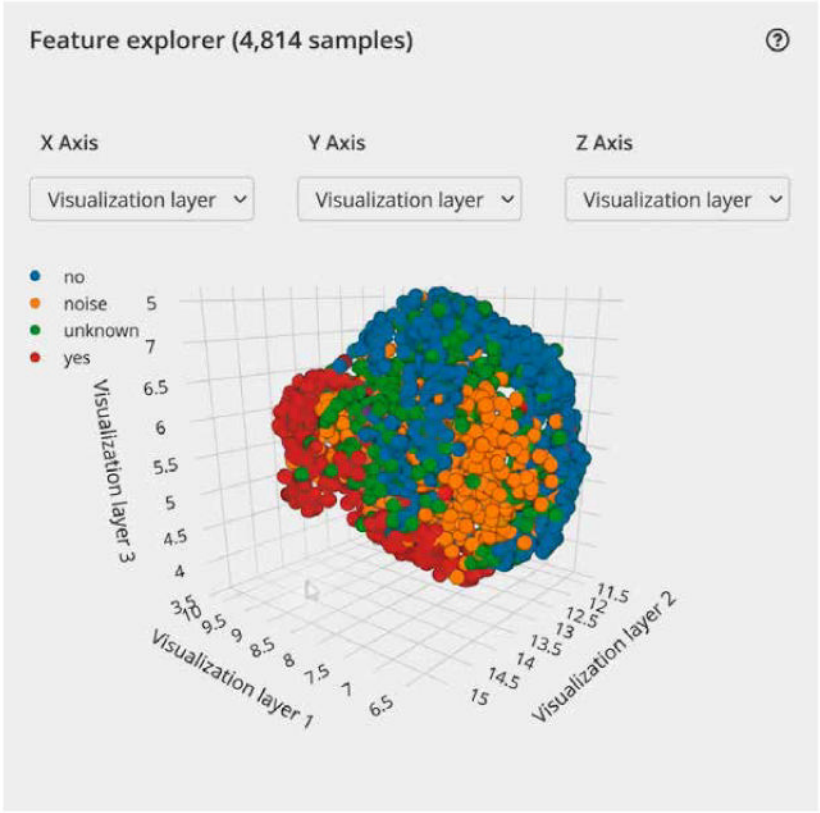
▸▸ **Language Models** (VLM) comme la connaissance de la position exacte des trains sur le réseau ou bien l'identification des trains eux-mêmes, avec le traitement qui s'impose. Mais l'IA en périphérie peut aussi analyser les signaux délivrés par les capteurs situés sur le réseau ou dans les locomotives et déterminer si tout est correct, voire faire de la maintenance prédictive comme on le fait dans l'industrie. L'IA en périphérie va très certainement s'améliorer dans l'avenir et on peut imaginer dès maintenant que les centrales DCC du futur se chargeront de modifier les CV des locomotives pour obtenir des réponses encore plus réalistes dans les tractions des trains en fonction des masses ou des conditions atmosphériques, et tout cela sans que l'utilisateur ait à se préoccuper de quoi que ce soit. L'IA en périphérie est donc une composante bien réelle, appelée à évoluer, qu'il ne faut pas négliger aujourd'hui si on ne veut pas être dépassé le moment venu.

## Découvrir Edge Impulse Studio

Il n'est pas possible au sein d'un court article de décrire de façon exhaustive toutes les possibilités d'un logiciel comme Edge Impulse Studio ; c'est pourquoi nous vous invitons à découvrir par vous-même et nous nous limiterons seulement à un rapide tour d'horizon comme le montrent les figures suivantes. La **figure 3** montre le principe de l'Intelligence en périphérie : acquisition de données, constitution d'un ensemble de qualité pour entraîner le modèle, entraînement du modèle d'IA, optimisation du modèle, estimation de ses possibilités, intégration à la carte à microcontrôleur qui peut alors produire de nouvelles données et la boucle est bouclée ! Commencez par créer un compte gratuit. Lorsque vous accédez à Edge Impulse Studio **(EIS)**, une fenêtre pop-up vous propose de découvrir deux tutoriels et vous pouvez vous laisser guider **(figure 4)**. L'accès à votre compte se fait par l'icône en haut à droite **(figure 5)**. La partie gauche **(figure 6)** permet d'accéder à toutes les fonctionnalités du logiciel et la partie centrale concerne votre projet (tableau de bord ou **dashboard**). À tout moment vous pouvez retrouver des tutos dans les trois principales catégories d'IA (reconnaissance de mouvement, d'audio ou d'images) comme le montre la **figure 7**. L'utilisation d'Edge Impulse est gratuite mais évidemment limitée ; vous pouvez à tout moment choisir un plan payant proposant d'autres options **(figure 8)**.

