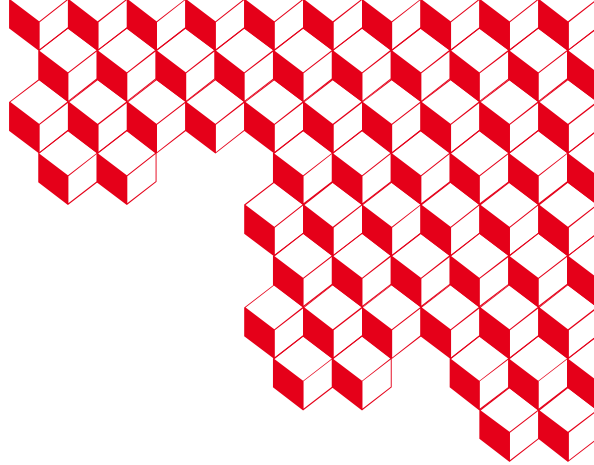
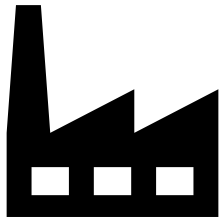


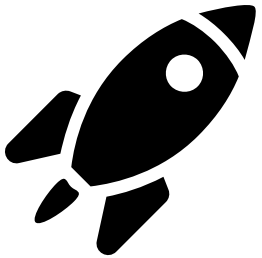
Entrainement robuste et quantification dans Aidge

Augustin Lemesle, Julien Lehmann, Zakaria Chihani

Contexte

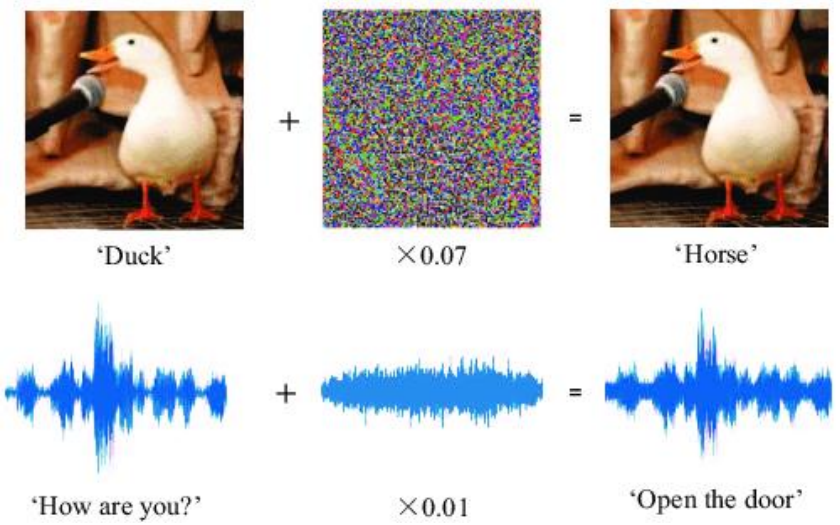
- Obtenir des **garanties fortes** de sûreté dans des systèmes critiques (nucléaire, spatial, aéro, …)
  - Sur des **composants IA embarqués**
  - **Preuve** de sûreté de fonctionnement, robustesse, …
  - Avec un objectif de **certification** du système





Problème

- Les IA statistiques ne sont pas robustes par construction
  - Attaques adverses,
  - Sensibilité aux entrées,…

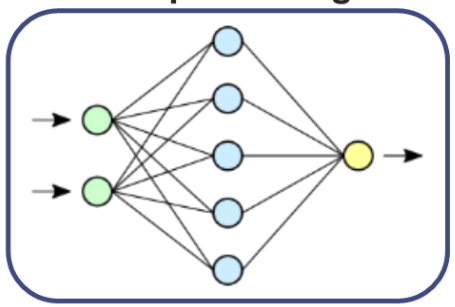


- Les IA sont différentes des logiciels classiques
  - Souvent très gros et plus difficiles à vérifier

