Cours: *Embedded Machine Learning*

PRÉSENTER LES CONCEPTS ET LES OUTILS PERMETTANT D'IMPLÉMENTER LA PHASE D'INFÉRENCE DE MODÈLES APPRENTISSAGE AUTOMATIQUE SUR DES CIBLES EMBARQUÉES

« Machine learning (ML) inference on the edge is an increasingly attractive prospect due to its potential for increasing **energy efficiency**, **privacy**, **responsiveness**, and **autonomy** of edge devices. Thus far, the field edge ML has predominantly focused on **mobile** inference, but in recent years, there have been major strides towards expanding the scope of edge ML to ultra-low-power (**tiny**) devices. » [1]

[1] MLPerf Tiny Benchmark – C. Baubury e

Rappels

- Définition de l'Intelligence Artificielle (IA): « Je dirais que l'intelligence artificielle est la capacité, pour une machine, d'accomplir des tâches généralement assurées par les animaux et les humains: percevoir, raisonner et agir » [2]
- Définition d'apprentissage automatique ou *Machine Learning* (ML): « L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés » [3]
- Phase de l'apprentissage automatique:
  - Apprentissage: Extraction des connaissances à partir de données
  - Inférence: Exploitation des connaissances apprises sur de nouvelles données

[2] Quand la machine apprend – Y. Le Cun (2019)
[3] Machine Learning avec Scikit-Learn – A. Géron

Motivations

- L’intelligence artificielle (IA), plus particulièrement le *Machine Learning* (ML) et le *Deep Learning* (DL), sont de plus en plus présents dans notre vie quotidienne (Chat GPT, Alexa, DALL-E, voitures autonomes, …).
- Le *Machine Learning* et *Deep Learning* demande une quantité importante de données ainsi que des moyens de calculs importants comme des serveurs de calculs (*Cloud*).
- *Edge Intelligence* [4]: Le but du *Edge Intelligence* est d’implémenter des algorithmes d’apprentissage automatique à proximité des sources de données.
- *TinyML* [1]: Le but du *TinyML* est d’implémenter (les phases d’inférences) des algorithmes de *Machine Learning* et de *Deep Learning* sur des cibles embarquées ultra basses consommation (*<1mW*).

[4] Vers des algorithmes d'apprentissage automatique économes en E/S pour les systèmes embarqués : application aux K-means et Random Forests - C.

Cloud, *Edge* et dispositif (1)

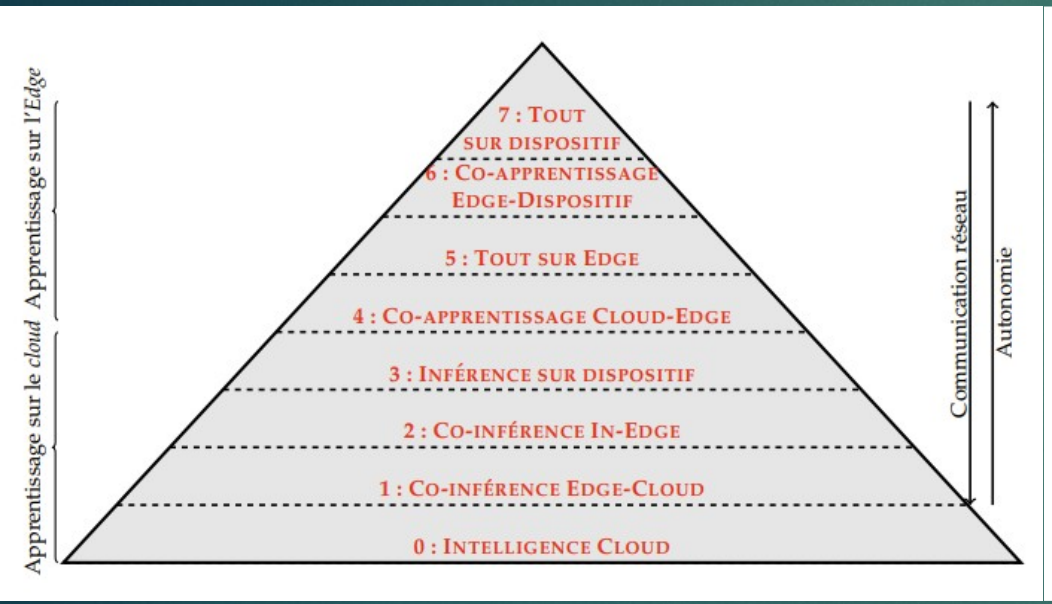
- Les différents types de paradigmes:
  - *Cloud*: Paradigme utilisant des serveurs informatiques à distance et hébergées sur internet pour stocker les données, entraîner les modèles ainsi que réaliser l'inférence.
  - *Edge* (Réseaux/*Mobile*): Paradigme utilisant des dispositifs proches de la source de données pour stocker les données, entraîner les modèles ainsi que réaliser l'inférence.
  - Dispositif (*Mobile*/*Tiny*): Paradigme utilisant les dispositifs collectant les données pour stocker les données, entraîner les modèles ainsi que réaliser l'inférence.

