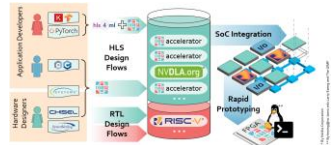
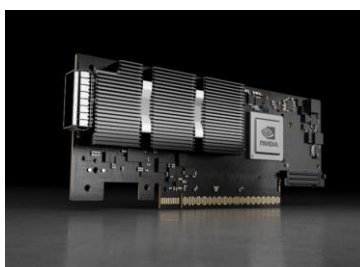
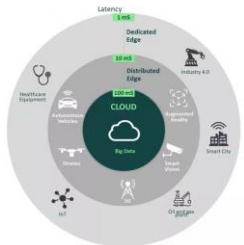


L'apprentissage profond dans nos applications

– La recherche en IA au SEA –



- ➔ Introduction rapide à l'apprentissage profond
- ➔ Application à l'instrument optimisé
- ➔ Systèmes embarqués – FPGA –
- ➔ La R&T THINK



Accelerator generation with hls4ml 	Automatic integration in ESP 	Full-system RTL simulation 	Full-system test on FPGA
--	----------------------------------	--------------------------------	------------------------------

Définition

« L'intelligence artificielle (IA) est un processus d'imitation de l'intelligence humaine qui repose sur la création et l'application d'algorithmes exécutés dans un environnement informatique dynamique.

Son but est de permettre à des ordinateurs de penser et d'agir comme des êtres humains. »

Pour y parvenir, trois composants sont nécessaires :

- **Des systèmes informatiques (Matériels)**
- **Des données avec des systèmes de gestion (Simulation/instrumentation)**
- **Des algorithmes d'IA avancés (code)**

Pour se rapprocher le plus possible du comportement humain, l'intelligence artificielle a besoin d'une **quantité de données** et d'une **capacité de traitement élevées**. »



Système embarqué

La bataille de l'informatique

1600

1950

SCIENCES EXPERIMENTALES

SCIENCES THEORIQUES

SCIENCES CALCULATOIRES

Informatique : traitement automatisé de l'information

Représentation de l'information

Algorithmes

Symbolisme

connectivisme

Probabilisme

Analogisme

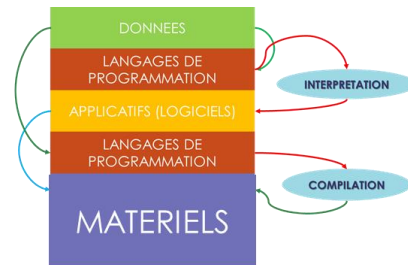
Matériels (Microprocesseurs)
Architecture de Von Neumann



STRUCTURES DE DONNEES

Base de données

- ☺ Table de hachage
- ☺ Listes
- ☺ Listes chaînées
- ☺ Vecteurs
- ☺ Pile / Tas
- ☺ Tableau N dimension (Tenseur)



Ordinateur quantique
Langage spécifique

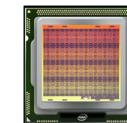
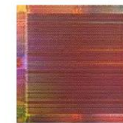
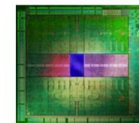


Ordinateur actuel
Central Processing Unit

Calcul haute performance



Langage C, C++ et outils spécifiques de compilation



CPUs

DSPs

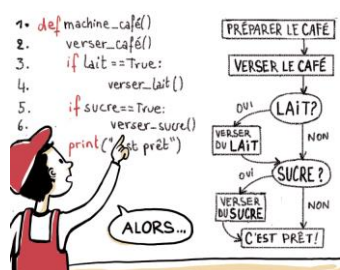
GPUs

FPGAs

NMC

- ➔ Diviser et régner
- ➔ Glouton
- ➔ Force brute
- ➔ Prog. Dynamique

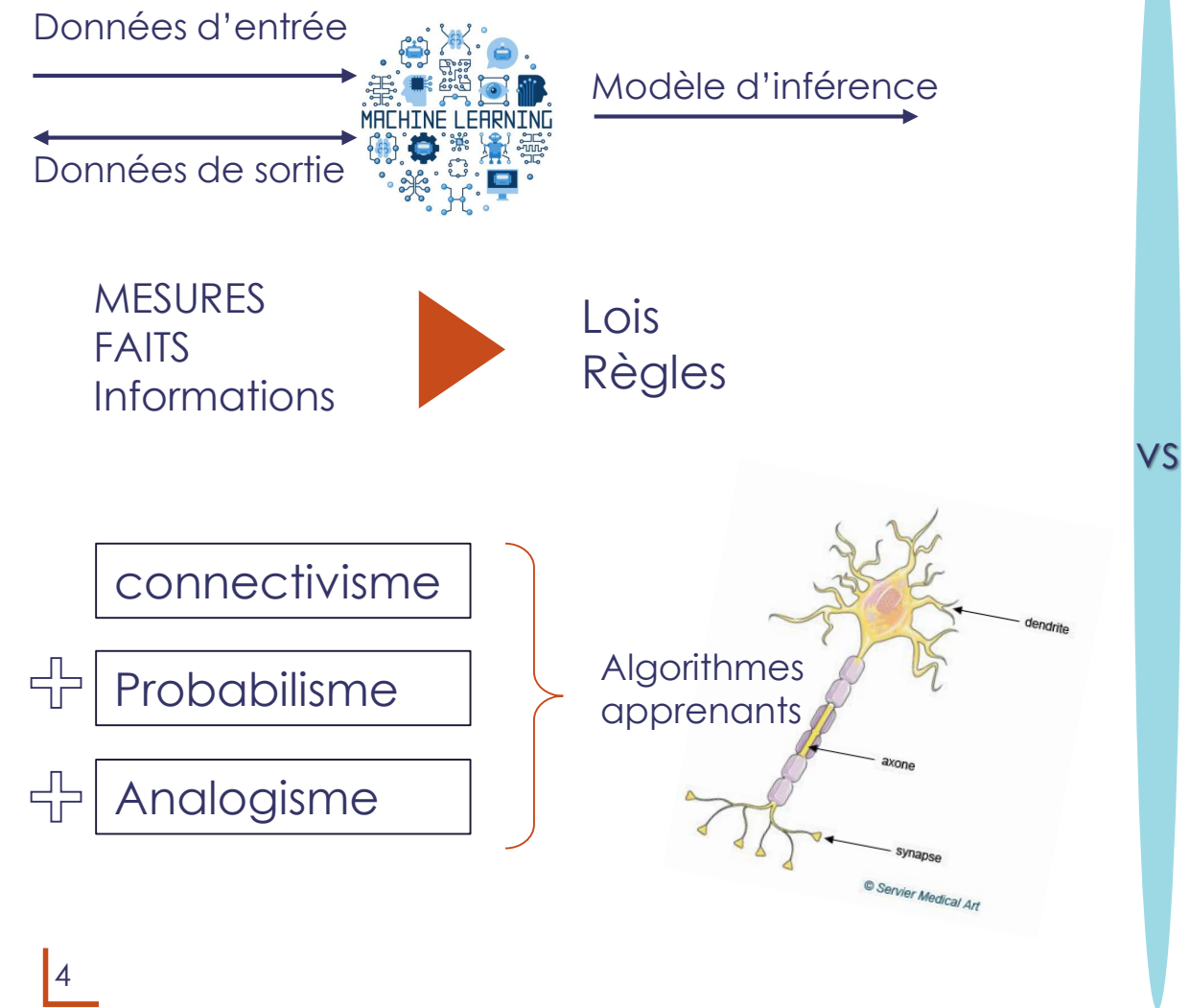
...



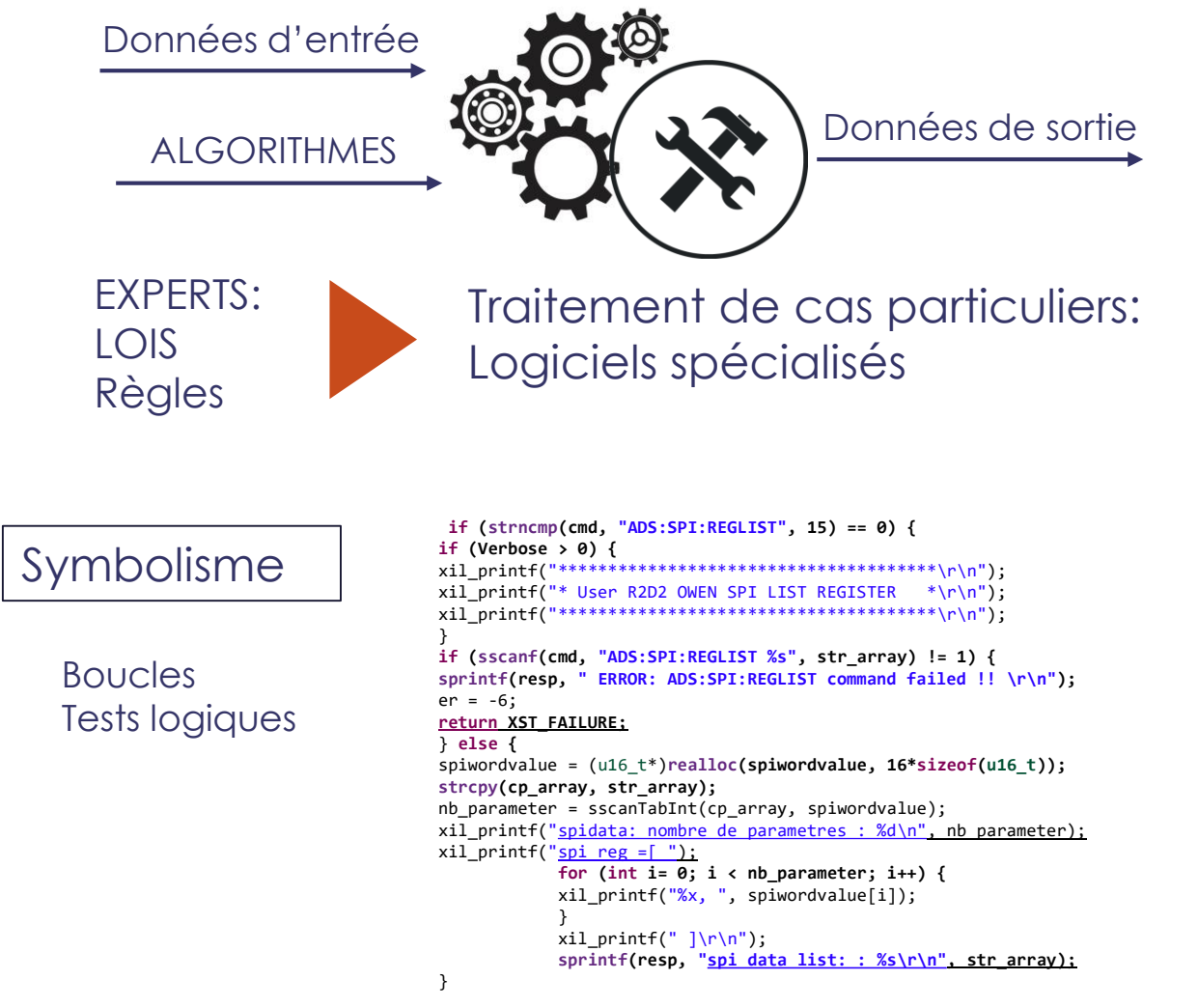
La bataille de l'informatique



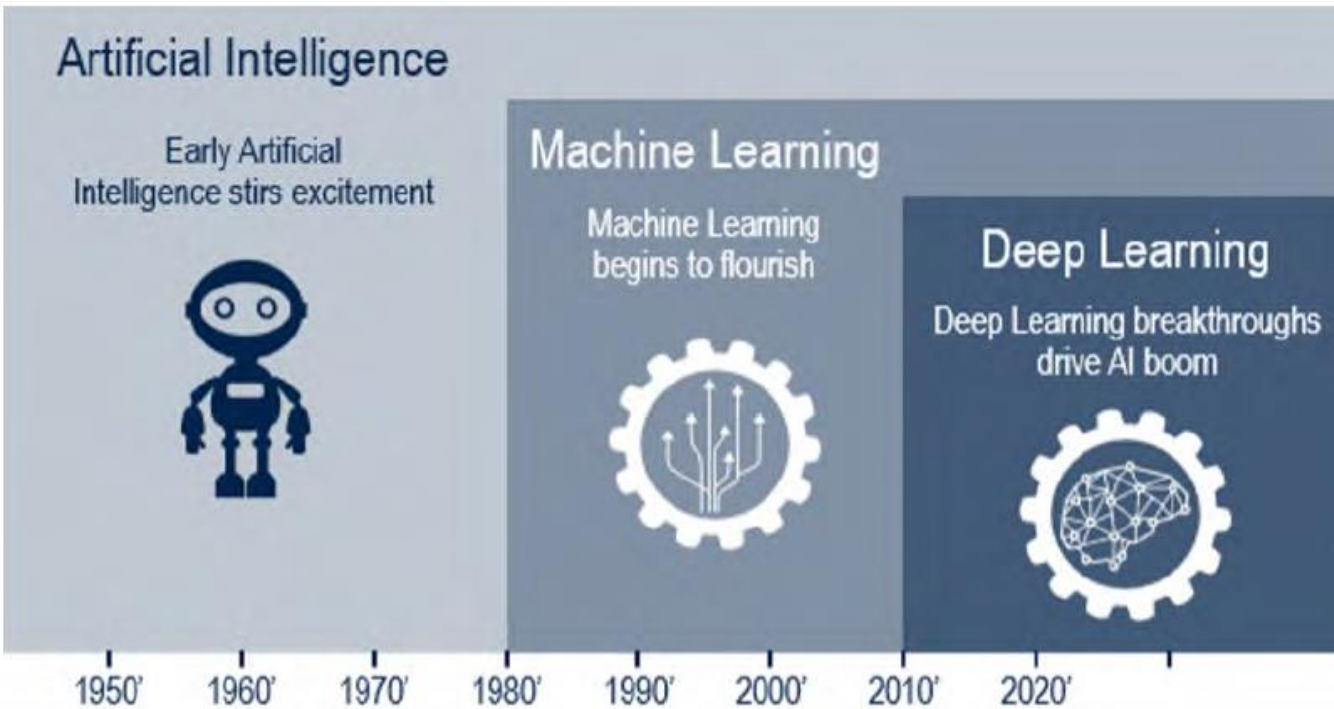
Approche inférentielle (Inductive)