Classical software

Loops
If/then/else
Memory allocation

Deep Learning



Huge matrices
Large networks

Idée

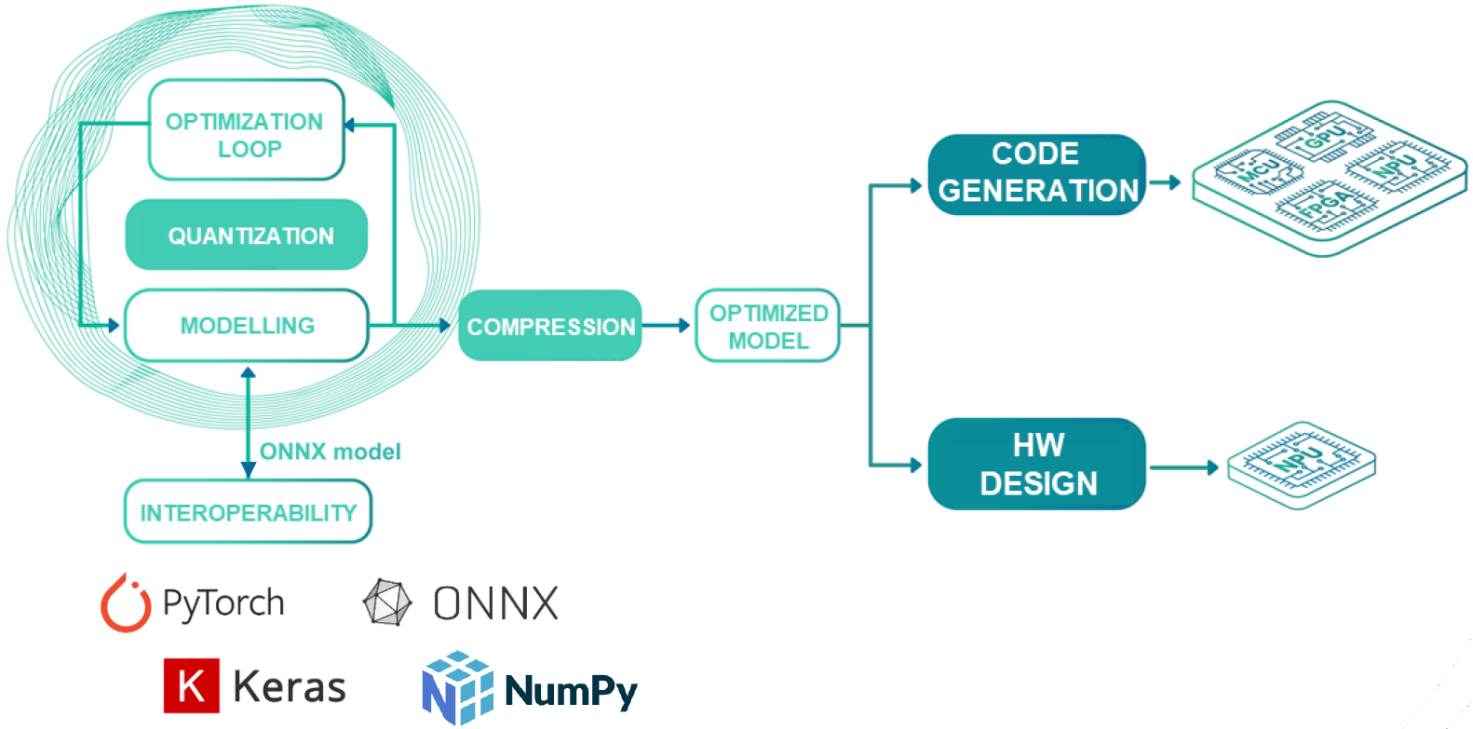
- Utiliser conjointement :
  - des techniques de **quantification de modèle** (en entier 8 bit par exemple),
  - des techniques **d'entrainement robuste,**
  - et de la **vérification formelle,**