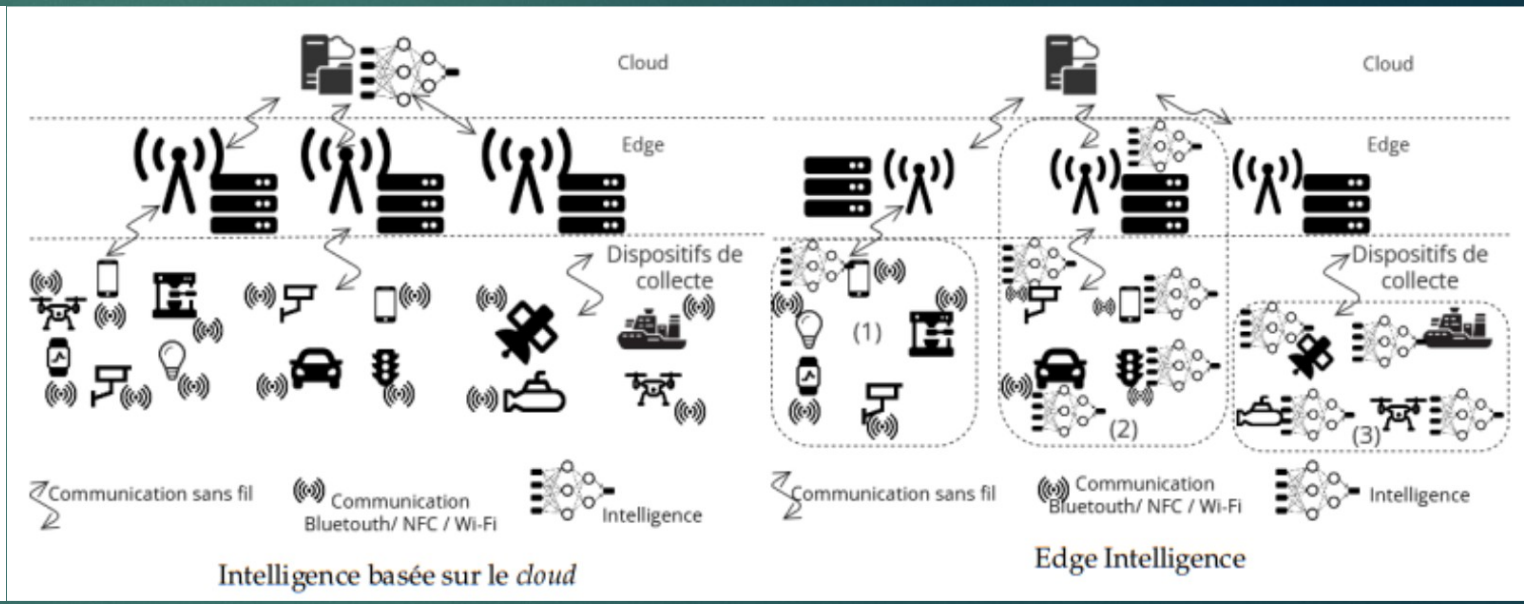
Platform	Freq.	Memory	Storage	Power	Price	Co_2-eq Footprint
Cloud	GHz	10+GB	TBs-PBs	~1 kW	$1000+	Hundreds of kgs
Mobile	GHz	Few GB	GBs	~1 W	$100+	Tens of kgs
Tiny	**MHz**	**KBs**	**Few MB**	**~1 mW**	**$10**	**Single kgs**