Approche déductive



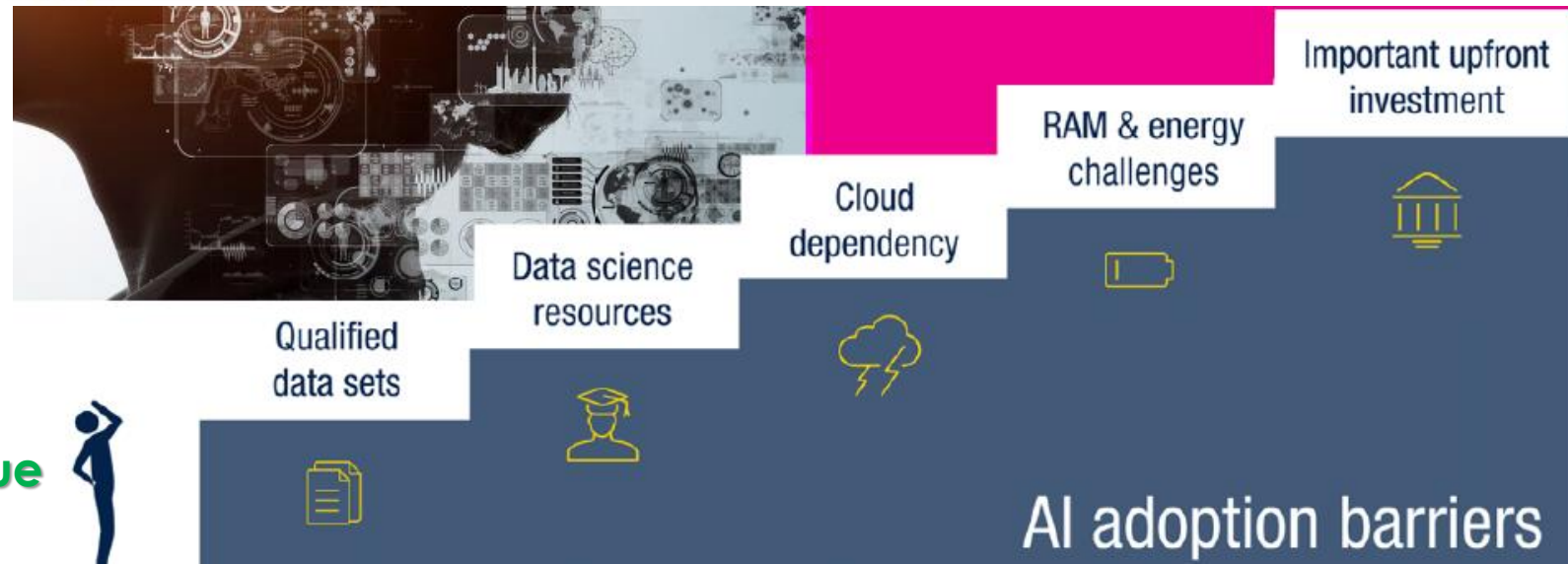
Historique



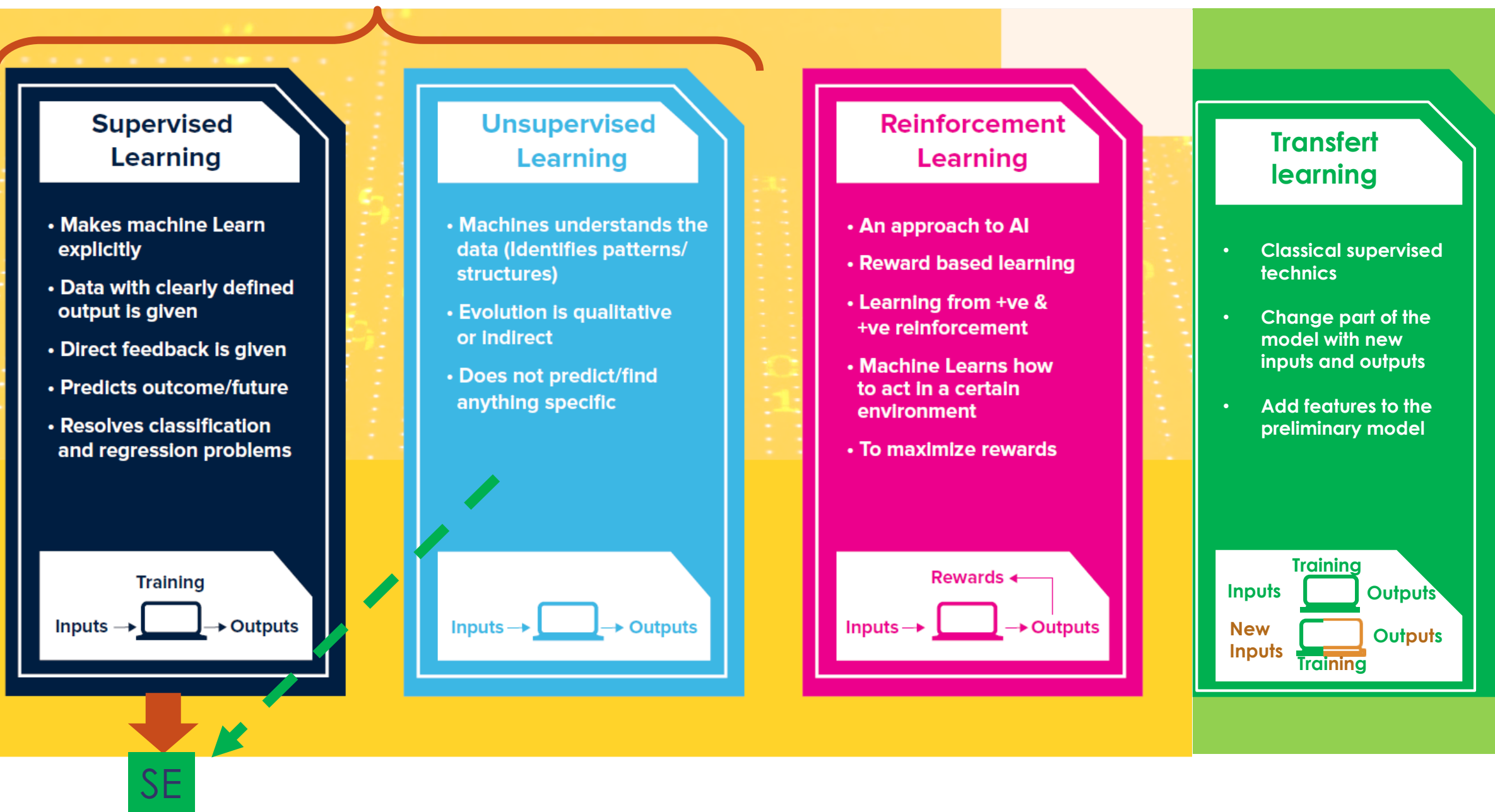
→ Evolution et progression de l'IA

→ Les barrières pour adopter l'IA dans les produits pour les sociétés.

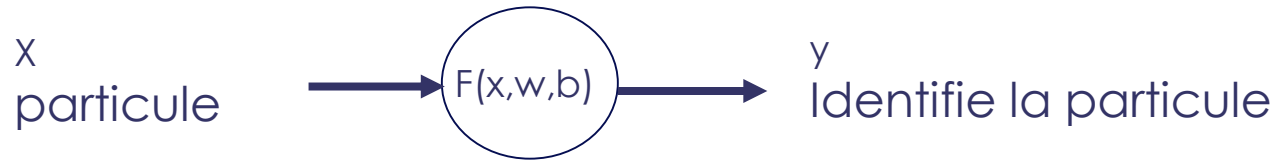
- investissement en RH, matériels, données
- Grandes compagnies



Les quatre catégories d'apprentissage profond



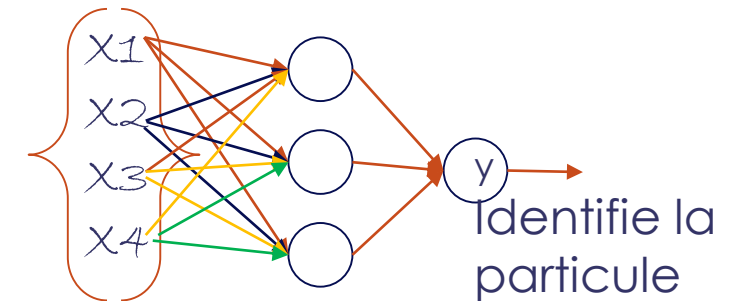
Apprentissage supervisé : Réseau de Neurones



Big Data : grand nombre de données
Gros modèles de taille importante de neurones
un grand nombre d'exemples d'entraînement

Puissance de calcul : CPU/GPU/FPGA....

Innovation dans les algorithmes (la descente du gradient)

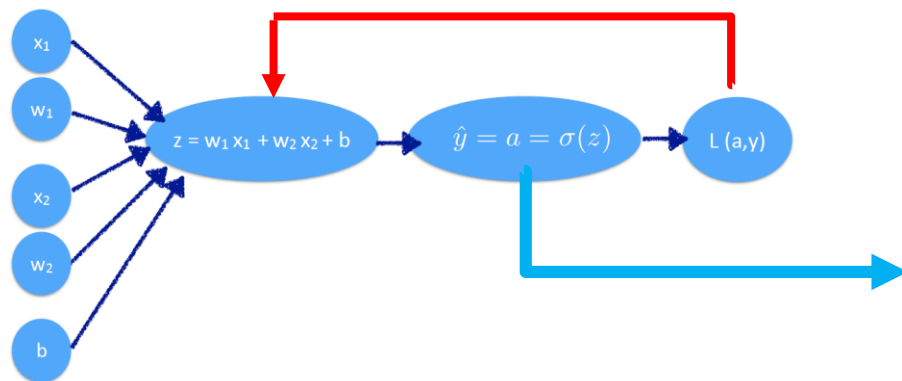


$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

Loss function:

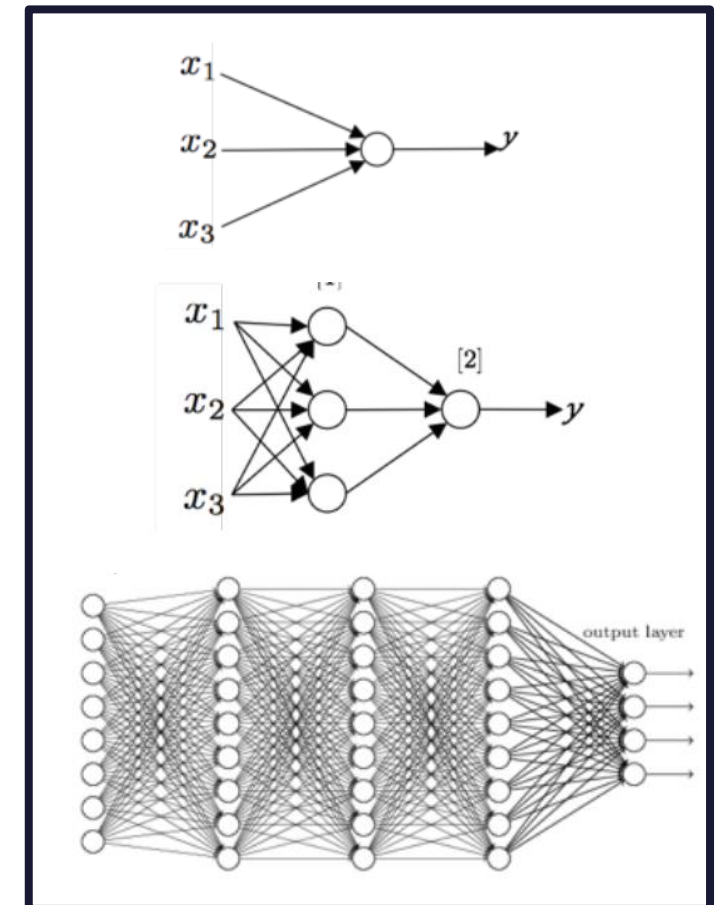
$$L(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



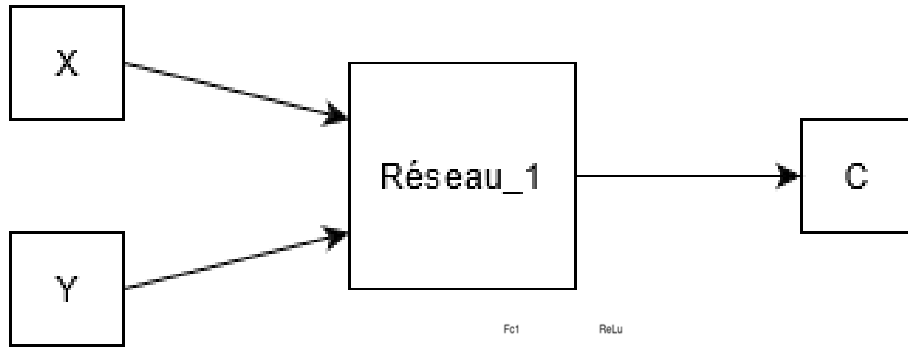
ACTIVATION FUNCTIONS

Name	Plot	Equation
Sigmoid		$g(z) = \frac{1}{1 + e^{-z}}$
Hyperbolic tangent		$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
Rectified linear unit (ReLU)		$g(z) = \max(0, z)$
Leaky ReLU		$g(z) = \max(\alpha z, z)$

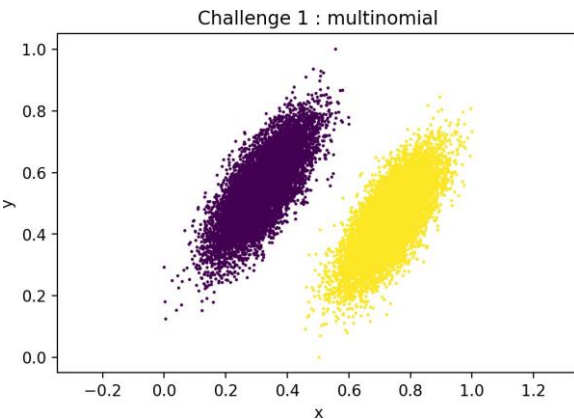
Using a linear activation function is useless un more than one layer networks.
 It's rarely used



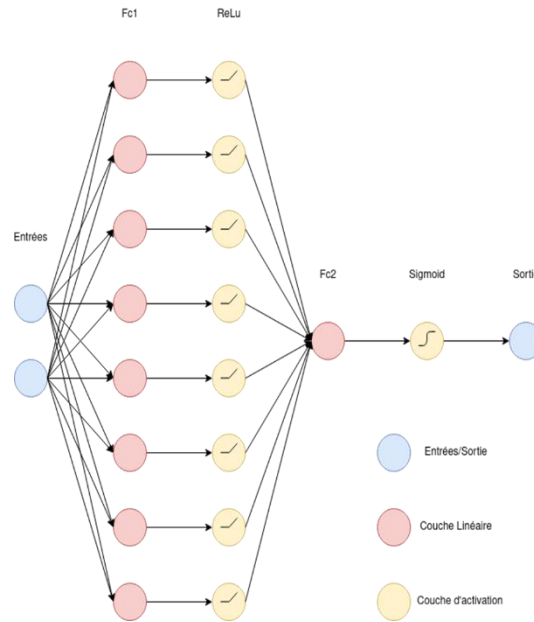
Un exemple de calcul: réseau de neurone peu profond



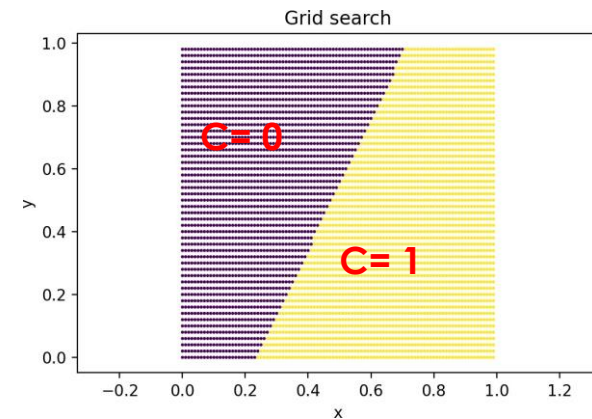
Principe: on fournit en entrée des coordonnées (x,y). Le réseau nous classe les coordonnées selon la région d'appartenance. Ici, c'est une simple régression logistique avec une séparation de deux région linéaire.