- pour avoir des modèles optimisés pour l'embarqué et vérifiés robustes

Aidge

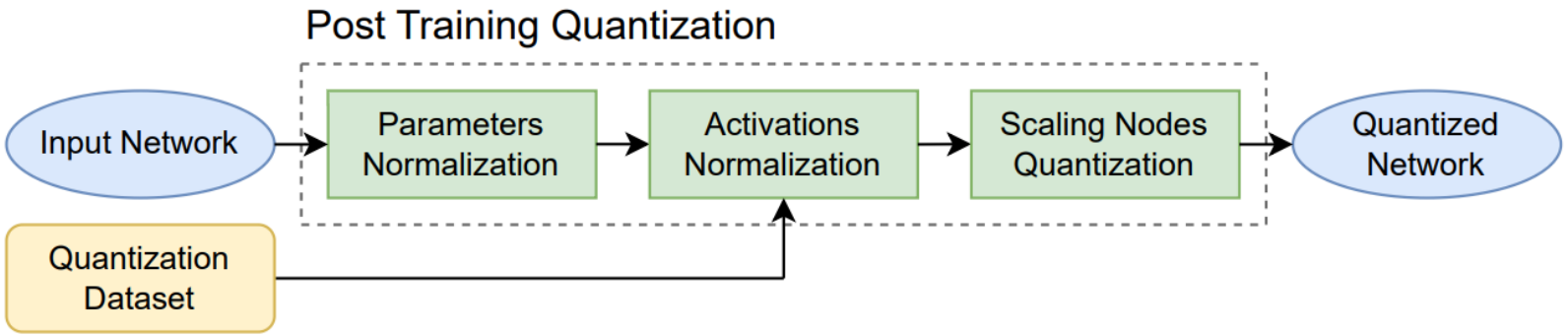
Plateforme open source souveraine pour optimiser et déployer des IA sur des cibles embarquées

- Optimisation et compression des IA (**quantification PTQ** en entiers de 1 à 8 bits)
- **Edition et simplification du graphe de calcul**
- Génération de code pour différentes cibles (C, C++, NVIDIA, FPGA, …)

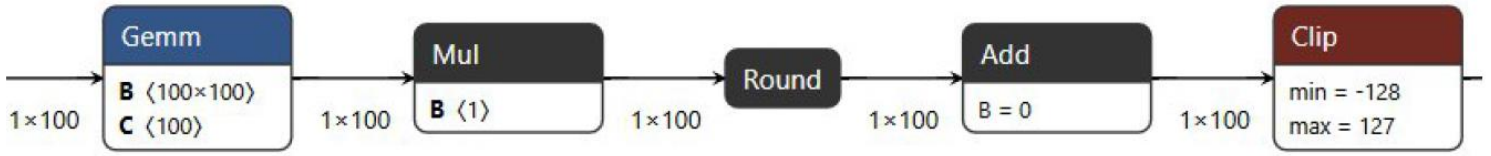


Quantification

Post Training Quantification



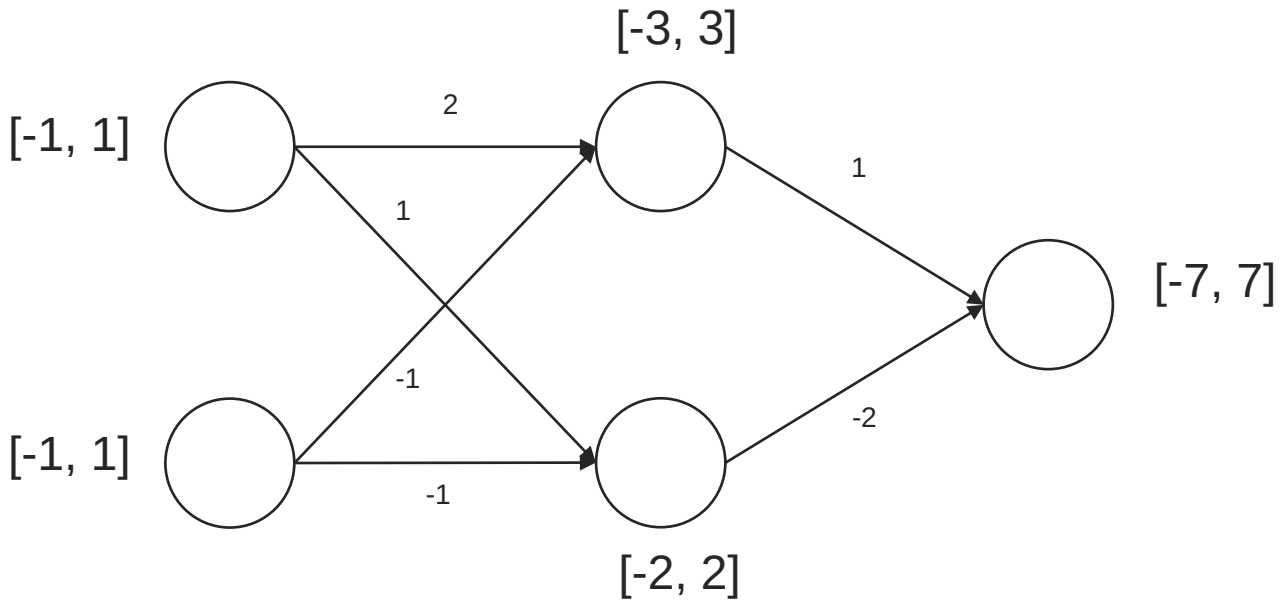
- Normalisation des poids selon le type d'entier choisi (ici 8 bits [-128, 127])
- Deux types de quantification possibles
  - *Fake quantize* : Normalisation mais poids gardés en flottants
  - *True quantize* : Poids en entiers