Comparaison Cloud/Mobile/Tiny [5]

[5] Is TinyML Sustainable ? – S. Prakash

Cloud, Edge et dispositif (2)



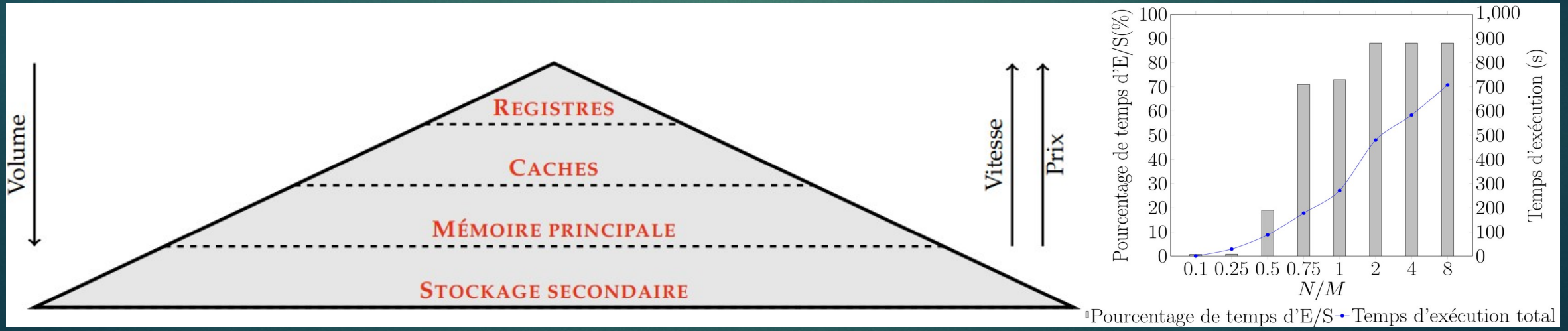


Niveaux et cas d'utilisation de *l'Edge Intelligence* [4]

- **Dans ce cours, nous nous focaliserons sur l'implémentation de la phase d'inférence sur un système embarqué (niveau 2/3) !**

Le problème de l’entraînement

- La tâche d’entrainement peut difficilement se faire sur les plateformes embarquées à cause, notamment, des restrictions en termes de mémoire :
  - *Les algorithmes de ML supposent que les jeux de données sont entièrement contenus en mémoire principale ;*
  - *L’espace mémoire disponible sur ces plateformes ne suffit pas toujours à contenir le jeu de données.*



Types de mémoire et impact des entrées/sorties (E/S) [4]

