Reconstruction de la position d'interaction de γ dans un scintillateur monolithique avec des réseaux de neurones

M.-A. Verdier, F. Bouvet, C. Trigila, A. Bricou, Y. Charon, M.-A. Duval, C. Esnault, L. Pinot, L. Ménard

Journée MI2B

verdier@imnc.in2p3.fr

Context

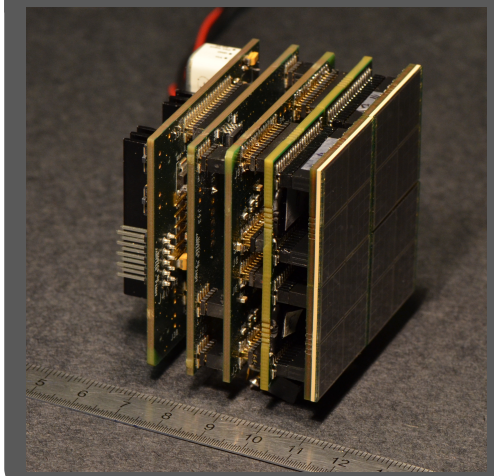
POCI/TRECAM, France

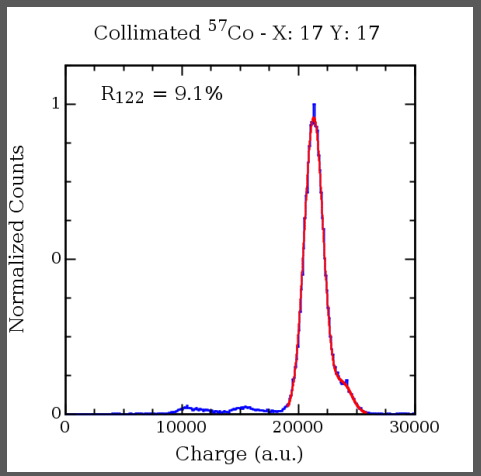
Sentinella, Spain

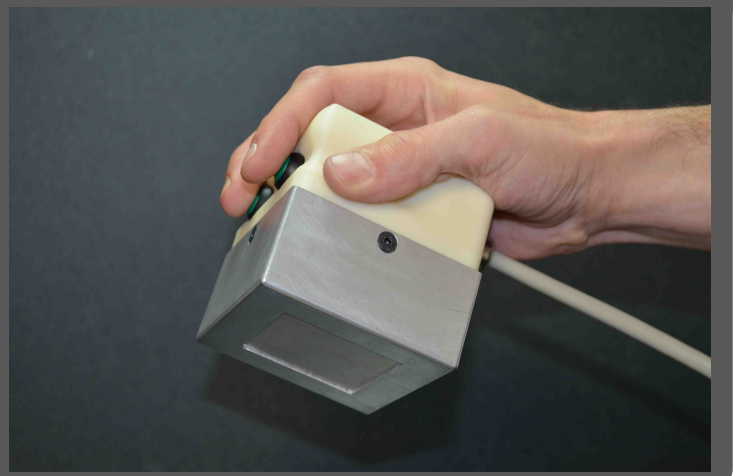
- Les γ-caméras compactes présente un intérêt en combinaison avec les sondes de comptage, pour la chirurgie radio-guidée (ex. SNOLL)
- Il y a un besoin de développement d'appareils compacts, utilisables au contact direct des tissus et avec une résolution spatiale, une sensibilité et une ergonomie améliorée.

γ-Camera @ IMNC

- γ-caméra légère (1.2 kg) compacte à hautes performances
  - Module de photodétection 256 pixels[1]
    - 4 MPPC 8×8 pixels. (SiPMs de 3 mm)
    - 8 ASICs EASIROC 32 voies avec DAC ajustables 8 bits
    - Carte mère FPGA + Alim.
    - Conso totale : 2.7 W
  - 5 × 5 cm^2 $LaBr_3$:Ce continu (5 ou 6 mm)
  - 15 mm Pb collimateur à trous parallèles
  - Hautes performances avec une résolution spatiale intrinsèque sous-millimétrique et 9% en résolution énergétique @122 keV