PyRAT

Garanties formelles de la sûreté d'un réseau de neurone

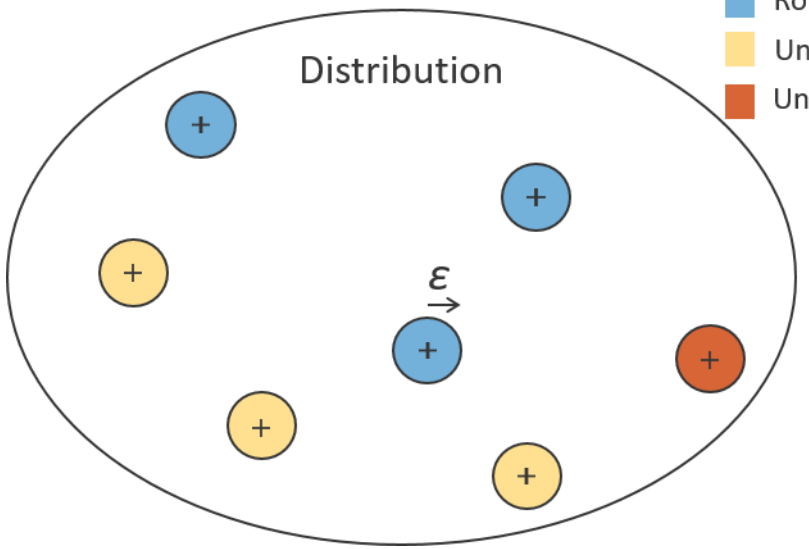
- Créé en 2019, PyRAT utilise l'interprétation abstraite et le calcul matriciel pour passer à l'échelle sur des gros réseaux de neurones
- A l'état de l'art et sur le podium de la VNNCOMP depuis 2023



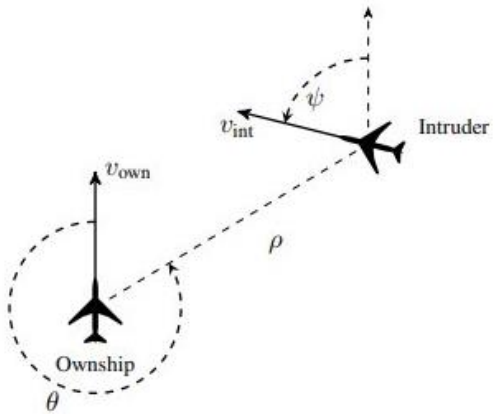
Propriétés vérifiées

- **Robustesse locale**
  - Stabilité de classification face à des perturbation d'intensité ε
  - Sensibilité des capteurs
  - Attaques adverses

- **Sûreté globale**
  - Si X > 0 alors la sortie devrait être Y
  - Si temperature > 18°C alors anomalie détectée,
  - …



$$\forall x', \|x - x'\|_\infty \leq \epsilon, \quad \arg\max N(x) = \arg\max N(x').$$