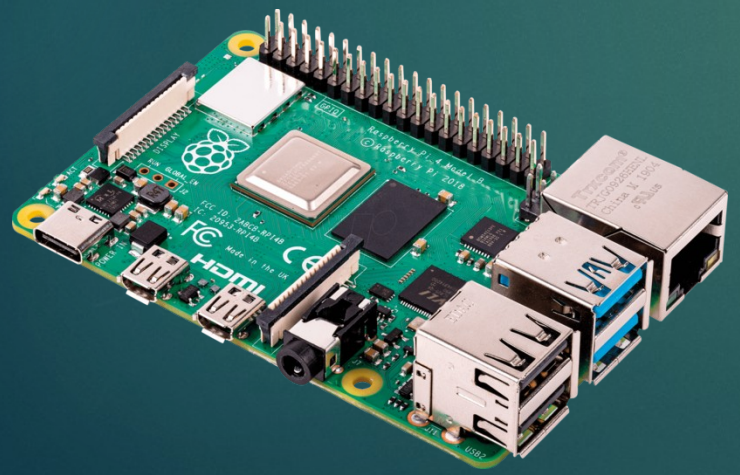
Avantages et inconvénients

- Avantages [1, 4, 5, 6]:
  - Réactivité: Réduction des latences dues aux communications
  - Consommation: Réduction de la consommation due à l'envoie des données
  - Confidentialité: Meilleur contrôle sur les données produites
- Désavantages:
  - Mise à l'échelle: Limitation liée à la taille des modèles
  - Occupation mémoire: Augmentation de la mémoire utilisée liée au stockage du modèle

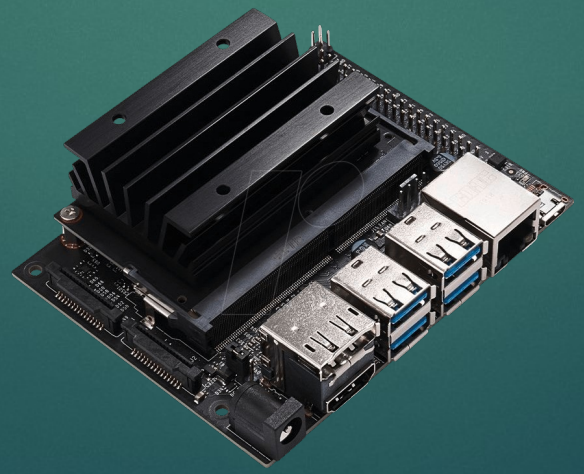
[6] Machine Learning Sensors – P. Warden et

Les types de plateformes embarquées (1)

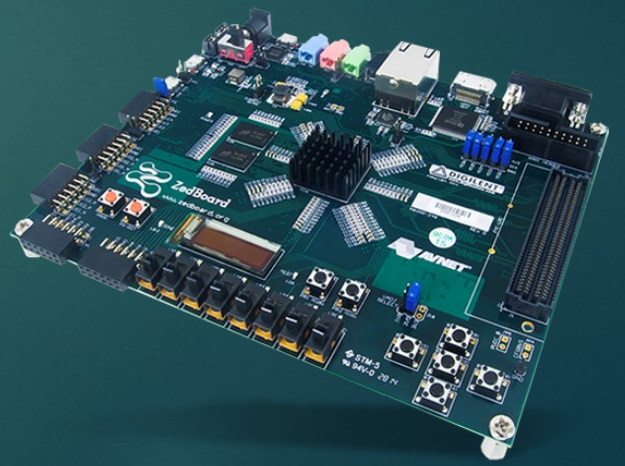
- Les plateformes d’exécution:
  - Processeur (CPU): STM32, Raspberry Pi 4 B, …
  - Carte graphique (GPU): Jetson Nano, Jetson Xavier, …
  - *Field-Programmable Gate Array* (FPGA): Zedboard, ZCU102, …
  - Autres: *Deep Learning Accelerator* (DLA), *In-Memory Computing* (IMC), …