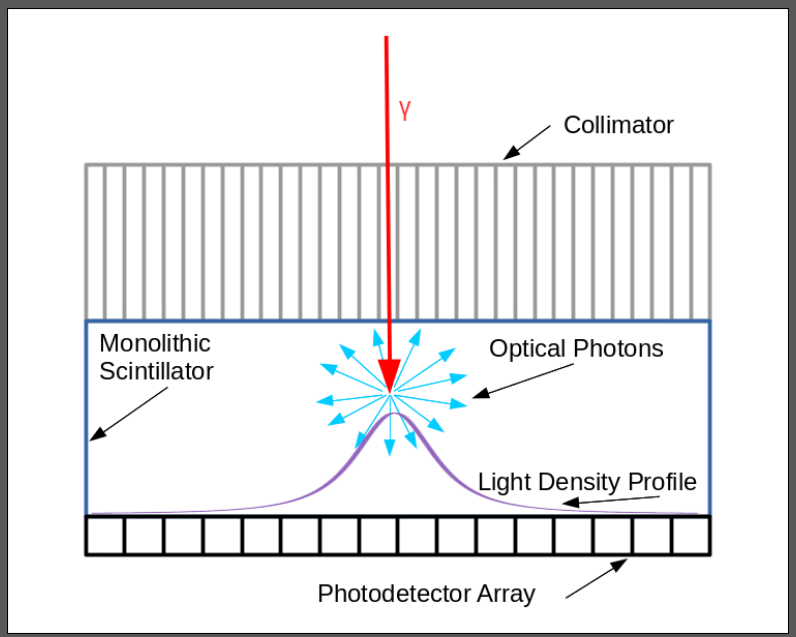




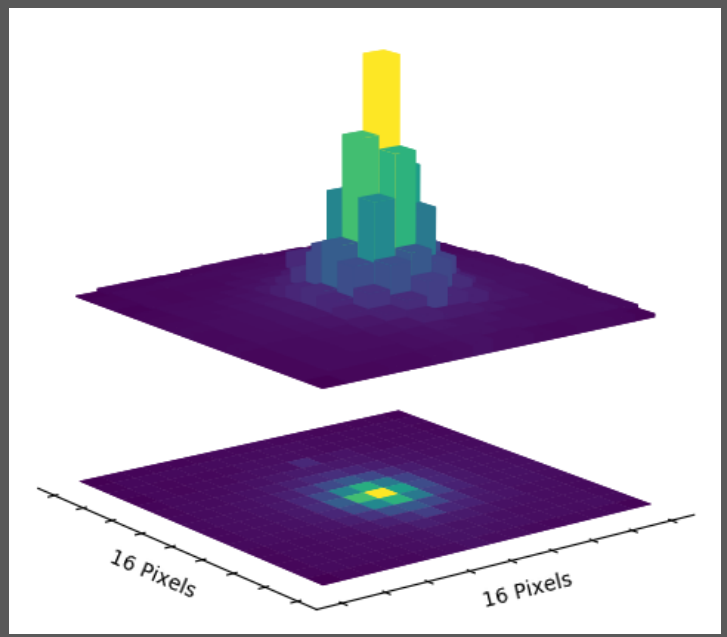
[1] N. Dinu *et al.* NIMA 2015

Caméra à Scintillateur Monolithique

schéma



Distribution de Lumière



- Émission de photons optiques isotrope au point d'interaction.
- Profile 2D de densité surfacique de lumière.

$\Rightarrow$ Reconstruction de la position d'interaction (x_0, y_0) des γ ?

Méthodes de Reconstruction

- Événement γ de paramètre $\theta = (x_0, y_0, z_0, N_0)$
- Chaque pixel p_i a la proba de produire N_k photoelectrons (Poisson) :

$$\begin{cases} \mathbb{P}_i(N_k|\theta) & = & \dfrac{\mu_i^{N_k}(\theta)e^{-\mu_i(\theta)}}{N_k!} \\ \mu_i(\theta) & \propto & N_0 \times \eta_i(x_0, y_0, z_0) \end{cases}$$

- $\eta_i(x_0, y_0, z_0)$: Fonction de réponse à la lumière : inconnue

Type	Méthode	Vitesse	Précision	Note
Centroïde	Barycentre	✓✓	××	Forte déformation
	Barycentre carré	✓✓	×	Forte déformation
Ajustement	Modèle Analytique	✓	✓	Modélisation difficile
	Interpolation	××	✓✓	Calibration nécessaire
Machine learning	KNN	××	✓×	Calibration nécessaire
	Réseau de neurones	✓✓	??	Entraînement nécessaire