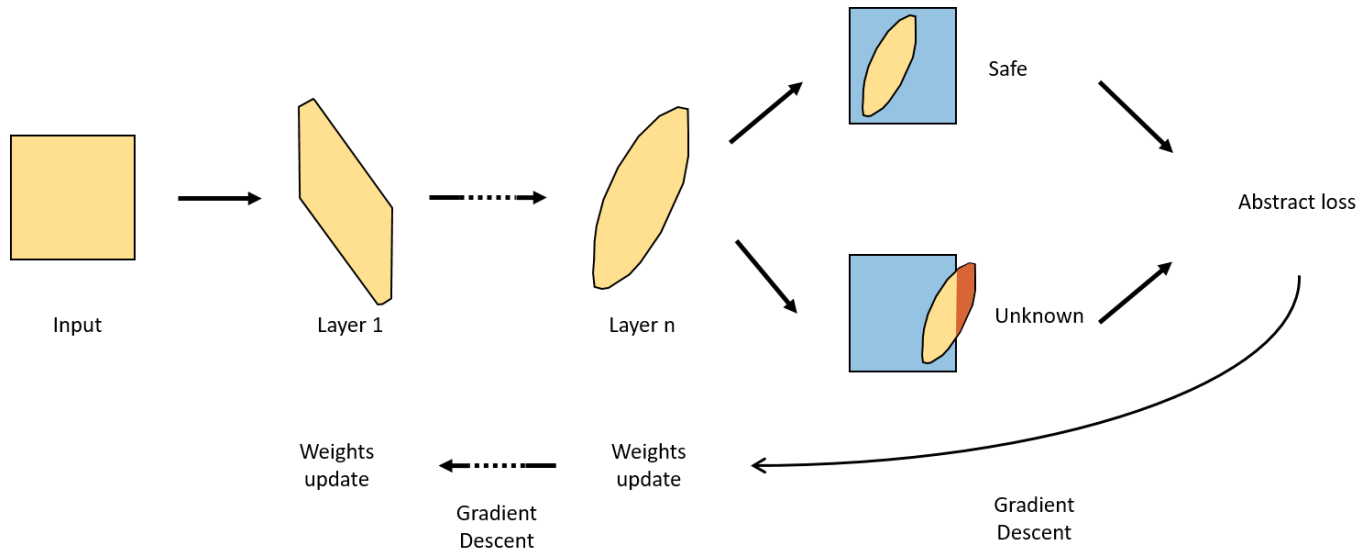
Aircraft Collision Avoidance System

$$\forall \theta \in \left[0, \frac{\pi}{2}\right], v_{int} \in [0, 10], v_{own}, \rho, \psi \Rightarrow Turn\ left$$

Entrainement robuste

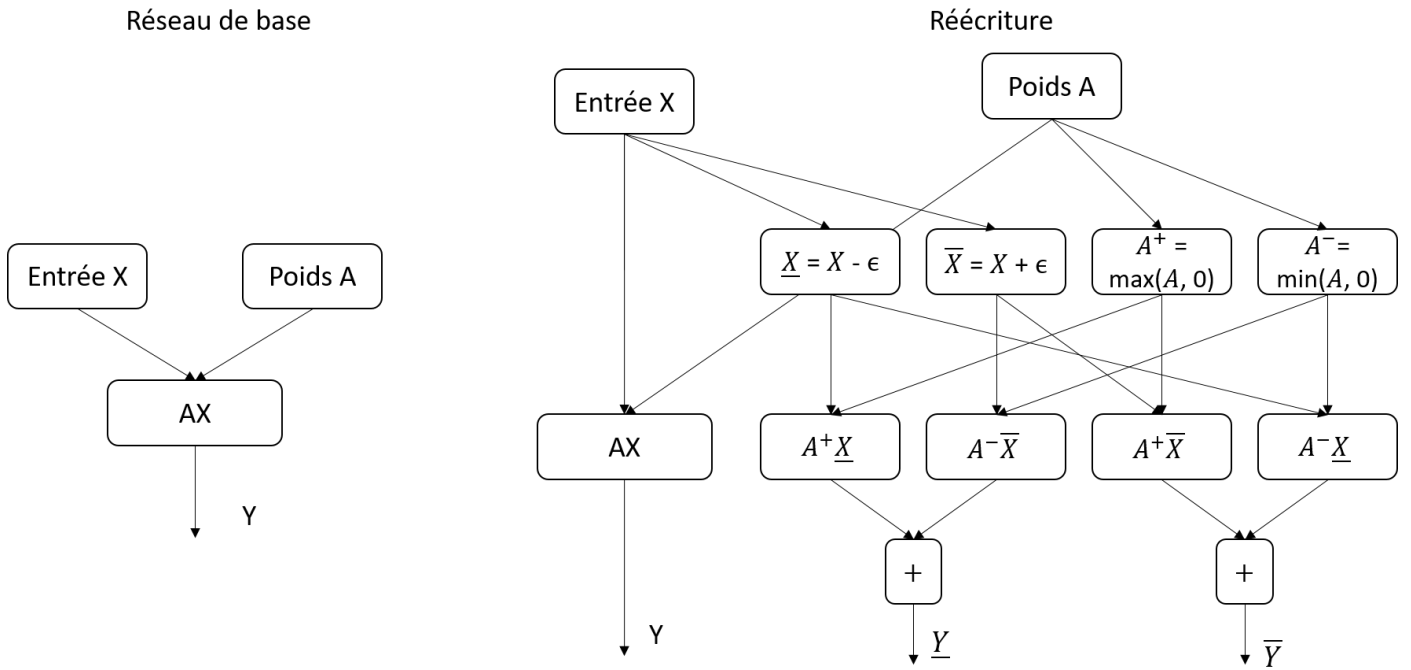
- Si un modèle n'est pas robuste
  - Inclure des méthodes formelles dès l'entrainement avec une loss basée sur l'arithmétique des intervalles