Raspberry Pi 4B

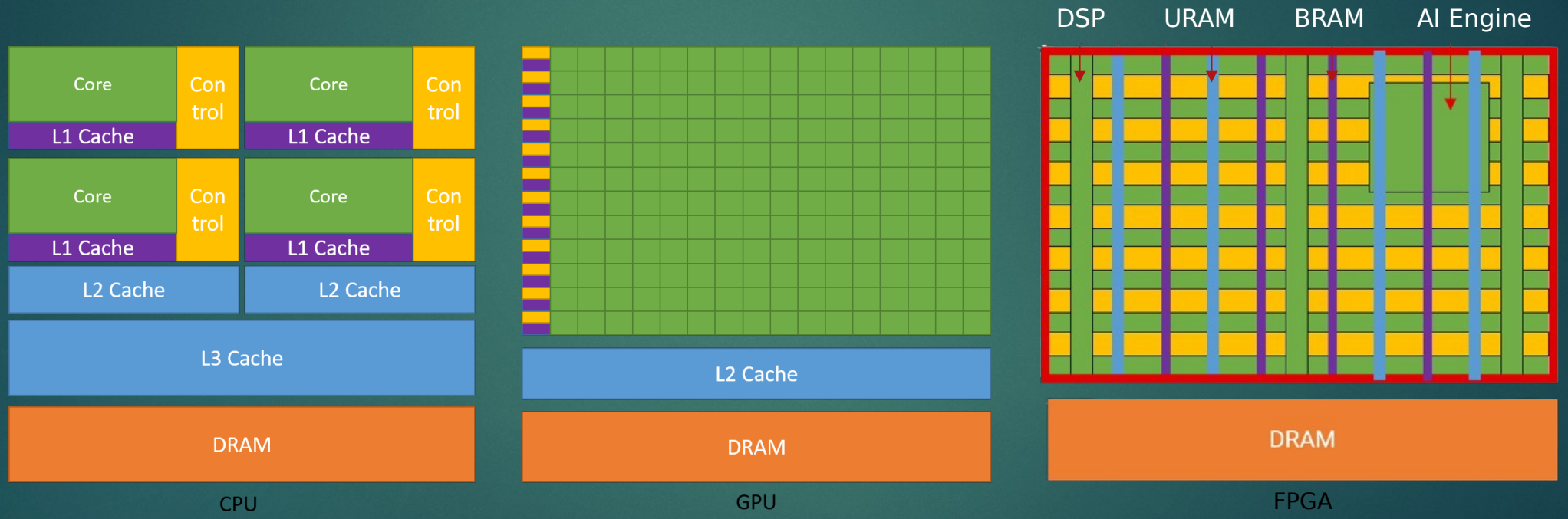


Jetson Nano



Zedboard

Les types de plateformes embarquées (2)



Comparatif des architectures matérielles [7-8]

[7], What Every Developer Should Know About GPU Computing - A. Upadhyay https://codeconfessions.substack.com/p/gpu-computing
[8] Migrating from CUDA to Vitis - F. Rivoallon https://www.xilinx.com/developer/articles/migrating-from-cuda-to-vitis.html

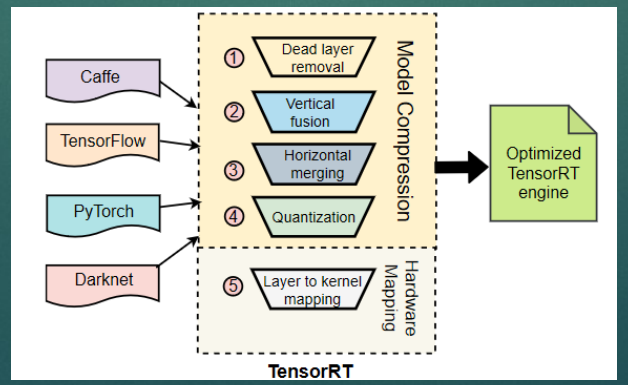
Inférence sur CPU

- Les différents types d'inférence sur CPU [9]:
  - *Hand coding*: Le « codage manuel » permet une implémentation bas-niveau et des optimisations spécifiques (théorique et logicielle). Cependant, ce type d'implémentation est couteux en temps, peu flexible et difficilement comparable.
  - *Code generation*: La génération de code produit un code bien optimisé sans effort (flexibilité) grâce à une abstraction et une automatisation. Cependant, ce type d'implémentation est difficilement comparable et moins optimisé.
    - Exemple: emlearn
  - *ML interpreters*: L'utilisation d'un moteur d'inférence permet une meilleure flexibilité ainsi qu'une meilleur portabilité/compatibilité. Cependant, le code est moins optimisé que les deux options d'implémentations précédentes.
    - Exemple: Tensorflow Lite

[9] Benchmarking TinyML Systems: Challenges and Direction – C. R. Bandury et al.