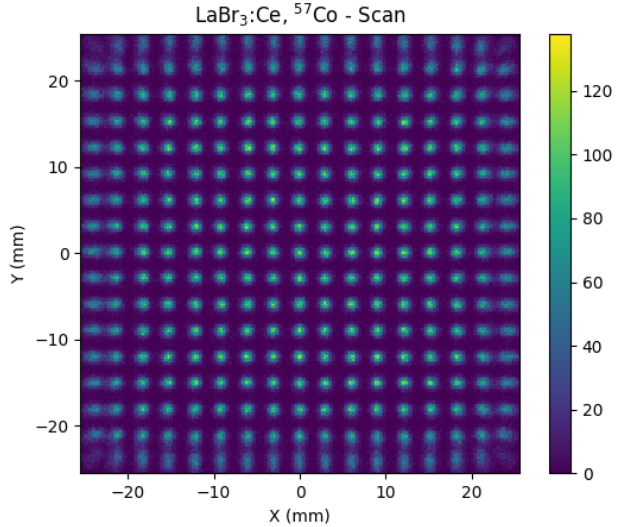
Ajustement itératif d'un modèle

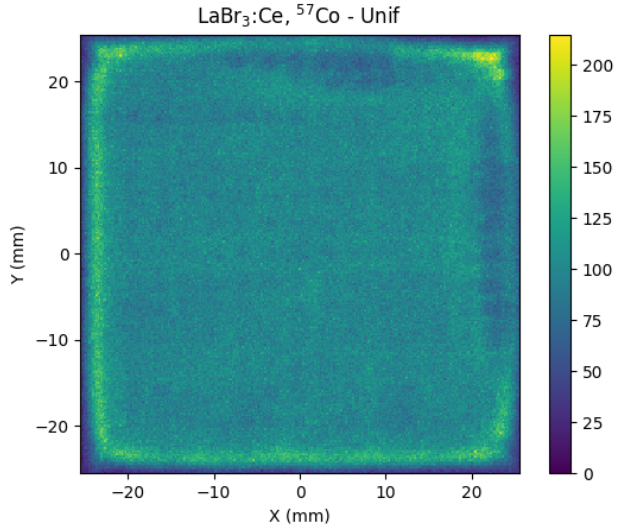
Modèle Scrimger-Baker

$$\eta_i(x_0, y_0, z_0) = \frac{1}{2\pi z_0^2 \left(1 + \frac{r^2}{z_0^2}\right)^{\frac{3}{2}}}$$

$$r = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$

- $\sim$ 1000 evt/s
- Réso. spatiale : 0.91 ± 0.06 mm
- Biais non lin. : 0.06 ± 0.04 mm