2x8x1



42 multiplications

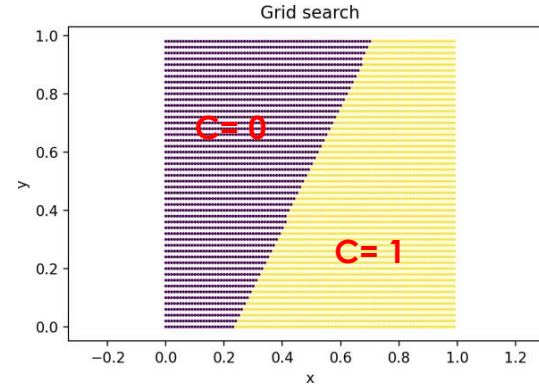
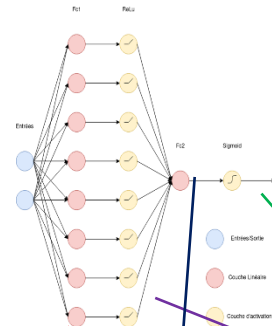
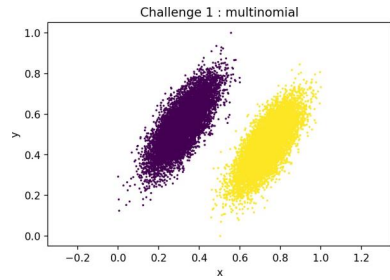


Il faut calculer



$$W = \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \text{ et } B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$
$$s = \sigma(\mathbf{W}\mathbf{x} + \mathbf{B})$$

Un exemple de calcul: réseau de neurone peu profond



X=0,2
Y=0,7

$$\begin{pmatrix} 1.7856 & -1.2345 \\ 2.4834 & -0.5797 \\ -0.3130 & 0.2658 \\ 0.3110 & -0.4116 \\ -0.4458 & 0.8010 \\ 3.4282 & -1.7166 \\ -0.6808 & 0.1631 \\ -0.5633 & 0.5694 \end{pmatrix} \times \begin{pmatrix} 0.2 \\ 0.7 \end{pmatrix} +$$

$$\begin{pmatrix} 0.0795 \\ -0.5202 \\ 0.4716 \\ -0.4349 \\ 1.3222 \\ -0.2305 \\ -0.3073 \\ 0.9498 \end{pmatrix} = \begin{pmatrix} -0.4275 \\ -0.4294 \\ 0.5951 \\ -0.6608 \\ 1.7938 \\ -0.7465 \\ -0.3293 \\ 1.2358 \end{pmatrix}$$

$$\begin{pmatrix} -0.4275 \\ -0.4294 \\ 0.5951 \\ -0.6608 \\ 1.7938 \\ -0.7465 \\ -0.3293 \\ 1.2358 \end{pmatrix} \xrightarrow{ReLU} \begin{pmatrix} 0 \\ 0 \\ 0.5951 \\ 0 \\ 1.7938 \\ 0 \\ 0 \\ 1.2358 \end{pmatrix}$$

$$(2.0441 \ 2.5193 \ -0.6255 \ 0.0535 \ -1.5080 \ 3.7406 \ 0.1052 \ -1.2330) \times \begin{pmatrix} 0 \\ 0 \\ 0.5951 \\ 0 \\ 1.7938 \\ 0 \\ 0 \\ 1.2358 \end{pmatrix} + (-1.4529) = (-6.0539)$$

$$(-6.0539) \xrightarrow{\text{sigmoïde}} (0.0023) \sim 0$$

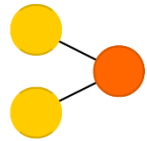
X=0,2
Y=0,7 Appartient à la zone **C=0**

A mostly complete chart of Neural Networks

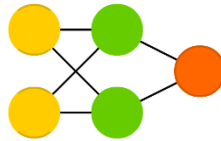
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

- Input Cell
- Backfed Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool

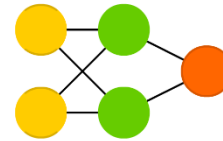
Perceptron (P)



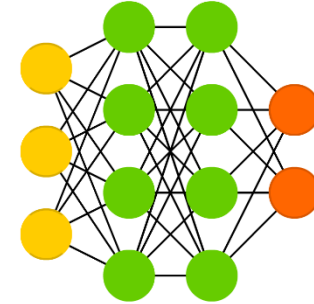
Feed Forward (FF)



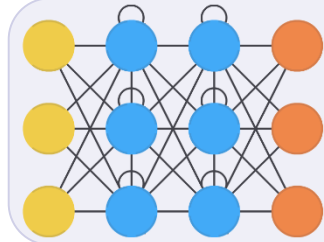
Radial Basis Network (RBF)



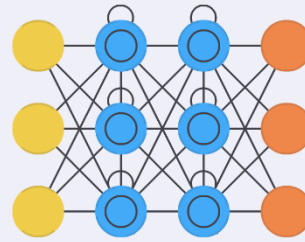
Deep Feed Forward (DFF)



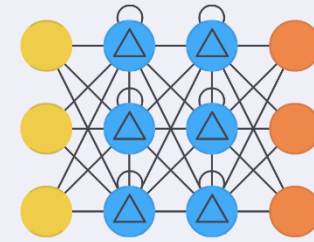
Recurrent Neural Network (RNN)



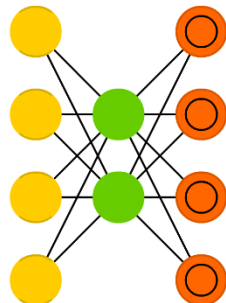
Long / Short Term Memory (LSTM)



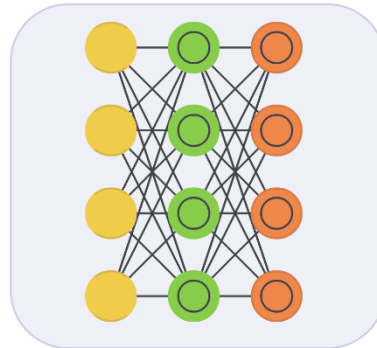
Gated Recurrent Unit (GRU)



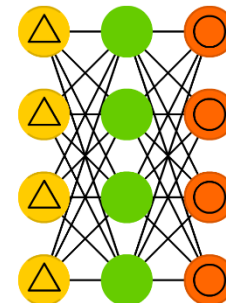
Auto Encoder (AE)



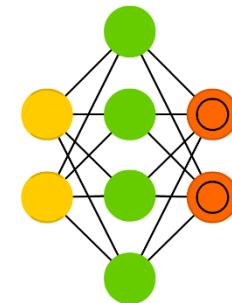
Variational AE (VAE)



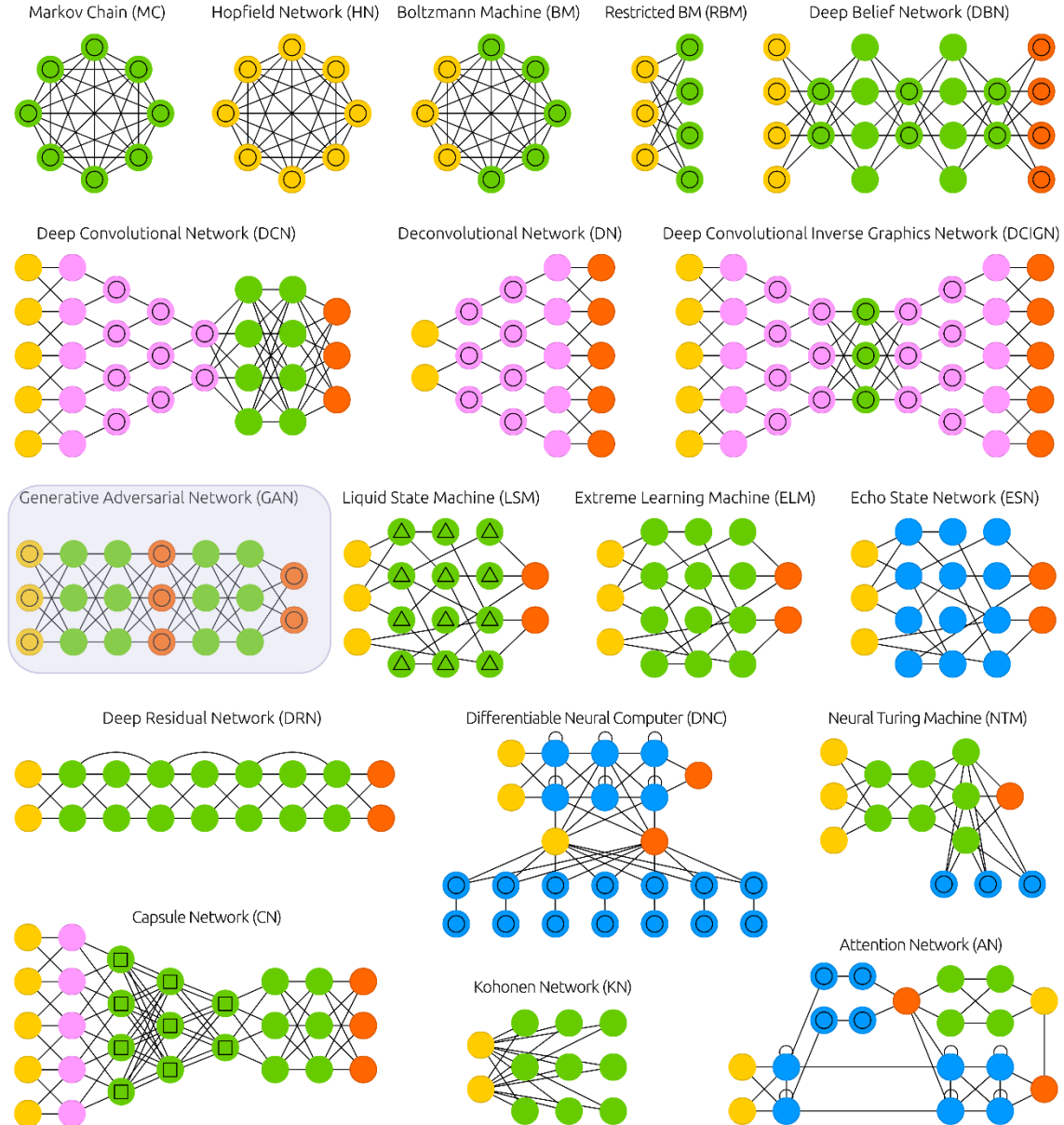
Denosing AE (DAE)



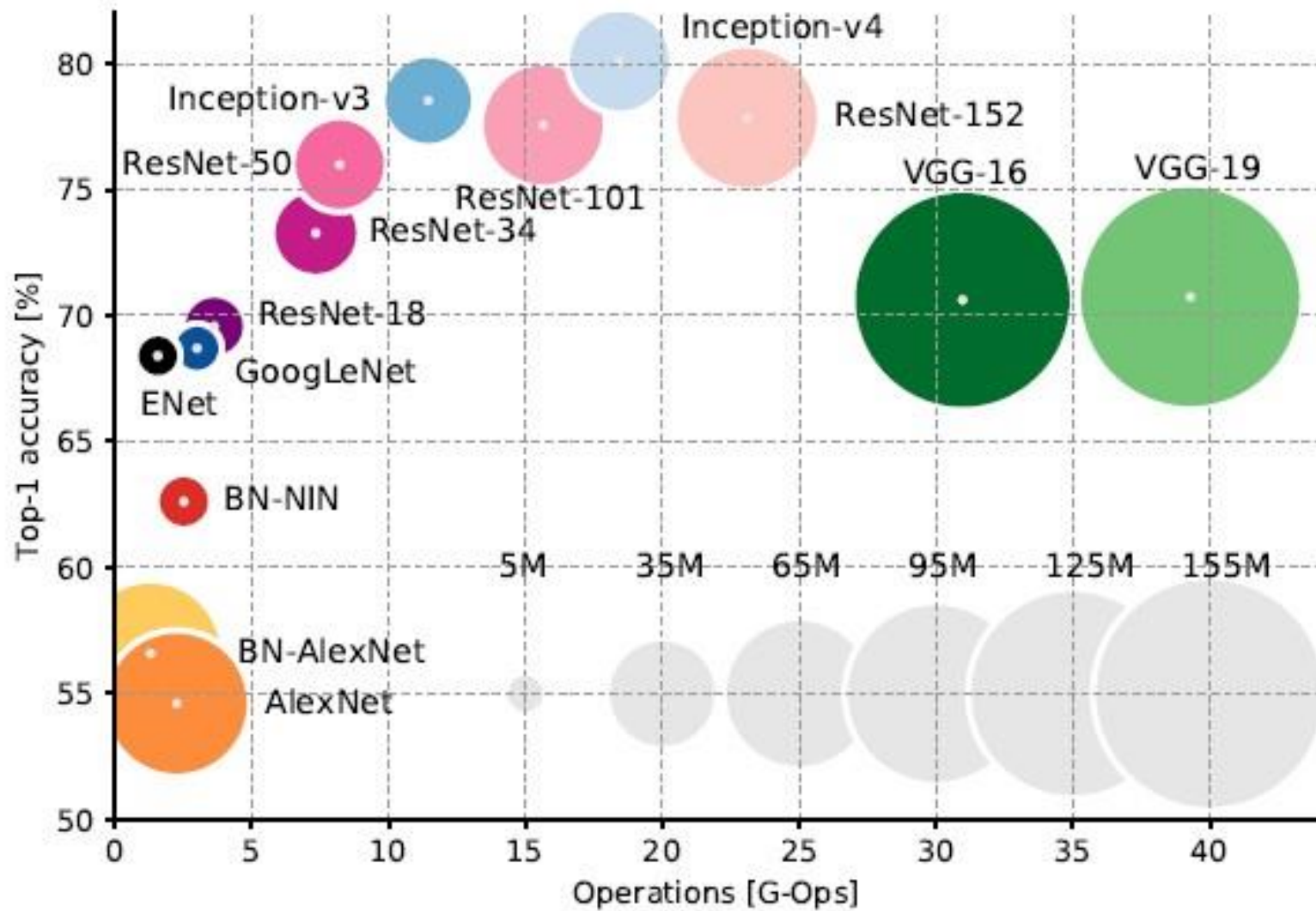
Sparse AE (SAE)



Zoo des NNs

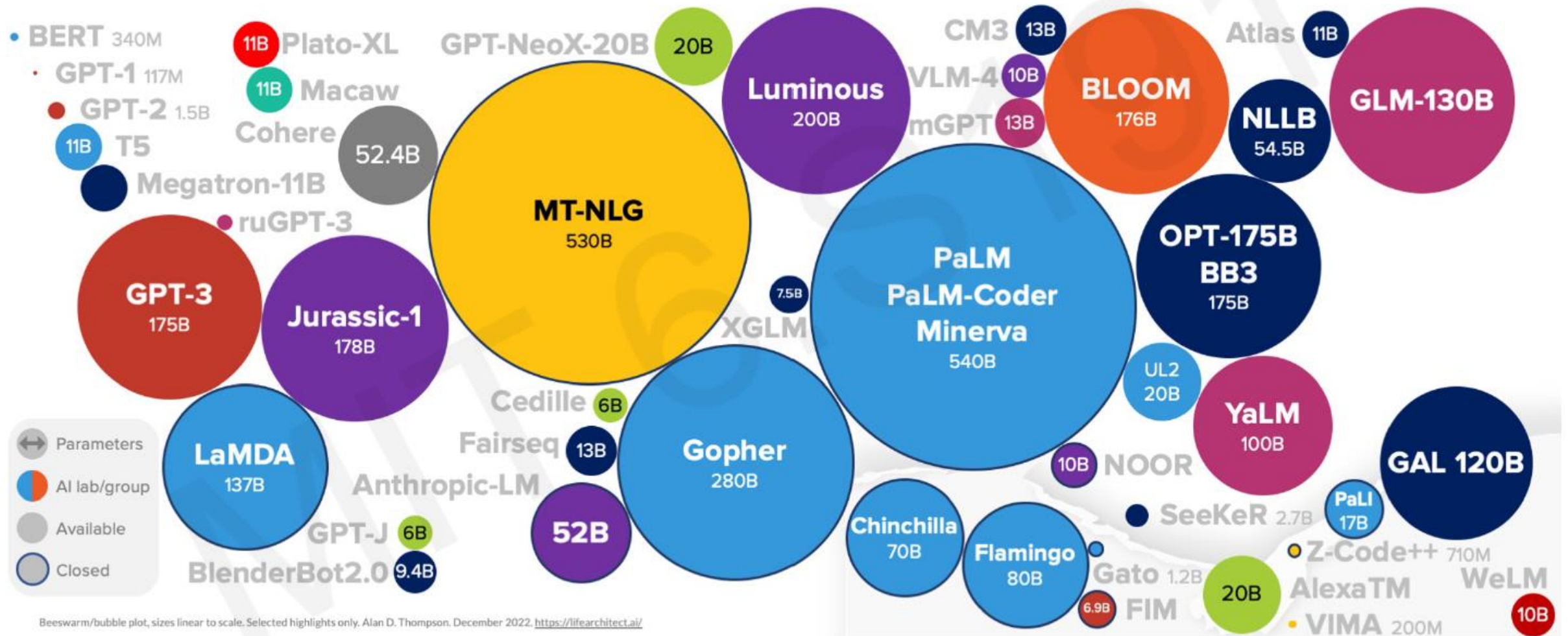


Les autres réseaux : Performances (avant 2020)



Modern Era of Statistics

Language Models size – up to Dec, 2022



Boosted decision Tree :

Discrimination signals /bruit de fond
applications: CMS

Autoencodeurs et classifieurs:

Détection d'anomalies dans l'analyse en physique
Application: Atlas

Reduction de dimensions:

simples paramètres de représentation pour l'analyse statistique classique
Discrimination signals/bruit de fond
applications: CMS

CNN, ANN:

Monitoring de la qualité des données (taux de trigger, latence...)
Applications: CMS

GAN, Variational autoencoder:

Simulation et modélisation d'interaction
Modèles génératifs

Implémentation de NN embarqués:

Triggers et réduction des données lors de l'acquisition
LHC, DUNE

Image classification

Reconstruction des données des détecteurs
Analyse des jets

Apprentissage non supervisé (k-nn, SVM, U-Net):

Regroupement des données (clustering) pour la reconstruction de trace
Application: DUNE

Graph-NN, Point-cloud NN:

Regroupement des données (clustering) pour la reconstruction de trace
Applications: JUNO, LHC, BELLE-II...