Inférence sur GPU

- Inférence sur GPU :
  - Codage manuelle: Utilisation de language de programmation orientée GPU.
    - Exemple : CUDA
  - Conversion: Traduction des opérations algorithmiques en opérations matricielles.
    - Exemple : HummingBird.ml
  - Moteurs d'inférence : mapping des modèles entrainés sur les cœurs de calcul.
    - Exemple : TensorRT

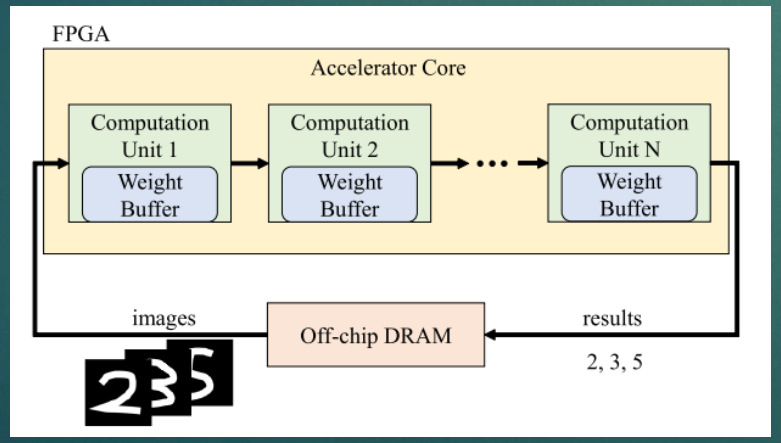


Flot d'optimisation selon TensorRT

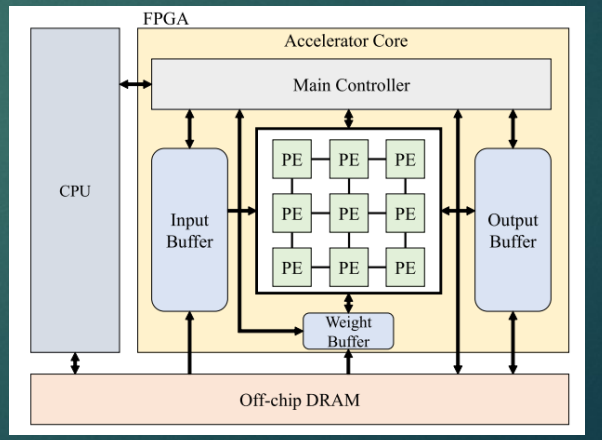
[10] Demystifying TensorRT: Characterizing Neural Network Inference Engine on Nvidia Edge Devices – O.

Inférence sur FPGA

- Les différents types d'inférence sur FPGA [11]:
  - *Dataflow*: Chaque couche est implémentée sur une unité de calculs différente.
    - Exemple: FINN, hls4ml, …
  - *Overlay*: Les différentes opérations d'une tâche d'inférence sont exécutées par un ensemble d'éléments de calculs.
    - Exemple: Vitis AI, Deep Learning HLS Toolbox, …



Architecture *dataflow* [11]



Architecture *overlay* [11]

[11] An Exploration of State-of-the-Art Automation Frameworks for FPGA-Based DNN Acceleration – F. Hamanaka et al.

Compression de modèles

- Le but de la compression de modèles en DL, et plus généralement en ML, a pour but de réduire la taille des modèles ainsi que leurs latences.
- Techniques de compression en DL [12, 13] :
  - La quantification: Le but de la quantification est de représenter les coefficients d'un modèle avec une moins bonne précision, en utilisant notamment des virgules fixes.
  - Le *pruning*: Le but du *pruning* est de supprimer des connections ou des neurones, voir des *kernels* ou des canaux (pour les CNN).
  - *Low-rank approximation*: Le but du *low-rank approximation* exploite la faible dimensionnalité des différentes couches.
  - *Knowledge distillation*: Le but du *knowledge distillation* est de transférer des connaissances apprises par un réseau large (*teacher*) vers un réseau plus petit (*student*).
  - *Neural Architecture Search* (NAS): Le but du NAS est de chercher l'architecture la plus adaptés à la résolution d'un problème.
- Couches spécifiques: *Depthwise Separable Convolution* [14], …

[12] A Survey of Model Compression and Acceleration for Deep Neural Networks - Cheng et al.
[13] An Overview of Model Compression Techniques for Deep Learning in Space - H. Perterson et al.