$$L_{total} = \lambda * L_{classic} + \ (1 - \lambda) * L_{robust}$$



Entrainement robuste dans Aidge

- Utilisation de la récriture de graph de Aidge pour calculer des intervalles



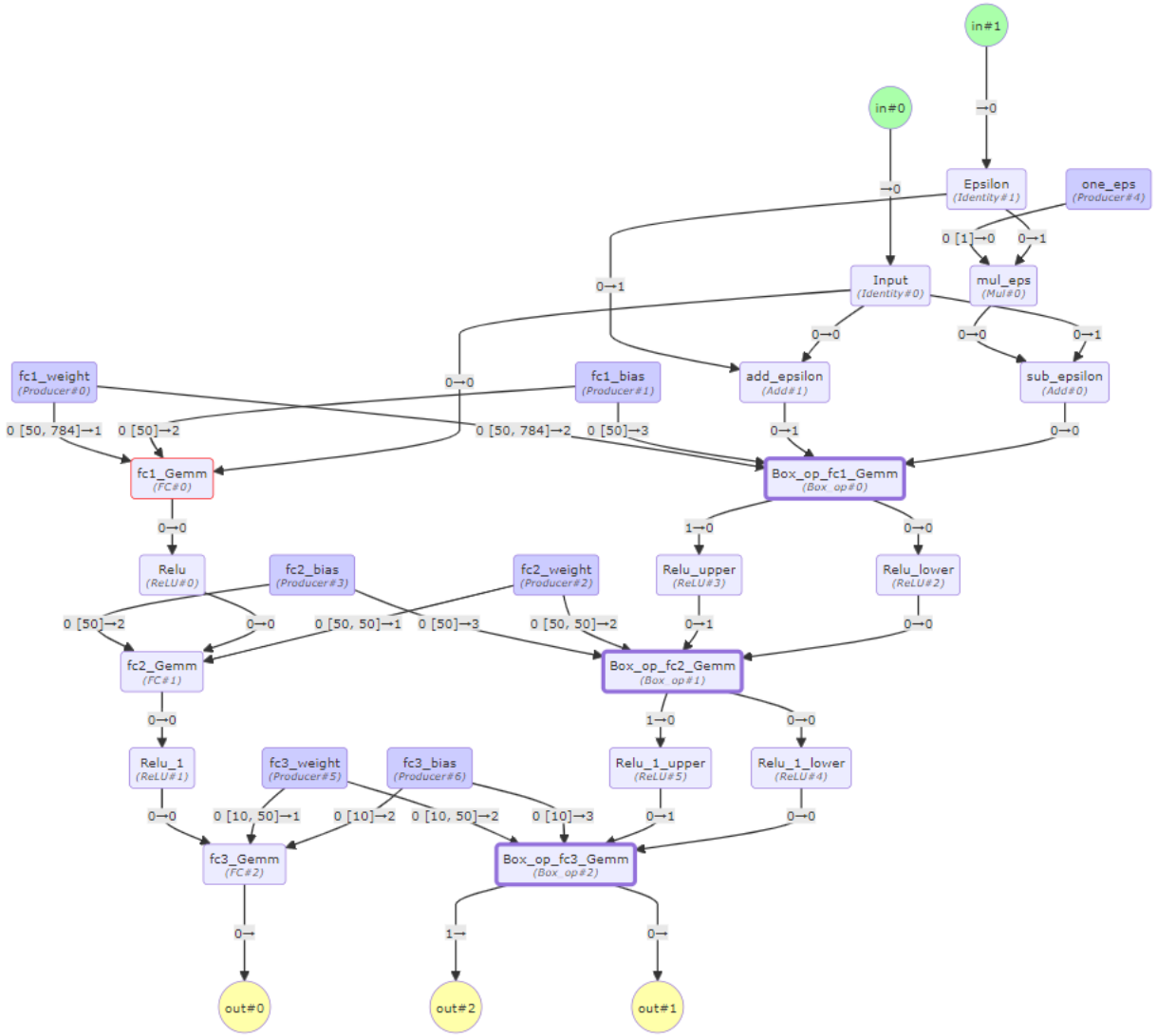
Entrainement robuste dans Aidge

- La réécriture ajoute des intervalles de autour de chaque sortie sur le modèle

```
Aidge prediction = 5
Output: 8.327168
Lower bound: 6.71582
Upper bound: 10.044268
```

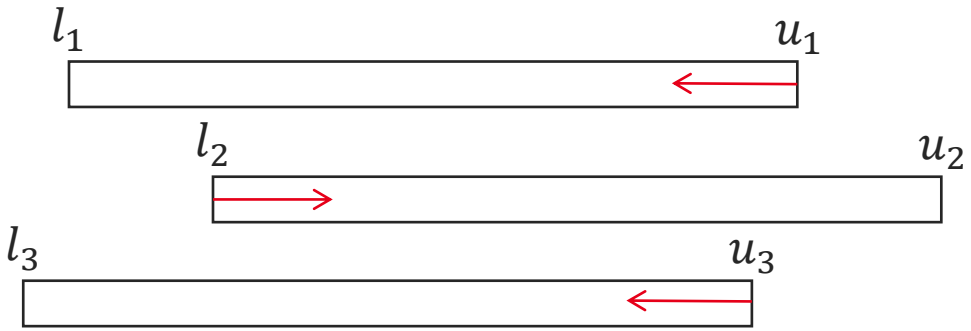
  - et peut être **exportée sur différentes cibles**