Intelligence Artificielle appliquée à la spectro-identification de radioéléments en environnement radiologique complexe.

Generation de données spectrales
Identification par CNN

Deconvolution, probleme mal-posé:

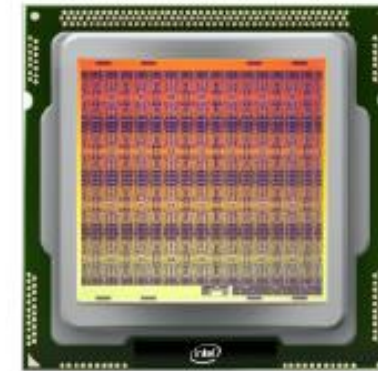
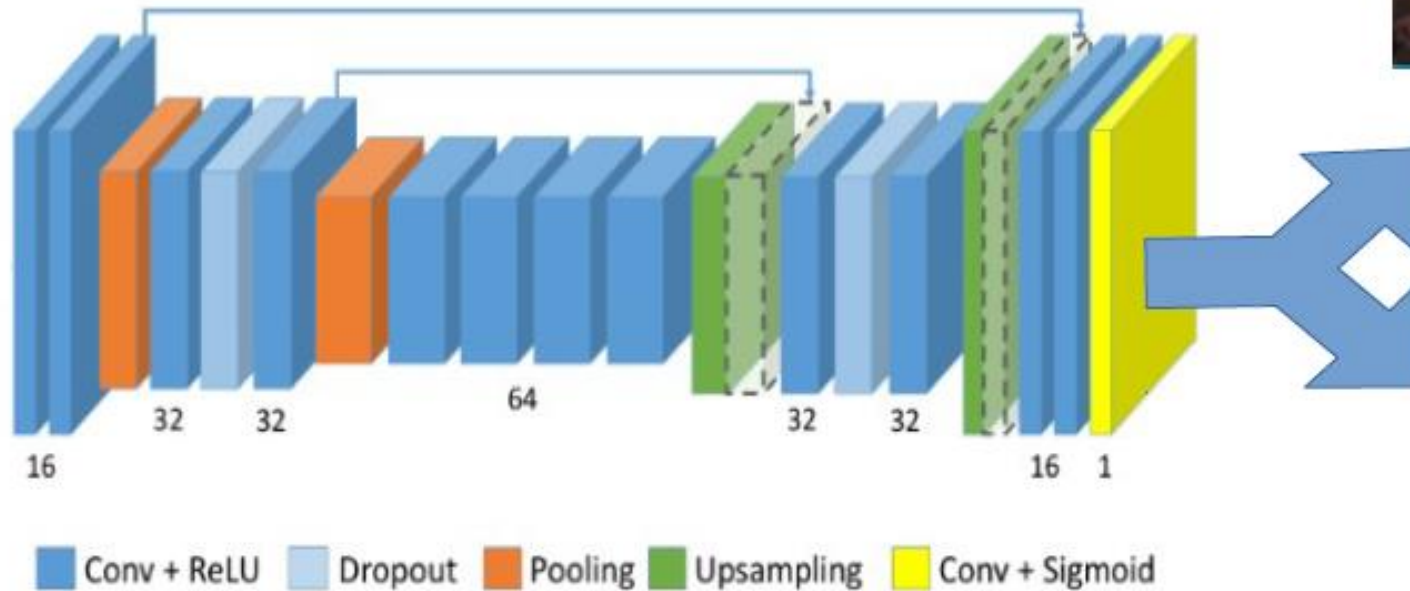
CNN, Autoencodeurs
Melange des galaxies → de-blending

Accélérateurs: constructions

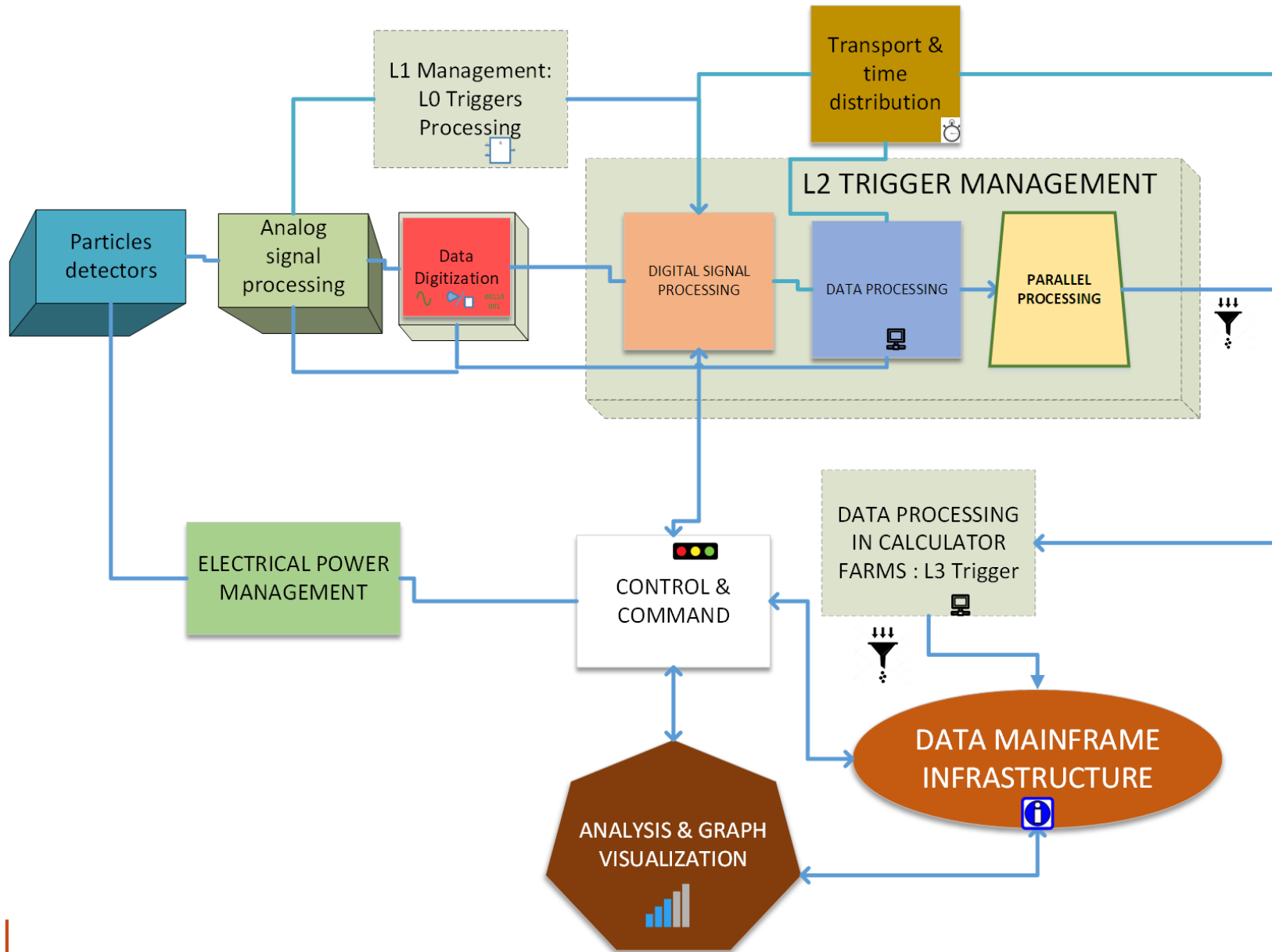
Aimants, monitoring d'anomalies,
Contrôle & commande

Notre objectif

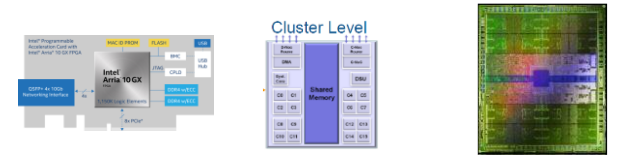
Implémenter les différents types de RN (DNN, CNN, RNN)
dans des SE



My research : Dominant Design in Instruments for research in fundamental physics



Intelligent algorithms



- **Reduced Data and Selection management:**
 - L1: FPGA, ASIC, SNN
 - L2: FPGA, GPU, SNN
 - L3: GPU, MPPA, Accelerated Card
 - DSP post-ADC

Challenges in the field

LHCb – 2032 ~2000 Exabytes/year
ATLAS+CMS 2027 ~ 260 Exabytes/year
Square Kilometers Array – 2030 ~ 30000 EB/year

2021 global Ethernet Dataflow ~2800 EB/year

DataStream before storage
LHCb – 2032 ~500TB/s
ATLAS+CMS 2027 ~ 20-40 TB/s

Forecast cost of storing data to disk (Annual)
LHCb – 2032 ~2,5 Billions of €
ATLAS+CMS 2027 ~ 325 Millions of €

Challenge: Real-time data reduction to avoid disk storage (very expensive):

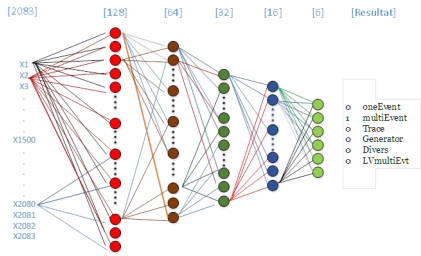
- **Embedded algorithms in decision nodes**
- **Optimize processing (classifications, Prediction, Selection)**
- **Use a mixed GPU, MPPA, FPGA**



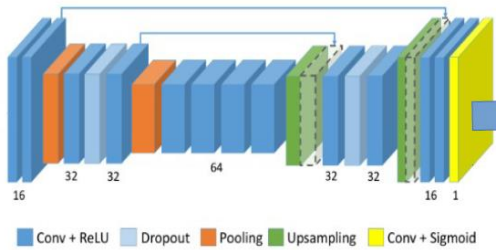
- **Use powerfull hardware component to compute ML Model**
- **And deploy them in ours instruments**

WHAT WE COULD DO WITH ML

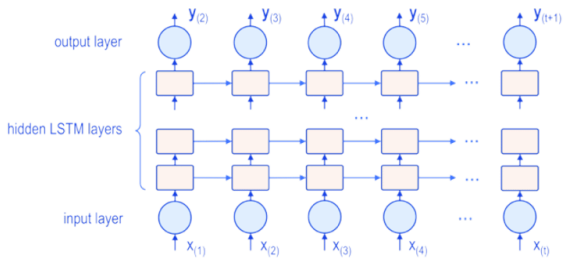
DEEP NEURAL NETWORK



CONVOLUTIONAL NEURAL NETWORK



RECURRENT NEURAL NETWORK



DECISION TREE



RANDOM FOREST



Off-line

- Signal generation
- Design Optimisation

On-line

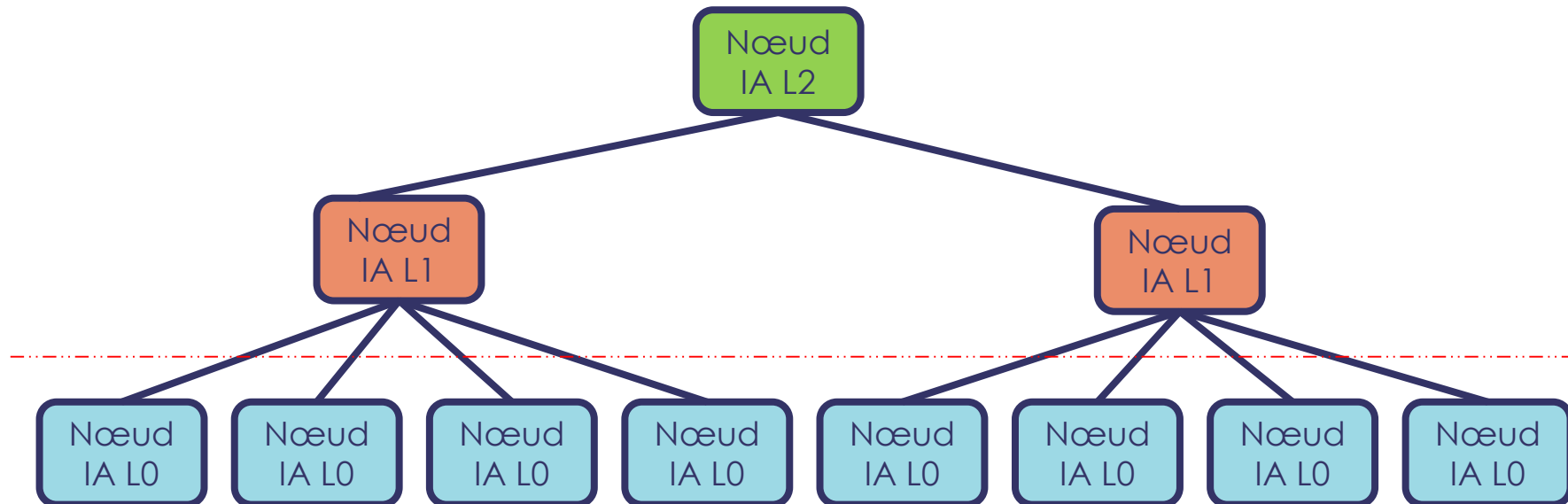
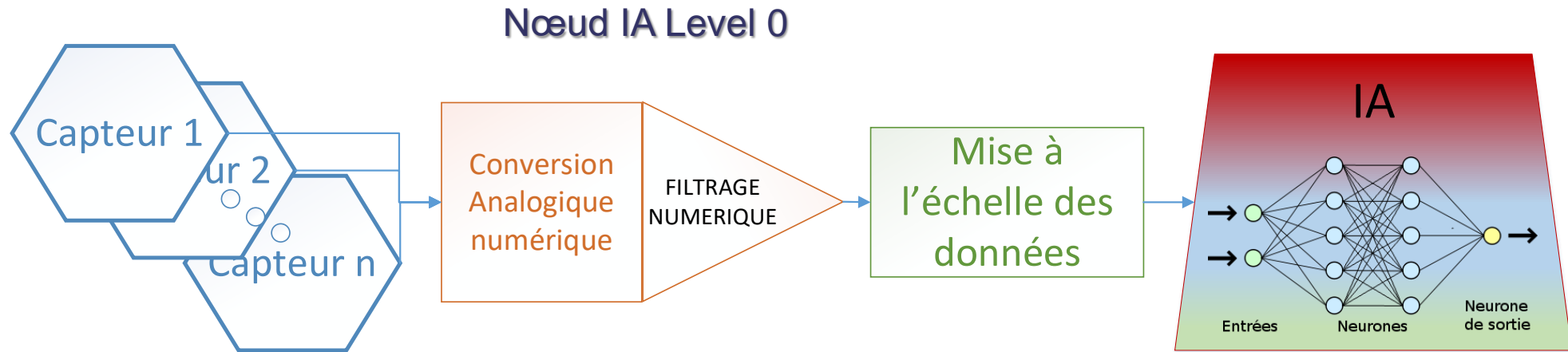
- Instrument Optimization
 - Signal recognition
 - Pile-Up recovery
 - Signal deconvolution
- Selection/ Classification/ Decision
 - Data selection
 - Data parameters prediction
 - Denoizing
- Data Compression
 - Reduced data format