Comparaison des plateformes (1)

Metric/ Feature	Analysis	Hardware
Neural network training	GPUs can be highly parallelised and are supported by many frameworks for training [26, 83, 84]	GPU
Neural network inference (batch size small)	FPGAs can be highly customised per layer to obtain the lowest latency	FPGA
Neural network inference (batch size large)	GPUs have large parallelisation capabilities	GPU
Interfaces	As FPGAs are close to hardware it can connect to many interfaces	FPGA
Resources/Size	FPGAs can be highly customised with minimal overhead	FPGA
Customization	FPGAs enable the possibility of a custom-tailored accelerator architecture in hardware	FPGA
Development/ Ease of use	CPUs are easier to program than GPUs with frameworks and both are easier than FPGAs	CPU

Comparaison entre CPU, GPU et FPGA [15]

[15] Ultra low latency deep neural network inference for gravitational waves interferometer – M. C. B de Rooij

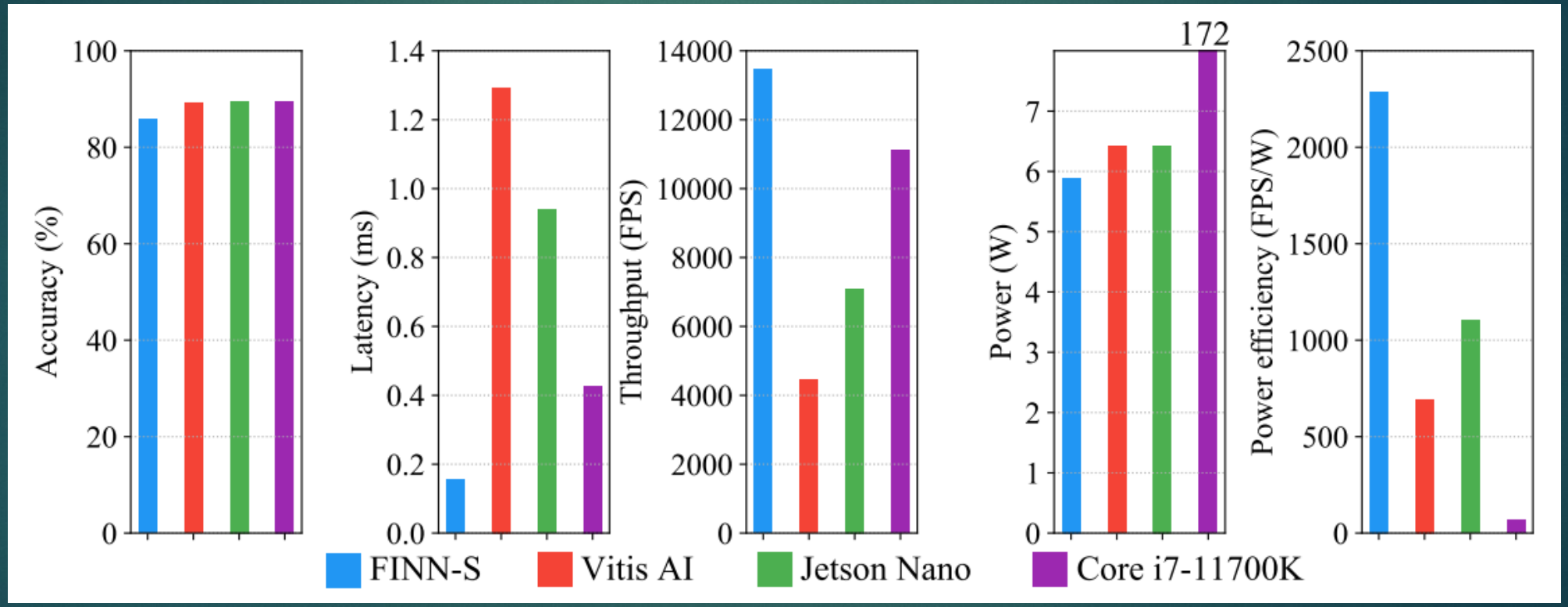
Comparaison des plateformes (2)