- Permet d'obtenir un gradient et une loss robuste

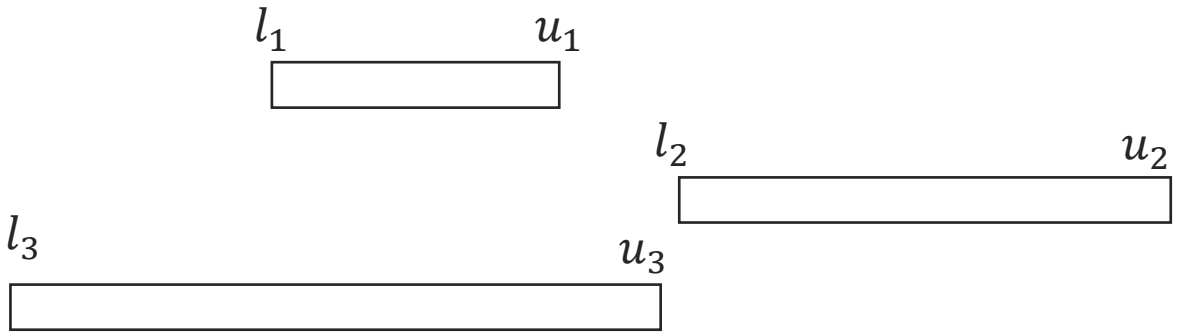


Loss robuste

Label de sortie : 2



Après l'entrainement



$$L_{robust} = CE(Y_{robust} = [u_1,\ l_2,\ u_3])$$

Points clefs de l’entrainement robuste

- Pas de **changement d’architecture** nécessaire, contrairement à des réseaux Lipschitz par exemple

- Trade-off robustesse / précision et temps à configurer
  - Dépend du cas d’usage
  - Généralement l’entrainement est plus long

- Modèles obtenus plus robustes mais aussi plus facilement vérifiés comme robuste