L1

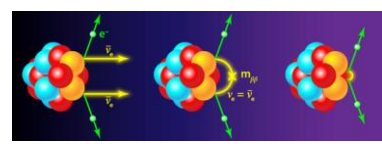
L2

L3



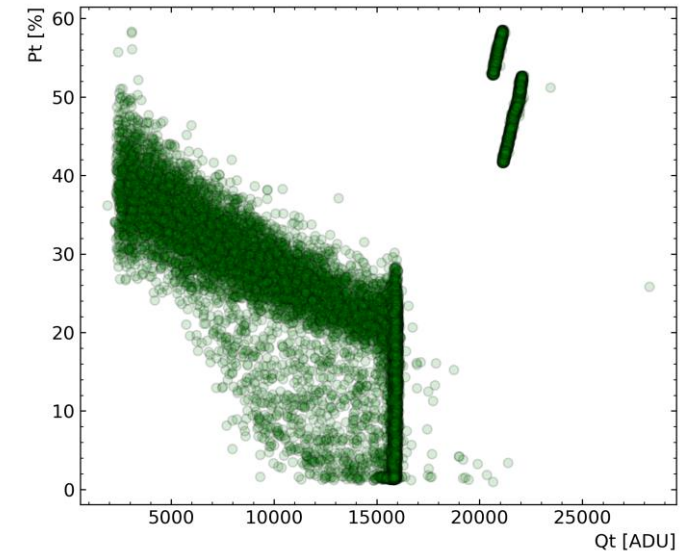
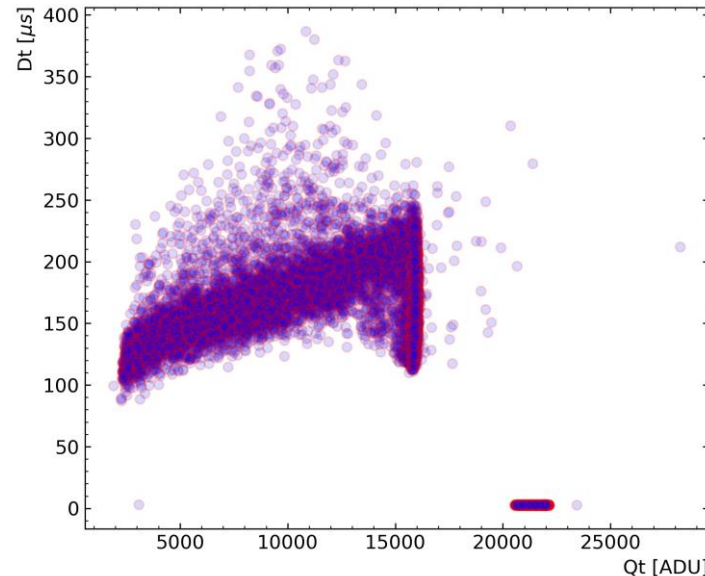
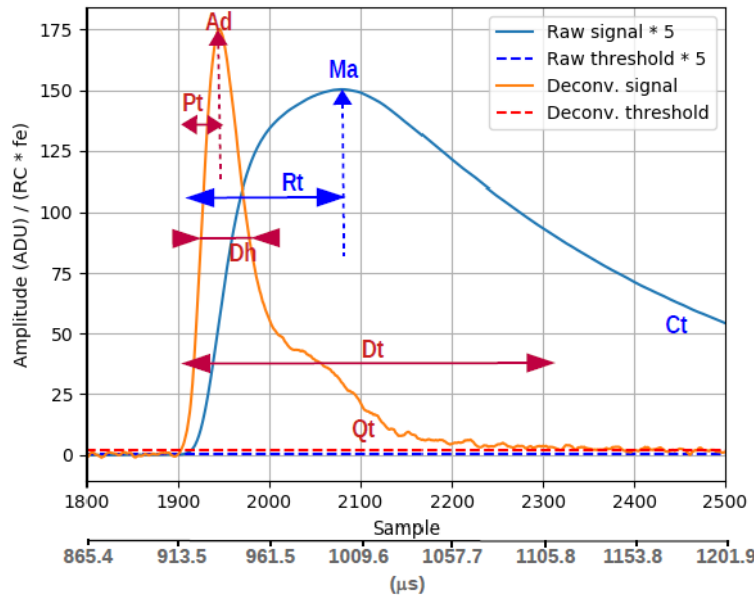


Remplacer le traitement classique du signal



- Avoir la meilleure précision sur la mesure
- Avoir la meilleure information sur la mesure
- Avoir l'information la plus exhaustive sur la mesure

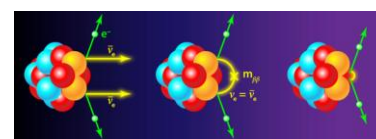
- Energie
- étiquette en temps
- forme d'ondes



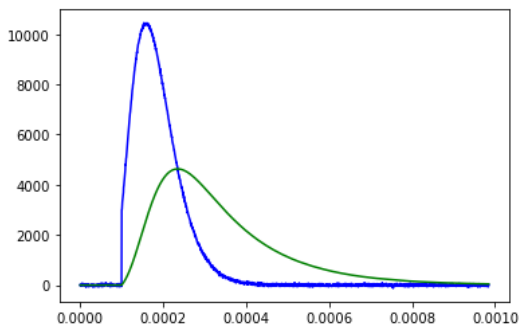
→ Analyse par déconvolution de la chaîne électronique off-line

→ Analyse par déconvolution de la chaîne électronique on-line : Utilisation des méthodes d'apprentissage

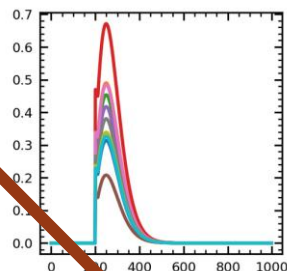
Générer des données réalistes rapidement



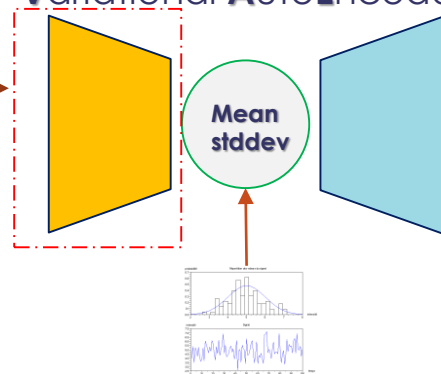
Issus des mesures : Modèle H(p)



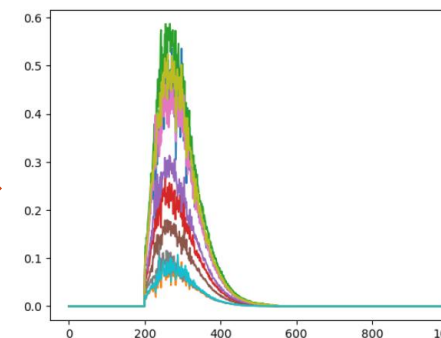
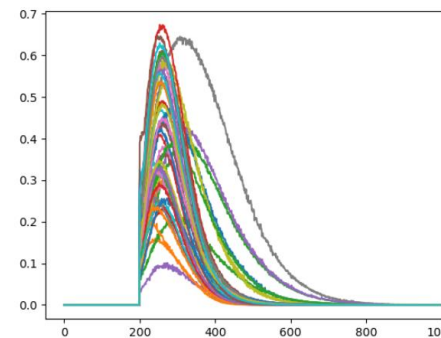
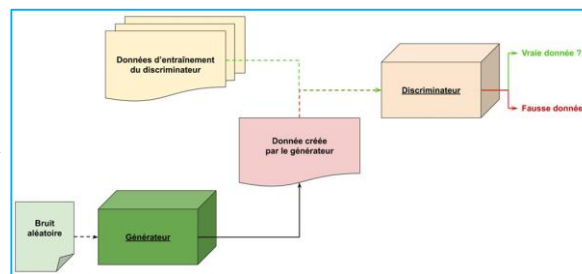
Cas limité du modèle



Variational AutoEncoder



Generative Adversal Network



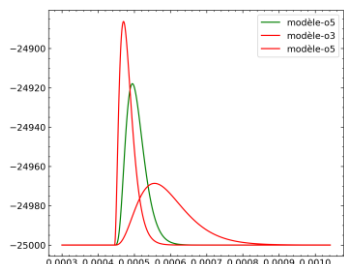
Génération de cas génériques avec ces deux modèles

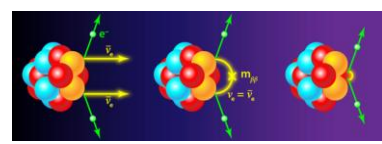
Millions de modèles de signaux

```
import sympy as sp
A,b = sp.symbols("A b", real=True)
G = A/(b*s+1)**3
h_t = inverse_laplace_transform(G, s, t)
print(h_t)
dh_t = sp.diff(h_t,t)
print(dh_t)
```

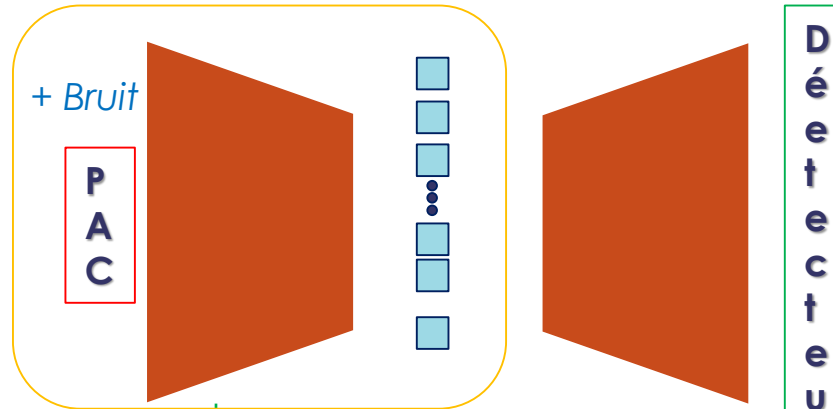
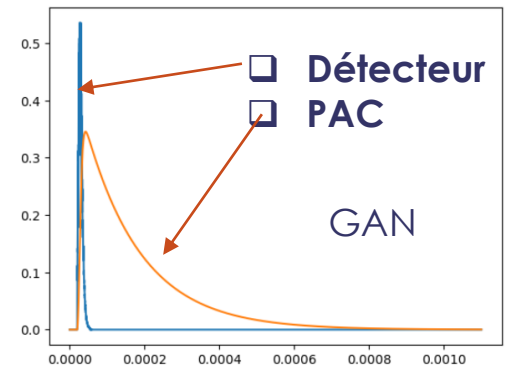
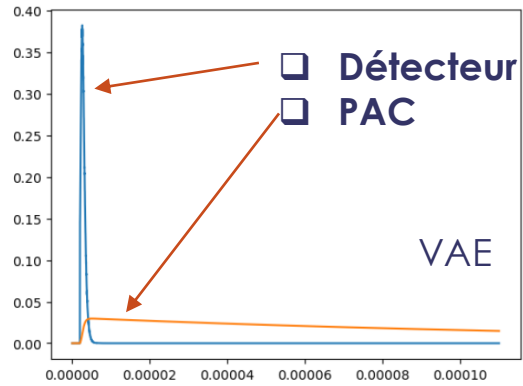
$$A*t**2*exp(-t/b)*Heaviside(t)/(2*b**3)$$

$$A*t**2*exp(-t/b)*DiracDelta(t)/(2*b**3) + A*t*exp(-t/b)*Heaviside(t)/b**3 - A*t**2*exp(-t/b)*Heaviside(t)/(2*b**4)$$

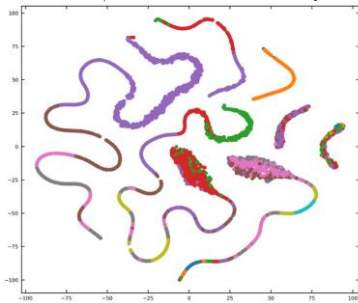




Preampli de charge



- Espace latent
- Caractéristiques du signal
- N paramètres



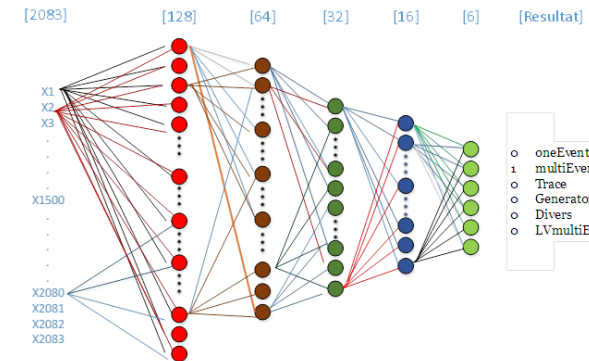
Exemple avec 6 paramètres

Déconvolution on line

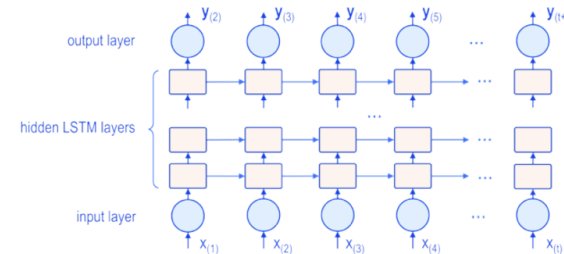


Mise en œuvre dans FPGA

- Régression pour déterminer l'énergie
- Régression pour la mesure d etemps



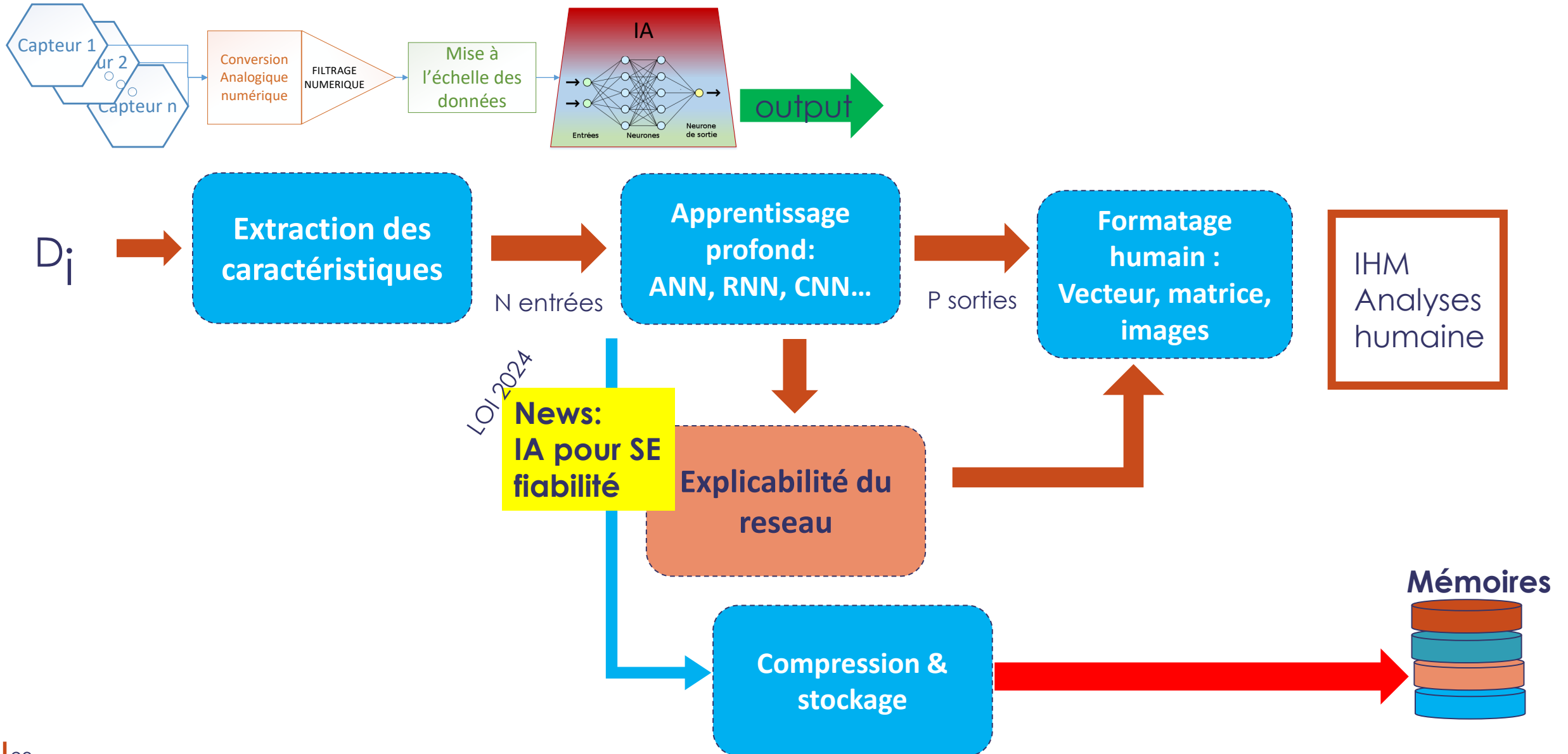
Classification:
Evenement rare
Evenement connu