## Le flux de travail

Le flux de travail (**workflow** en anglais), ou si vous préférez la méthode pour entraîner une IA en périphérie est pratiquement la même pour toutes les catégories d'IA. On commence par acquérir un jeu de données de qualité qui servira d'une part à entraîner l'IA (80 % des données) et d'autre part à la tester (20 % restant). C'est un peu comme un examen sous forme QCM : vous disposez de questions pour vous entraîner mais le jour de l'examen, les questions sont différentes pour voir si vous avez réellement compris et pas simplement mémorisé questions et réponses. L'IA ne doit pas être testée avec les mêmes échantillons que pour l'entraînement.
Ces données représentent plusieurs classes : par exemple, en reconnaissance audio, celle du mot-clé, celle de mots inconnus et celle du bruit. Le modèle fonctionnera d'autant mieux si ces classes sont équilibrées en nombre d'échantillons. EIS propose aussi de télécharger des données comme celles du bruit, ce qui peut éviter de longs enregistrements. Ce jeu de données doit être




**9**

prétraité avant de servir à l'IA et le mieux est de faire confiance à EIS qui propose des méthodes de filtrage adaptées aux types de données (audio, images, etc.). Pour cela, une fois l'impulse créée (avec Create impulse), on peut ajouter un **bloc de traitement** (Add a processing block) et choisir la méthode la plus appropriée. Il faut se fier aux tutoriels pour choisir surtout au début où on n'a guère d'expérience. Par exemple, le traitement MFCC est conseillé pour la voix humaine alors que d'autres méthodes sont possibles pour d'autres types d'audio. Ces traitements vont produire des **spectrogrammes** dont la signature sera différente selon la donnée. Vient ensuite l'entraînement de l'IA avec ces données pour qu'elle reconnaisse les différentes classes possibles (objet, inconnu ou bruit) avec une probabilité de réussite la plus élevée possible. Pour cela, on ajoute un **bloc d'apprentissage** et là aussi plusieurs choix sont possibles. Adoptez les choix préconisés dans les tutoriels, par exemple un traitement par réseau de neurones multicouches (en anglais **NN** pour **Neural Network**). Il s'agit d'un modèle d'apprentissage inspiré du cerveau humain. Chaque « neurone » reçoit des entrées (spectrogrammes MFCC par exemple) et applique un traitement avant de transmettre son résultat à la couche de neurones suivante (on parle dans ce cas d'apprentissage profond par opposition à l'apprentissage classique). Finalement, l'entrée est transformée progressivement en probabilités que l'échantillon appartienne aux différentes classes.

On peut modifier les paramètres pour configurer le réseau de neurones et on peut à tout moment revenir en arrière afin d'optimiser le processus. EIS permet d'évaluer le modèle sur la carte à microcontrôleur qu'on veut utiliser et, lorsque tout paraît correct, génère une bibliothèque de type Arduino pour le composant cible. Ce flux de travail permet d'optimiser l'IA en périphérie afin d'obtenir un modèle fiable, avec un temps de latence le plus faible possible, et qui tienne dans les possibilités mémoire de la carte utilisée. EIS permet aussi de garder les différentes versions de développement du modèle afin de revenir à une version antérieure si nécessaire et aussi pour partager le résultat avec d'autres makers.

Si le modèle semble bien fonctionner avec les données d'entraînement, il faut le tester sur des données nouvelles (les 20 % mis de côté). Ceci se fait dans Model Testing avec Classify all. Vous pouvez analyser les échantillons mal classés, réécouter l'audio, examiner avec une figure en 3D montrant comment sont organisés **les clusters des différentes classes (figure 9)**. Le principe est simple, les clusters de couleurs différentes, doivent être bien séparés. Si un point (échantillon) d'une couleur se retrouve dans un cluster d'une autre couleur, c'est qu'il y a un problème pour classer cet échantillon (mot mal prononcé, incomplètement, etc.) et vous pouvez alors corriger le problème. Il est possible de faire tourner la figure sur ses axes ou bien de zoomer dessus pour voir un point particulier. Les performances du modèle sont estimées en fonction de la carte cible que vous allez utiliser. Vous pouvez ainsi accéder au temps nécessaire pour reconnaître un échantillon (inferencing time), l'utilisation de la RAM ou de la ROM (en kilo-octets) et également la précision de la reconnaissance (accuracy).

Il reste maintenant à déployer le modèle sur votre carte à microcontrôleur : Edge Impulse va tout regrouper dans une bibliothèque C ++ à inclure dans votre logiciel embarqué (option à sélectionner). Le compilateur **Edge Optimized Neural (EON)** doit être réglé sur Quantised (int8) car cela optimise tout en restant suffisant. La bibliothèque est générée sous forme de fichier ZIP et peut être incluse dans l'IDE, comme pour n'importe quelle bibliothèque. Il faut savoir que le temps de compilation peut être très long (parfois 15 minutes).

## En conclusion

Voici un rapide tour d'horizon sur ce qu'est l'Intelligence en périphérie et comment vous pouvez la déployer sur vos cartes à microcontrôleur. Si le sujet vous intéresse, je ne peux que vous conseiller d'essayer car avec un peu de pratique, tout ce qui est décrit ici deviendra votre seconde nature et vous verrez que ce n'est pas si difficile. Les tutos sont nombreux sur le site d'Edge Impulse et également sur d'autres sites. Si c'est la reconnaissance d'images qui vous intéresse, voici une petite vidéo de **Dronebot Workshop** (en anglais) qui montre la méthode pour faire reconnaître un petit robot ou une batterie à une IA sur carte à microcontrôleur ESP32 : https://www.youtube.com/watch?v=HDRvZ_BYd08. Entre la reconnaissance vocale et la reconnaissance d'images, vous voilà fin prêt pour démarrer de nouveaux projets à base d'IA embarquée ! ■