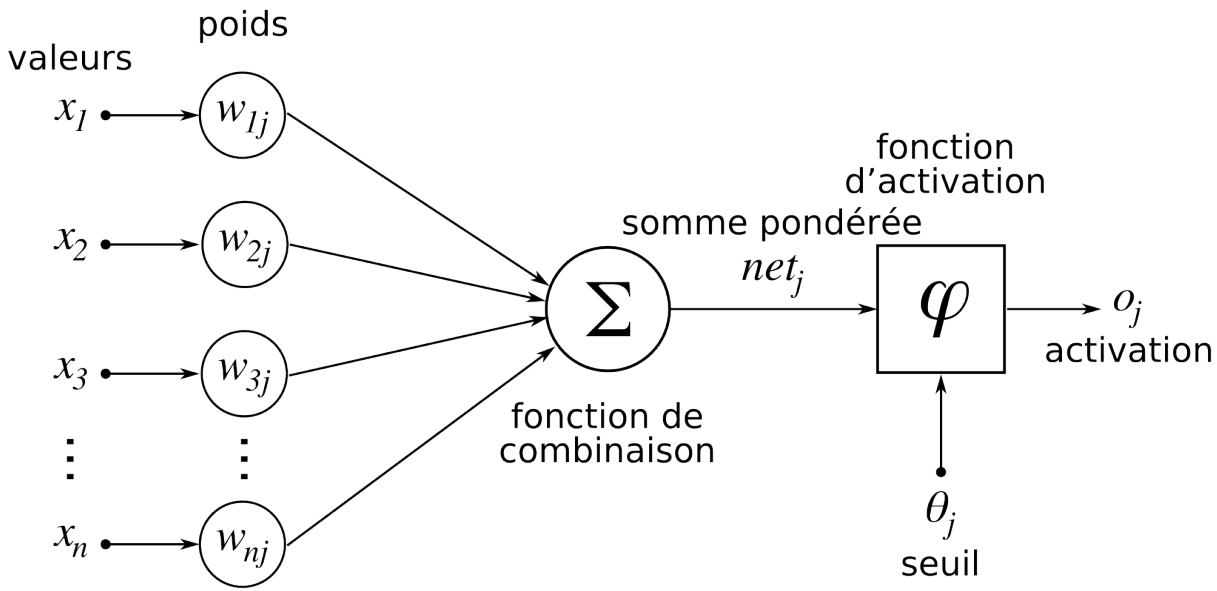


⇒ Bon compromis vitesse/perfo

⇒ Avec un RN capable de déterminer une position d'interaction quelconque (x_0, y_0) à partir jeu d'entraînement (scan) ?

Fonctionnement d'un neurone

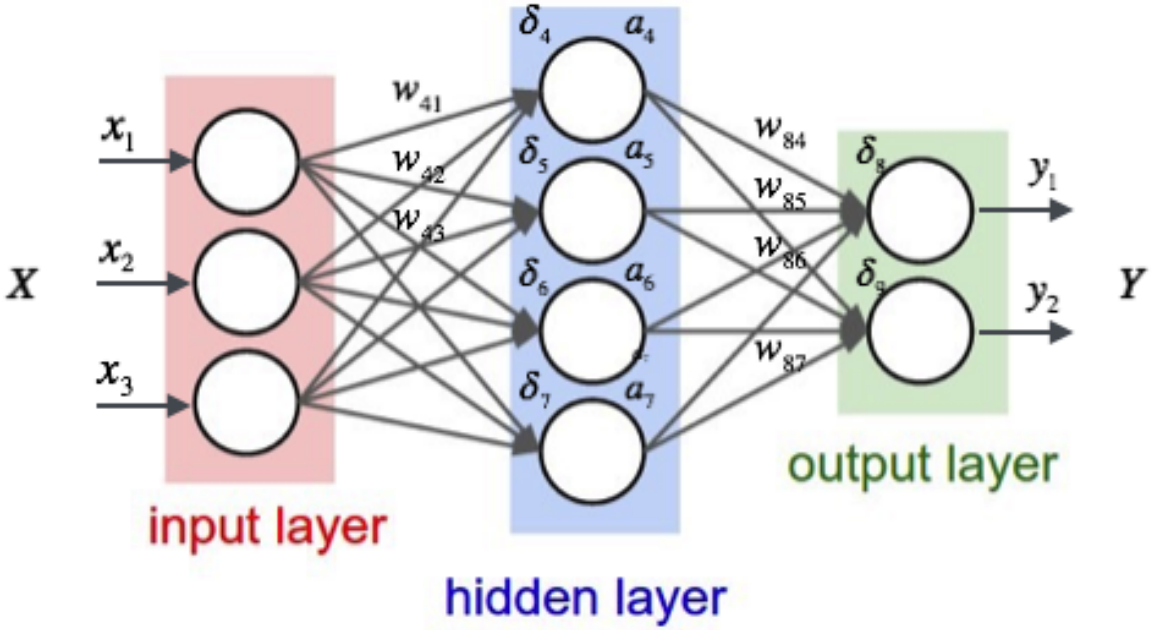
- Neurone j
- N entrées x_i (tous les neurones de la couche précédente)
- N poids w_{ij}
- Une fonction d'activation φ
- Un seuil θ_j



$\Rightarrow$ Output : $o_j = \varphi(\sum_i x_i w_{ij} + \theta_j)$ ou encore $o_j = \varphi(\mathbf{x}_j \cdot \mathbf{w}_j) + \theta_j$

Réseau *Fully Connected*

- Réseau constitué de plusieurs couches de neurones
- Chaque neurone est connecté avec tous les neurones des couches antérieurs et postérieurs
- Chaque couche possède sa fonction d'activation