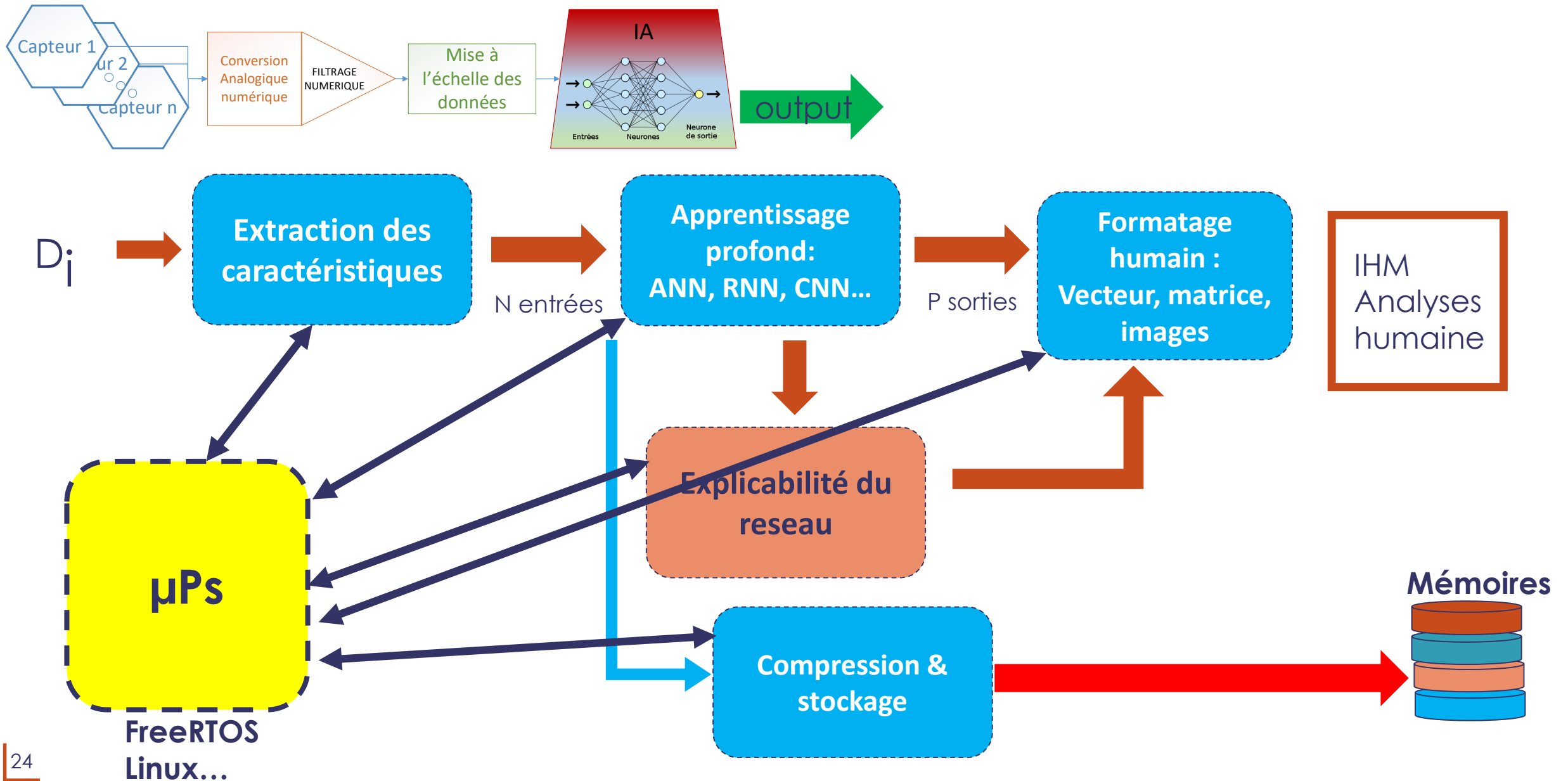
Régression
Erreur balistique
Erreur d'empilement
compression

Remplacement du TNS classique (plus besoin de shaper)

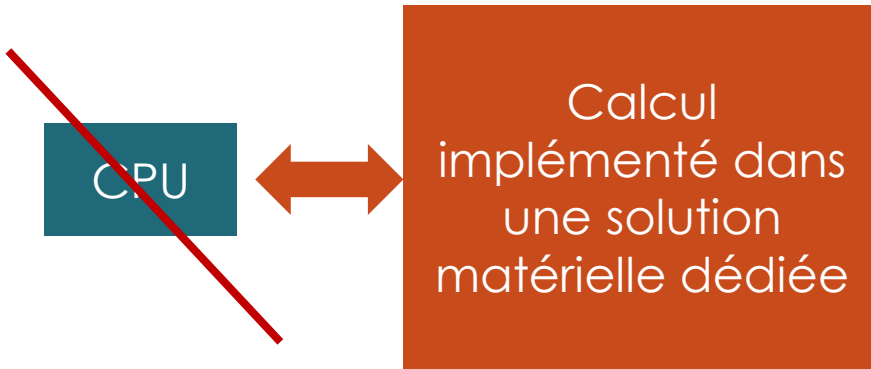
FPGA: L'architecture générale / idéale



FPGA: L'architecture generale / ideale



IA embarqué : 2 solutions



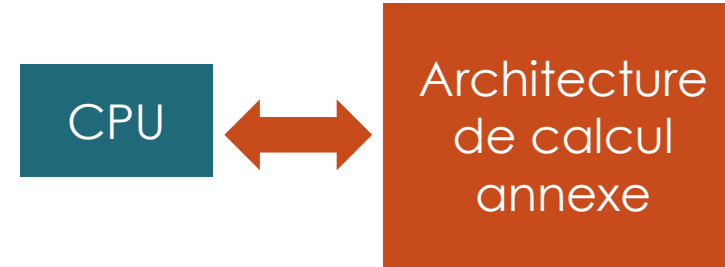
Spatial Accelerators

-- basé sur des fonctions matérielles dédiées aux calculs sans logiciel (calcul matriciel)

ASIC
Circuit neuromorphique
FPGA



En cours de développement



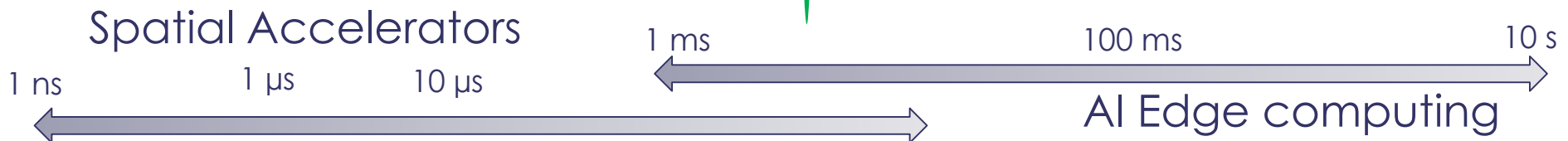
Edge computing

-- Reste basé sur de la programmation logicielle

MMPA
GPU
FPGA
PU redesigned pour l'IA (TPU, KPU...)



Pilote les développements actuels



L'approche embarquée : Une question d'optimisation

Entrées

RNN : modèle avec un énorme nombre de paramètres

Aspects matériels:

Créer un modèle synthétisable
Gestion des ressources

Objectifs

Transférer l'implémentation sur calculateur du réseau au niveau matériel

Optimiser le réseau pour trouver un compromis entre la performance (précision) et ressources matériels

Méthodes

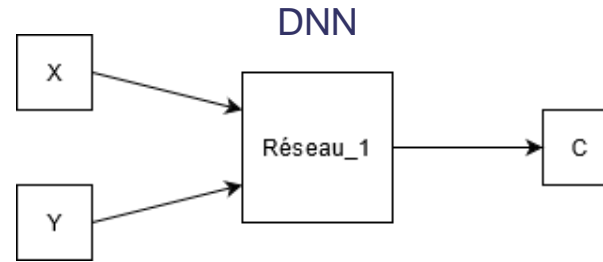
Elagage (Pruning):
Réduire la taille du réseau entraîné

Trouver un réseau optimisé en taille en même temps que l'entraînement (quantification)

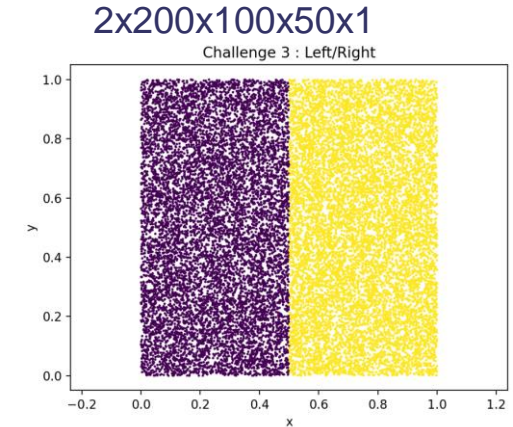
Zynq XcZ7020 & DNN: TF, Keras, QKeras, HLS4ML

XcZ7020: 70€ - 150€

- Dual-core ARM Cortex-9
- Maximum frequency 667MHz-867MHz
- 2x AXI Master & 2 AXI Slave 32 bits
- 53,2 k LUT
- 220 DSP
- 106,4 k Flip-flop



Raw implementation <32,11>



== Performance Estimates

+ Timing:

* Summary:

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	9.883 ns	0.62 ns

+ Latency:

* Summary:

Latency (cycles)		Latency (absolute)		Interval		Pipeline
min	max	min	max	min	max	Type
382	386	3.775 us	3.815 us	100	100	dataflow

== Utilization Estimates

* Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2858	-
FIFO	0	-	3515	30732	-
Instance	229	1006	133648	95258	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	6336	-
Register	-	-	706	-	-
Total	229	1006	137869	135184	0
Available	280	220	106400	53200	0
Utilization (%)	81	457	129	254	0

Model: Keras,
 No Pruning:
 tf.keras.optimizers,
 Quantification:
 HLS4ML

```

=====
== Performance Estimates
=====
+ Timing:
  * Summary:
  +-----+-----+-----+-----+
  | Clock | Target | Estimated | Uncertainty |
  +-----+-----+-----+-----+
  | ap_clk | 5.00 ns | 9.143 ns | 0.62 ns |
  +-----+-----+-----+-----+
+ Latency:
  * Summary:
  +-----+-----+-----+-----+-----+-----+
  | Latency (cycles) | Latency (absolute) | Interval | Pipeline |
  | min | max | min | max | min | max | Type |
  +-----+-----+-----+-----+-----+-----+
  | 374 | 378 | 3.419 us | 3.456 us | 100 | 100 | dataflow |
  +-----+-----+-----+-----+-----+-----+
  
```

```

=====
== Utilization Estimates
=====
* Summary:
+-----+-----+-----+-----+-----+-----+
| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
+-----+-----+-----+-----+-----+-----+
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 2858 | - |
| FIFO | 0 | - | 3515 | 18114 | - |
| Instance | 101 | 255 | 36726 | 82229 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 6336 | - |
| Register | - | - | 706 | - | - |
+-----+-----+-----+-----+-----+-----+
| Total | 101 | 255 | 40947 | 109537 | 0 |
+-----+-----+-----+-----+-----+-----+
| Available | 280 | 220 | 106400 | 53200 | 0 |
+-----+-----+-----+-----+-----+-----+
| Utilization (%) | 36 | 115 | 38 | 205 | 0 |
+-----+-----+-----+-----+-----+-----+
  
```

Model:
 Quantification:
 QKeras,
 No Pruning:
 tf.keras.optimizers,

```

=====
== Performance Estimates
=====
+ Timing:
  * Summary:
  +-----+-----+-----+-----+
  | Clock | Target | Estimated | Uncertainty |
  +-----+-----+-----+-----+
  | ap_clk | 5.00 ns | 9.143 ns | 0.62 ns |
  +-----+-----+-----+-----+
+ Latency:
  * Summary:
  +-----+-----+-----+-----+-----+-----+
  | Latency (cycles) | Latency (absolute) | Interval | Pipeline |
  | min | max | min | max | min | max | Type |
  +-----+-----+-----+-----+-----+-----+
  | 373 | 377 | 3.410 us | 3.447 us | 100 | 100 | dataflow |
  +-----+-----+-----+-----+-----+-----+
  
```

```

=====
== Utilization Estimates
=====
* Summary:
+-----+-----+-----+-----+-----+-----+
| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
+-----+-----+-----+-----+-----+-----+
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 2850 | - |
| FIFO | 0 | - | 3505 | 18054 | - |
| Instance | 100 | 253 | 36030 | 81756 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 6318 | - |
| Register | - | - | 704 | - | - |
+-----+-----+-----+-----+-----+-----+
| Total | 100 | 253 | 40239 | 108978 | 0 |
+-----+-----+-----+-----+-----+-----+
| Available | 280 | 220 | 106400 | 53200 | 0 |
+-----+-----+-----+-----+-----+-----+
| Utilization (%) | 35 | 115 | 37 | 204 | 0 |
+-----+-----+-----+-----+-----+-----+
  
```

Model 2

Model: Keras,
Pruning:
tf.keras.optimizers,
Quantification:
HLS4ML
Optim for resources

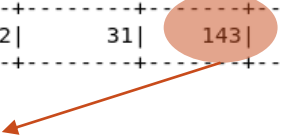
```
=====
== Performance Estimates
=====
+ Timing:
  * Summary:
  +-----+-----+-----+-----+
  | Clock | Target | Estimated | Uncertainty |
  +-----+-----+-----+-----+
  | ap_clk | 5.00 ns | 9.143 ns | 0.62 ns |
  +-----+-----+-----+-----+

+ Latency:
  * Summary:
  +-----+-----+-----+-----+-----+-----+
  | Latency (cycles) | Latency (absolute) | Interval | Pipeline |
  | min | max | min | max | min | max | Type |
  +-----+-----+-----+-----+-----+-----+
  | 2476 | 2480 | 22.638 us | 22.675 us | 1000 | 1000 | dataflow |
  +-----+-----+-----+-----+-----+-----+

=====
== Utilization Estimates
=====
* Summary:
+-----+-----+-----+-----+-----+-----+
| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
+-----+-----+-----+-----+-----+-----+
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 2858 | - |
| FIFO | 0 | - | 3515 | 18314 | - |
| Instance | 25 | 27 | 29254 | 48722 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 6336 | - |
| Register | - | - | 706 | - | - |
+-----+-----+-----+-----+-----+-----+
| Total | 25 | 27 | 33475 | 76230 | 0 |
+-----+-----+-----+-----+-----+-----+
| Available | 280 | 220 | 106400 | 53200 | 0 |
+-----+-----+-----+-----+-----+-----+
| Utilization (%) | 8 | 12 | 31 | 143 | 0 |
+-----+-----+-----+-----+-----+-----+
```