Mise en commun

- Sur le dataset MNIST

Réseau
3 couches
denses

Dataset MNIST

Entrainement robuste du modèle

avec $\epsilon = \frac{4}{256}$

Quantification avec Aidge

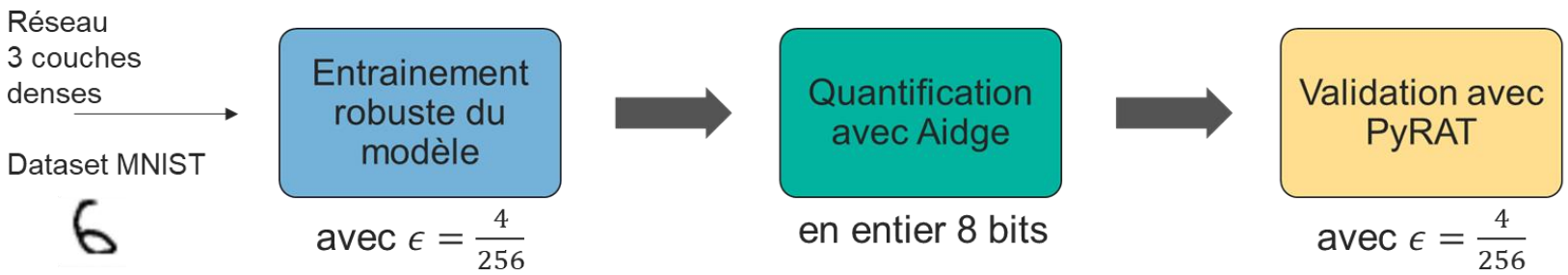
en entier 8 bits

Validation avec PyRAT

avec $\epsilon = \frac{4}{256}$

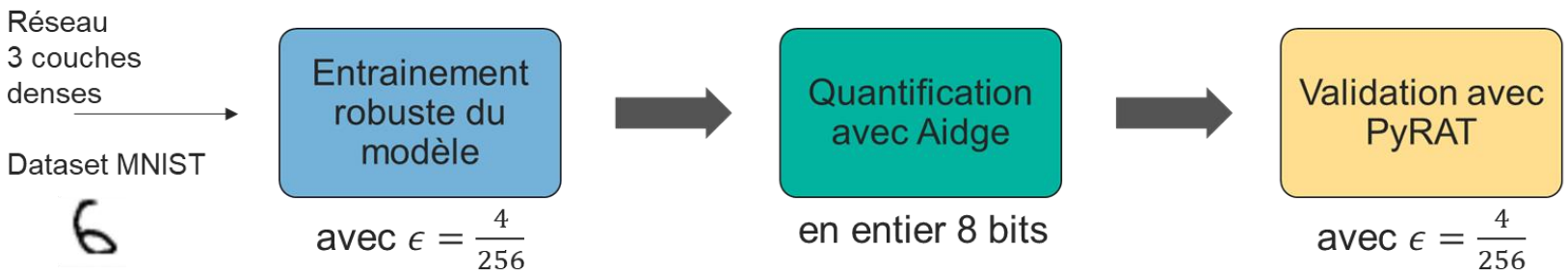
+ Vérification de non dépassement des bornes entières

Résultats



	Précision	Robustesse ($\epsilon = 4/256$)		
		Robuste	Non Robuste	Inconnu
Classique	**97,4%**	13,0%	23,7%	63,3%
Robuste	96,2%	**92,6%**	6,8%	0,6%

Résultats



	Précision	Robustesse ($\epsilon = 4/256$)		
		Robuste	Non Robuste	Inconnu
Classique	**97,4%**	13,0%	23,7%	63,3%
Robuste	96,2%	**92,6%**	6,8%	0,6%
Classique 8 bits	**97,5%**	0,0%	3,8%	96,2%
Robuste 8 bits	96,0%	**82,9%**	4,4%	12,7%

Conclusion et perspectives

- La quantification dans Aidge **préserve la robustesse** du modèle
- Aidge et PyRAT permettent d'obtenir **un modèle certifié robuste** et de petite taille (<= 8bits), facile à embarquer
- **Des bornes formelles sur les sorties** peuvent être obtenues sur cible avec le modèle

Prochaines étapes :

- Application à d'autres dataset / cas d'usages
  - Modèle robuste et quantifié sur l'ACAS-Xu
  - Suite des travaux « **A study of an ACAS-Xu exact implementation using ED-324/ARP6983"**
- Approfondir le lien avec l'implémentation sur cible et la specification
  - Groupe de travail Safety ONNX