Core Type	Custom ASIC	Typical Power Consumption	Description	Strengths	Constraints
CPU		High	Flexible, general purpose processing units	• Complex instructions and tasks • System management	• Possible memory access bottlenecks • Few cores (4-16)
GPU		High	Parallel cores for high quality graphics rendering	• High performance AI processing • Highly parallel core with 100's or 1,000's of cores	• High power consumption • Large footprint
FPGA		Medium	Configurable logic gates	• Flexible • In-field reprogrammability	• High power consumption • Programming complexity
ASIC		Low	Custom logic designed with libraries	• Fast and low power consumption • Small footprint	• Fixed function • Expensive custom design
	VPU	Ultra-low	Image and vision processor/co-processor	• Low power and small footprint • Dedicated to image and vision acceleration	• Limited dataset and batch size • Limited network support
	TPU	Low to medium	Custom ASIC developed by Google	• Specialized tool support • Optimized for TensorFlow	• Proprietary design • Very limited framework support

Comparaison entre CPU, GPU, FPGA et ASIC [16]

[16] Embedded Hardware for Processing AI at the Edge: GPU, VPU, FPGA, and ASIC Explained https://blog.adlinktech.com/2021/02/19/embedded-hardware-processing-ai-edge-gpu-vpu-fpga-asic/

Comparaison des plateformes (3)



Comparaison sur le MLPerf Tiny Benchmark [8]

Merci pour votre écoute

Bibliographie (1)

- [1] MLPerf Tiny Benchmark – C. Baubury e al.
- [2] Quand la machine apprend – Y. Le Cun (2019)
- [3] Machine Learning avec Scikit-Learn – A. Géron (2019)
- [4] Vers des algorithmes d'apprentissage automatique économes en E/S pour les systèmes embarqués : application aux K-means et Random Forests - C. Slimani
- [5] Is TinyML Sustainable ? – S. Prakash et al.
- [6] Machine Learning Sensors – P. Warden et al.
- [7] A. Upadhyay, What Every Developer Should Know About GPU Computing (2023) https://codeconfessions.substack.com/p/gpu-computing
- [8] F. Rivoallon, Migrating from CUDA to Vitis (2022) https://www.xilinx.com/developer/articles/migrating-from-cuda-to-vitis.html
- [9] C. R. Bandury et al. – Benchmarking TinyML Systems: Challenges and Direction
- [10] Demystifying TensorRT: Characterizing Neural Network Inference Engine on Nvidia Edge Devices – O. Shafi et al.

Bibliographie (2)

- [11] An Exploration of State-of-the-Art Automation Frameworks for FPGA-Based DNN Acceleration – F. Hamanaka et al.
- [11] Cheng et al. - A Survey of Model Compression and Acceleration for Deep Neural Networks
- [12] A Survey of Model Compression and Acceleration for Deep Neural Networks - Cheng et al.
- [13] An Overview of Model Compression Techniques for Deep Learning in Space - H. Perterson et al.
- [14] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications – Howard et al.
- [15] Ultra low latency deep neural network inference for gravitational waves interferometer – M. C. B de Rooij et al.
- [16] Embedded Hardware for Processing AI at the Edge: GPU, VPU, FPGA, and ASIC Explained https://blog.adlinktech.com/2021/02/19/embedded-hardware-processing-ai-edge-gpu-vpu-fpga-asic/