Code HLS generé avec HLS4ML



Model 3 - #PRAGMA ARRAY_PARTITION dim=3

```
=====  
== Performance Estimates  
=====
```

```
+ Timing:  
* Summary:  
+-----+-----+-----+-----+  
| Clock | Target | Estimated| Uncertainty|  
+-----+-----+-----+-----+  
| ap_clk | 5.00 ns | 9.062 ns | 0.62 ns |  
+-----+-----+-----+-----+  
  
+ Latency:  
* Summary:  
+-----+-----+-----+-----+-----+-----+-----+  
| Latency (cycles) | Latency (absolute) | Interval | Pipeline |  
| min | max | min | max | min | max | Type |  
+-----+-----+-----+-----+-----+-----+-----+  
| 199 | 203 | 1.803 us | 1.840 us | 50 | 50 | dataflow |  
+-----+-----+-----+-----+-----+-----+-----+
```

```
=====  
== Utilization Estimates  
=====
```

```
* Summary:  
+-----+-----+-----+-----+-----+  
| Name | BRAM_18K | DSP48E | FF | LUT | URAM |  
+-----+-----+-----+-----+-----+  
| DSP | - | - | - | - | - |  
| Expression | - | - | 0 | 2858 | - |  
| FIFO | 0 | - | 3515 | 14157 | - |  
| Instance | 171 | 10 | 35019 | 207910 | - |  
| Memory | - | - | - | - | - |  
| Multiplexer | - | - | - | 6336 | - |  
| Register | - | - | 706 | - | - |  
+-----+-----+-----+-----+-----+  
| Total | 171 | 10 | 39240 | 231261 | 0 |  
+-----+-----+-----+-----+-----+  
| Available | 280 | 220 | 106400 | 53200 | 0 |  
+-----+-----+-----+-----+-----+  
| Utilization (%) | 61 | 4 | 36 | 434 | 0 |  
+-----+-----+-----+-----+-----+
```

Model 4: REMOVE ALL #PRAGMA ARRAY_PARTITION in HLS code

```
=====
== Performance Estimates
=====
+ Timing:
  * Summary:
  +-----+-----+-----+
  | Clock | Target | Estimated | Uncertainty |
  +-----+-----+-----+
  | ap_clk | 5.00 ns | 7.144 ns | 0.62 ns |
  +-----+-----+-----+
+ Latency:
  * Summary:
  +-----+-----+-----+-----+
  | Latency (cycles) | Latency (absolute) | Interval | Pipeline |
  | min | max | min | max | min | max | Type |
  +-----+-----+-----+-----+
  | 91251 | 91255 | 0.652 ms | 0.652 ms | 50000 | 50000 | dataflow |
  +-----+-----+-----+-----+
```

```
=====
== Utilization Estimates
=====
* Summary:
+-----+-----+-----+-----+-----+
| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
+-----+-----+-----+-----+-----+
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 238 | - |
| FIFO | 0 | - | 255 | 1326 | - |
| Instance | 25 | 27 | 33689 | 36552 | - |
| Memory | 5 | - | 0 | 0 | 0 |
| Multiplexer | - | - | - | 486 | - |
| Register | - | - | 56 | - | - |
+-----+-----+-----+-----+-----+
| Total | 30 | 27 | 34000 | 38602 | 0 |
+-----+-----+-----+-----+-----+
| Available | 280 | 220 | 106400 | 53200 | 0 |
+-----+-----+-----+-----+-----+
| Utilization (%) | 10 | 12 | 31 | 72 | 0 |
+-----+-----+-----+-----+-----+
```



At one point, we need to optimize HLS code
At one point, we could develop directly in VHDL

Discussion autour des FPGA: Spatial Accelerators

- Zynq + HLS4ML: io_parallel / io_stream / Reusefactor / Resource / Latency

Une question de budget
(nombre de voies DAQ)

	ARTY-Z7 CH1	ARTY Z7 CH3	ARTY Z7 CH3 Optimisé	ZCU102 CH3	CH4	ZCU102 CH4 Qbit<16,2>	ZCU102 CH5	ZCU102 CH7 Optim Ressource	ZCU102 CH7 Optim Latence	ARTY Z7 CH7 Optim HLS Stream
Nombre de cellules	16	25801	25801	25801	658951	658951	156951	156951	156951	156951
Horloge de reference	4,166ns	9,408ns	9,408ns	4,396ns		4,369ns	4,369ns	4,369ns	4,028ns	9,410ns
Temps de latence	70ns	19,804us	19,804us	2,510us		5,510us	5,510us	10,010us	10,015us	141us
BRAM	0%	41%	8%	6%		36%	18%	9%	15%	72%
DSP48E	21%	115%	11%	10%		59%	59%	12%	24%	6%
FF	2%	85%	28%	10%		28%	22%	32%	53%	167%
LUT	1%	280%	68%	46%		103%	113%	81%	104%	423%
URAM	0%	0%	0%	0%		0%	0%	0%	0%	0%

€ ←-----→ €€€

depend on VITIS-HLS
#pragma
The way HLS handles
vector/matrix before
DSP

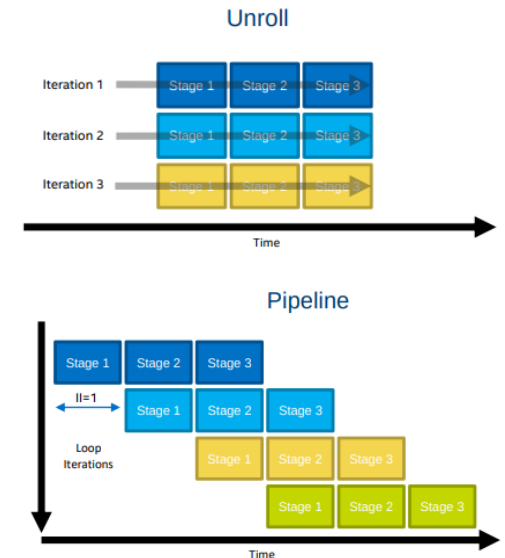
- Intel Arria 10 + Intel HLS (expert HLS) : €€

	ALUTs	FFs	RAMs	MLABs	DSPs
Ch1 vanilla	602 (0%)	547 (0%)	4 (0%)	2 (0%)	1.5 (0%)
Ch1 pipeline	610 (0%)	624 (0%)	4 (0%)	5 (0%)	1.5 (0%)
Ch1 unroll	515 (0%)	245 (0%)	4 (0%)	1 (0%)	0 (0%)
Ch1 u+p	515 (0%)	245 (0%)	4 (0%)	1 (0%)	0 (0%)

	ALUTs	FFs	RAMs	MLABs	DSPs
Ch4 vanilla	4 442 (0%)	6 415 (0%)	355 (1%)	20 (0%)	3.5 (0%)
Ch4 pipeline	5 555 (0%)	10 899 (0%)	362 (1%)	113 (0%)	3.5 (0%)
Ch4 unroll	Problème d'implémentation				
Ch4 u+p					

	ALUTs	FFs	RAMs	MLABs	DSPs
Ch3 vanilla	1 408 (0%)	1 809 (0%)	30 (1%)	12 (0%)	2.5 (0%)
Ch3 pipeline	2 093 (0%)	4 460 (0%)	32 (1%)	48 (0%)	2.5 (0%)
Ch3 unroll	118 041 (14%)	36 737 (2%)	5 (0%)	37 (0%)	0 (0%)
Ch3 u+p	27 524 (3%)	42 546 (2%)	1 855 (68%)	298 (1%)	75 (5%)

	ALUTs	FFs	RAMs	MLABs	DSPs
Ch5 vanilla	1 669 (0%)	2 193 (0%)	32 (1%)	16 (0%)	2.5 (0%)
Ch5 pipeline	2 617 (0%)	6 074 (0%)	35 (1%)	76 (0%)	2.5 (0%)
Ch5 unroll	569 259 (67%)	196 051 (11%)	6 (0%)	40 (0%)	0 (0%)
Ch5 u+p	17 560 (2%)	23 244 (1%)	507 (19%)	237 (1%)	50.5 (3%)

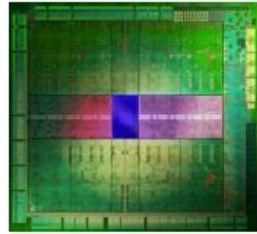


Question of HLS optimisation

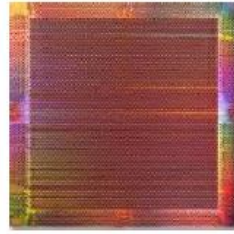
Exemples Technologiques



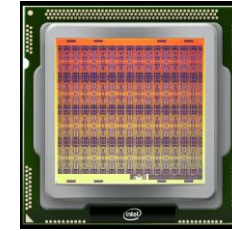
CPUs
MPPA



GPUs



FPGAs



NMC



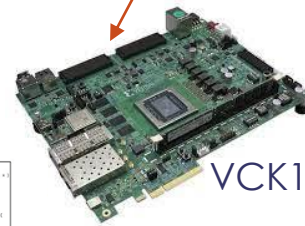
nVidia



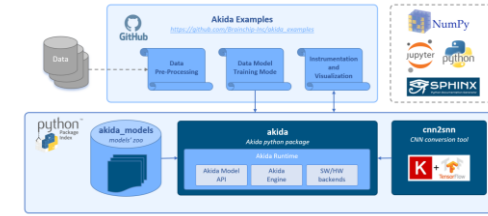
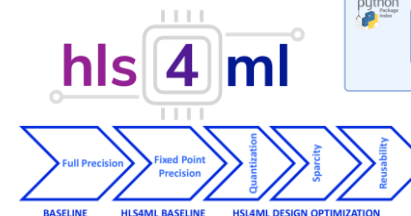
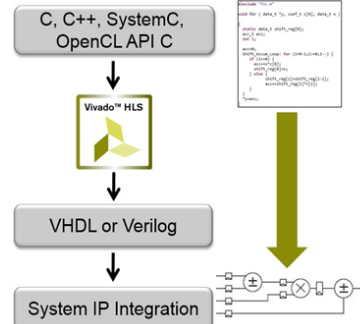
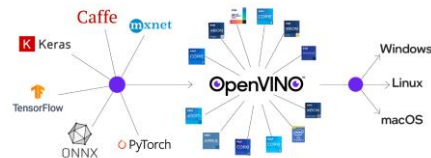
DE10-Agilex



ZCU102,
ZCU104











VCK190

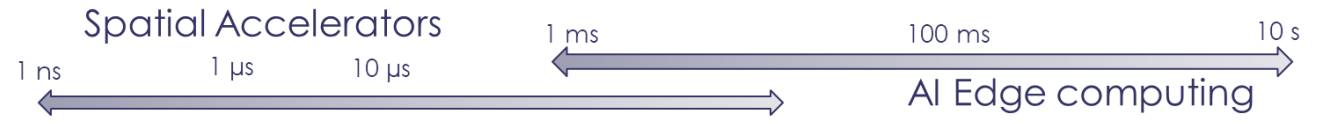


Features with others components

Microcontrollers
AI edge computing
RISC-V Architecture

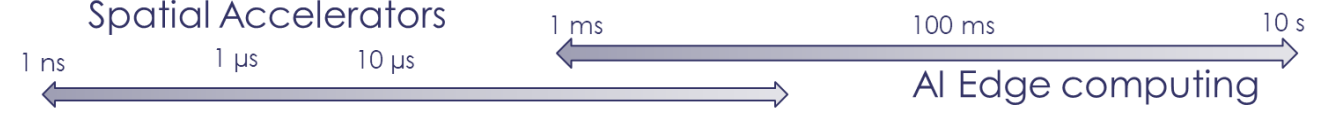
Type	Company	Components	Dev Boards	Pictures	Soft Tools
AI Edge computing	AMD Xilinx	Zynq	ARTY Z7 PYNQ ZCU104 SCU102		VITIS-AI
Spatial acceleration				Tensorflow/Keras + HLS4ML PyTorch + FiNN+Brevitas	
AI Edge computing	AMD Xilinx	VERSAL	VK290		HLS+VITIS-AI
AI Edge computing	nVidia	Jetson	Nano TX2 Xavier NX AGX Xavier Orin		Jetpack SDK + DeepStream SDK + TensorRT
Spatial acceleration	Intel	AGILEX ARRIA	DE10- AGILEX ARRIA 10 GX		Intel HLS
AI Edge computing				FPGA AI Suite OpenVino	
Spatial acceleration	Brainchip	AKD1000	Akida PCIe		Akida MetaTF ML framework
AI Edge computing	SiPEED	Kendryte210 RISC-V	MAIXDUINO		SiPEED SDK + micropython
AI Edge computing	ST Microelectronics	STM32	Nucleo64F411		CubeMX + Cube.AI
AI Edge computing	Analog Devices	MAX78000	MAX78000E VKIT		Maxim Micros SDK

DNN, CNN 1D, RNN, GNN

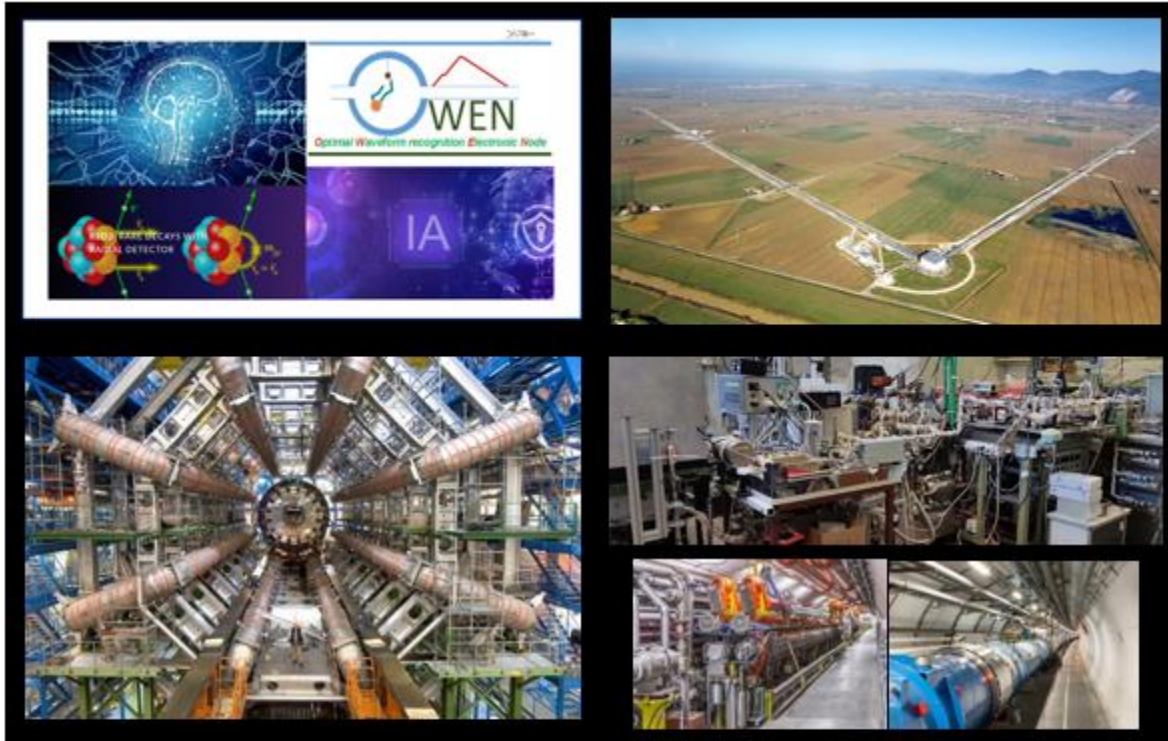


DNN, CNN 1D, RNN, GNN
Spatial Accelerators

Tools not stable



Chaque composant a son logiciel d'IA !



5 laboratoires, 23 contributeurs:

- LLR
- IPHC
- IP2I Lyon
- CPPM
- LP2I Bordeaux

Thème 1 : les particules de hautes énergies

Thème 2 : Les ondes gravitationnelles

Thème 3 : Physique des Neutrinos

Thème 4 : Les instruments optimisés (Analyse spectrale)

Test of Hardware Inferring Neural network

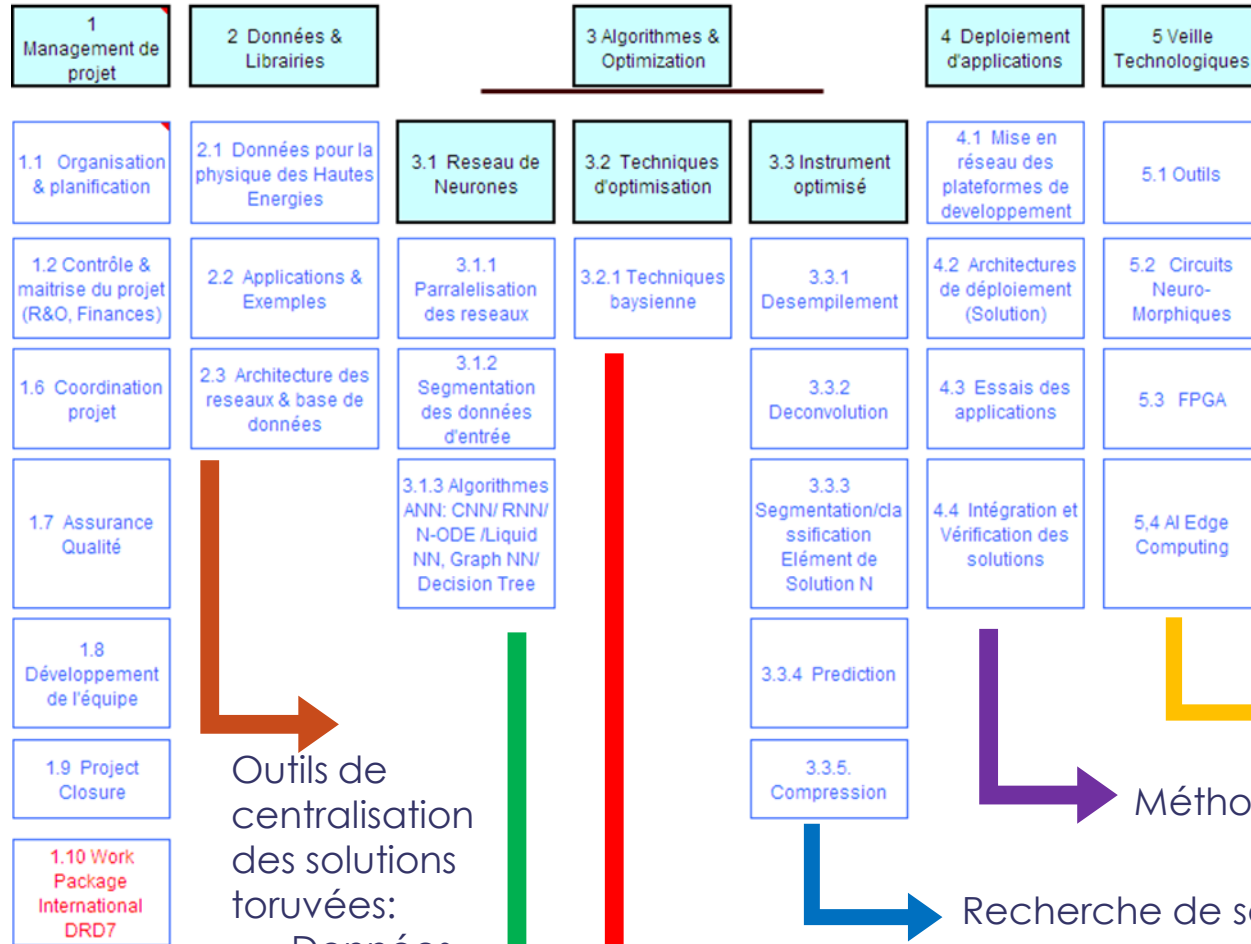


Reseau DAQ – Hardware-Firmware-Software-

1. Investissement dans des matériels avancés
2. Démocratisation de l'accès aux ressources
3. Tests de modèles d'IA
4. Application au dominant design de notre instrumentation
5. Gestion des risques par rapport aux ambitions du projet

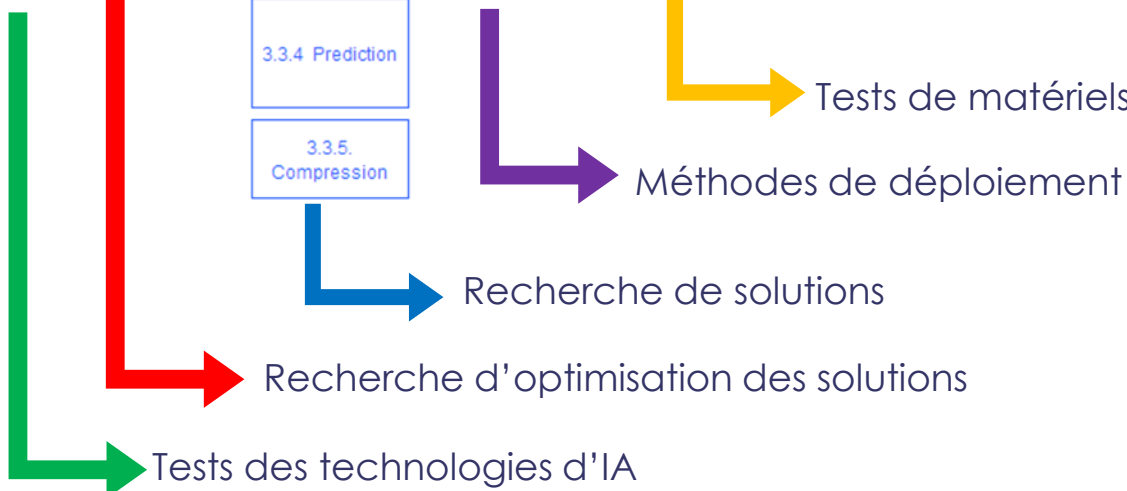


WBS THINK PHASE 2 IS

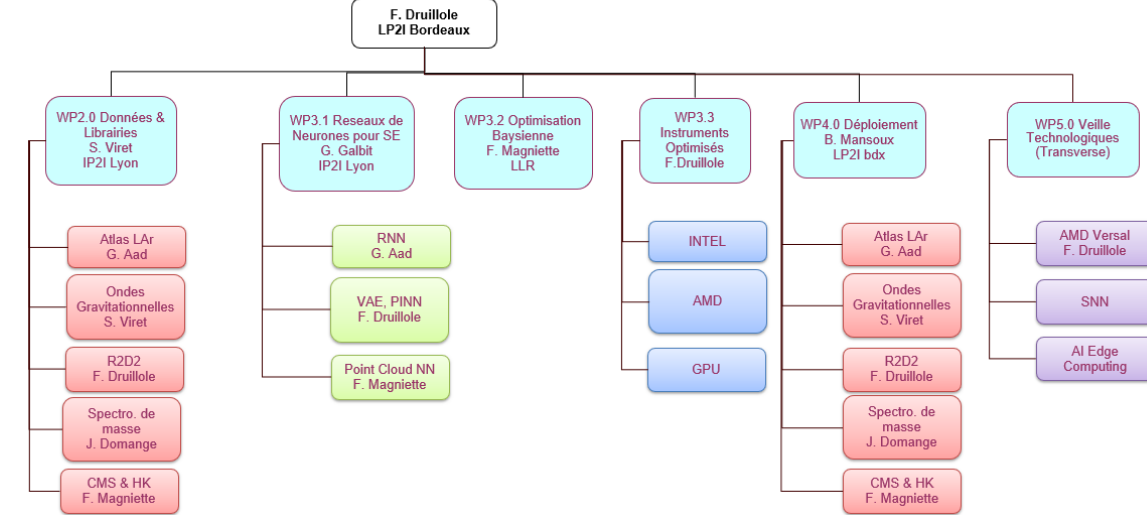


Outils de centralisation des solutions trouvées:

- Données
- Modèles



OBS THINK PHASE 2



les Défis de l'IA embarqué: Une culture à avoir.



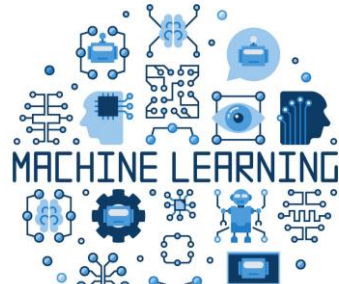
Roles & competencies

- *Data Physicist*
- *System Engineering team*
- *ML Engineer*
- *Software Engineer*
- *Hardware Engineer*
- *Infra & Security teams*

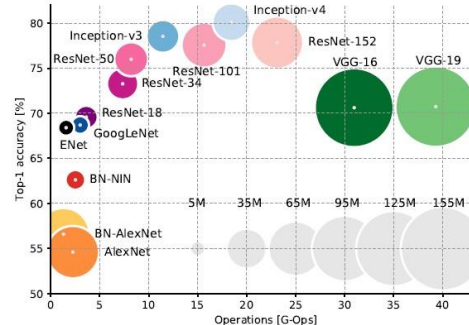


Tools

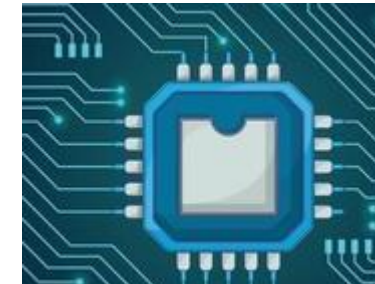
- *ML Tools:*
 - *TF-KERAS, PyTorch ...*
- *HLS4ML (Xilinx...)*
- *HLS*
- *Brevitas & FiNN(Xilinx)*
- *CONIFER (LLR)*
- *N2D2 (CEA)*
- *VHDL*
- ...



Artefacts & ML zoology



- *Model*
- *Code source...*



Digital hardware technologies

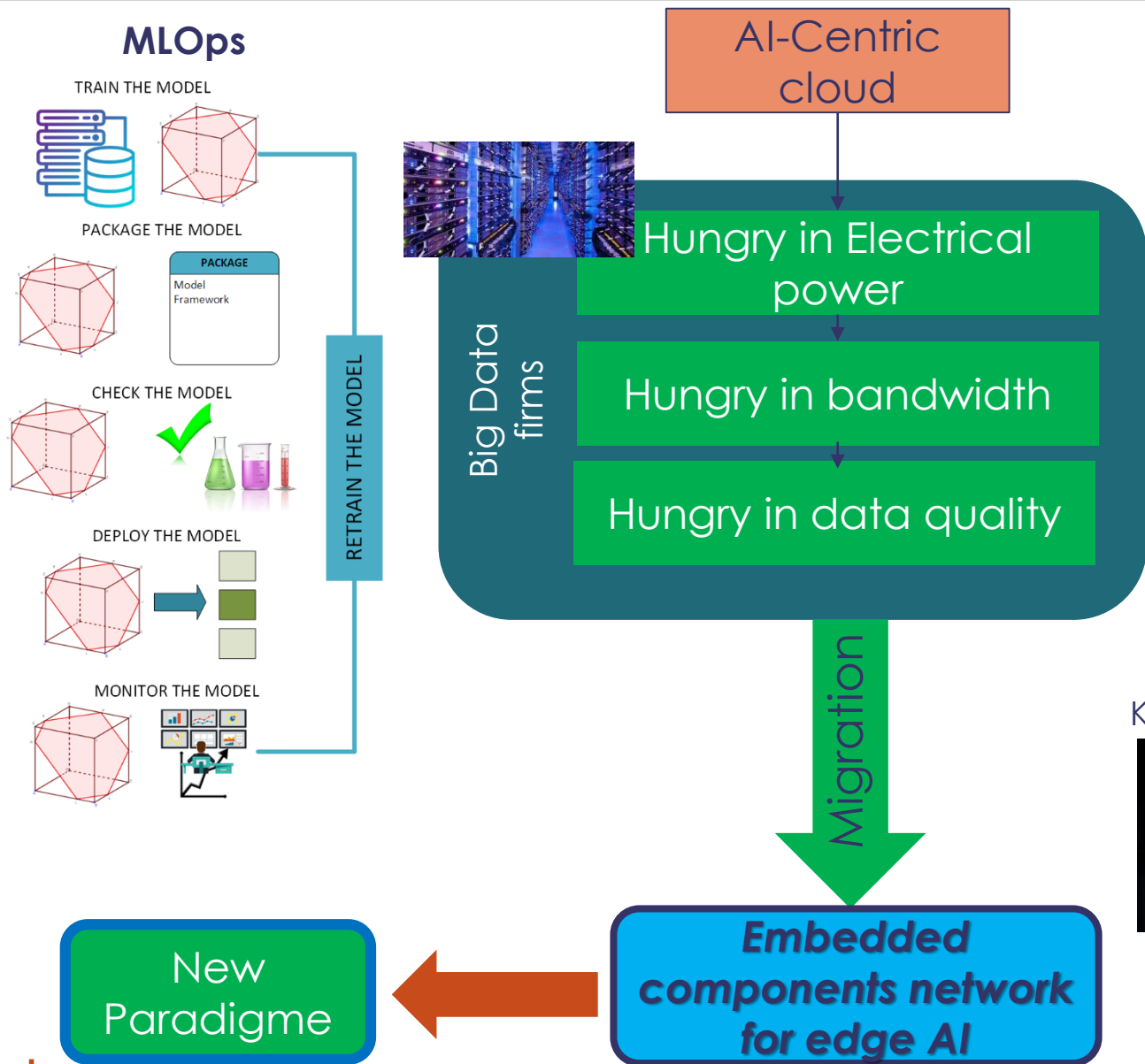
- *CPU*
- *FPGA SOM*
- *SNN*
- *MPPA*
- *GPU*
- ...



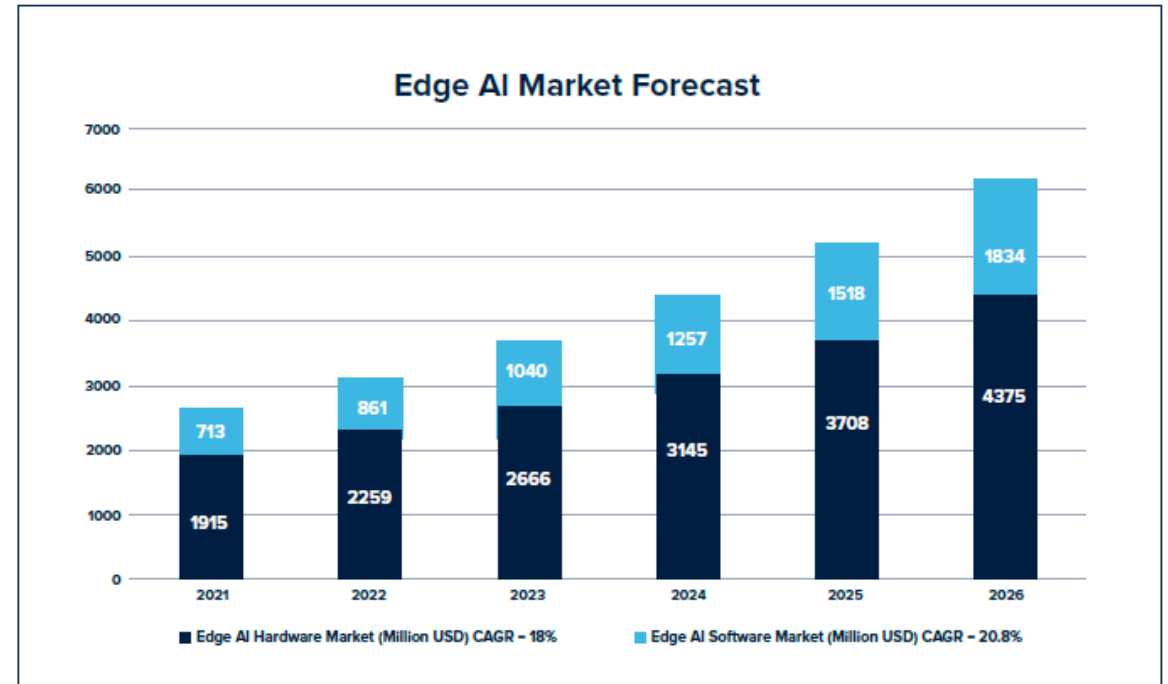
Deployment & Operational AI

- *GitLab/Git*
- *Training Service skew*
- *Model Monitoring*
- *Responsible AI*
- ...

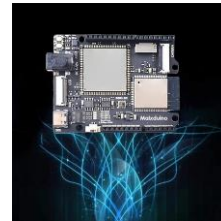
Stakeholders → Responsive AI on the Edge



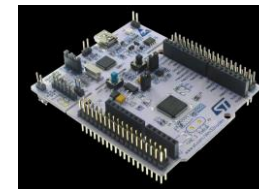
STMicroelectronics 2022



Kendryte K210



STM32 Cube AI



nvidia



Digilent

AMD-Xilinx



Intel