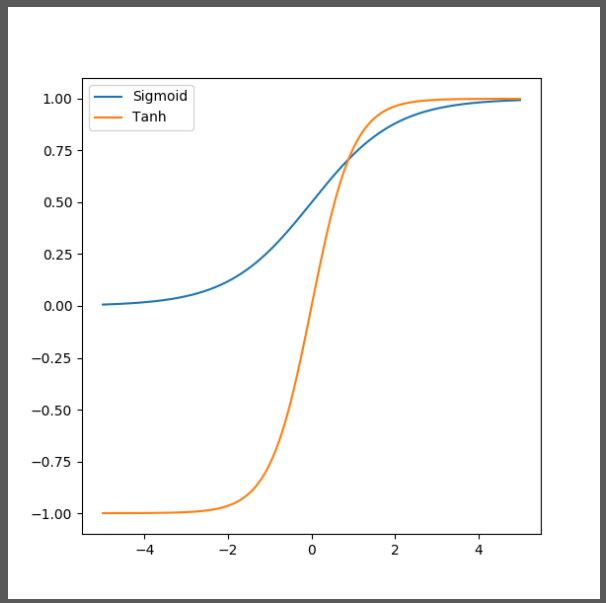


$\Rightarrow$ Pour un réseau de 3 couches (3 - 4 - 2), la valeur de la sortie $y_1 = \varphi_8(\delta_8 + \sum_{i=4}^{7} a_i w_{8i})$ et où les $a_i = \varphi_i(\delta_i + \sum_{j=1}^{3} x_j w_{ij})$

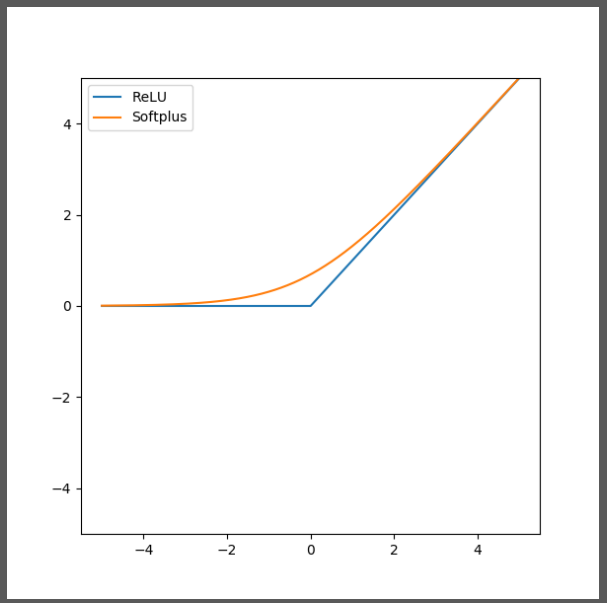
Fonction d'Activation

- Différents type de fonctions d'activation
  - *Sigmoid* : $f(x) = \frac{1}{1+e^{-x}}$
  - *Tanh* : $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
  - *Linear* (identité) : $f(x) = x$
  - *Relu* (Rectified Linear Unit) : $f(x) = max(0, x)$
  - *Softplus* : $f(x) = log(1 + e^{x})$

Sigmoid, Tanh



Relu, Softplus



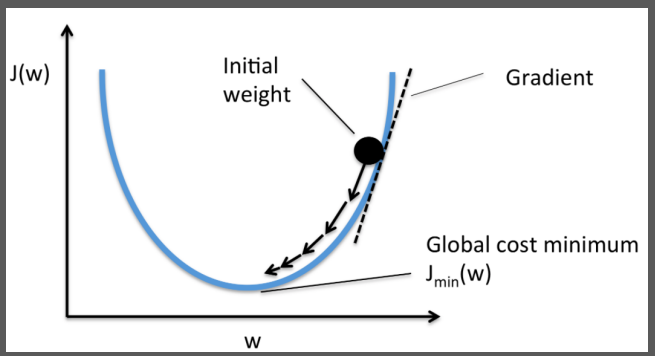
Déterminer les poids **w**

- Jeu de données d'entraînement **X** avec résultats **y**
- Fonction de calcul d'erreur (pertes)
  - MSE (Mean Squared Error) : $L = \frac{1}{N}\sum_i (\hat{y}_i - y_i)^2$
  - *etc*

- Rétropropagation
  ⇒ Pénaliser les poids suivant leur influence
  ⇒ Fonction d'optimisation
  - Stochastic Gradient Descent
  - Adam[2] (et ses variations : Adamax, Adadelta, *etc*)

Descente de gradient

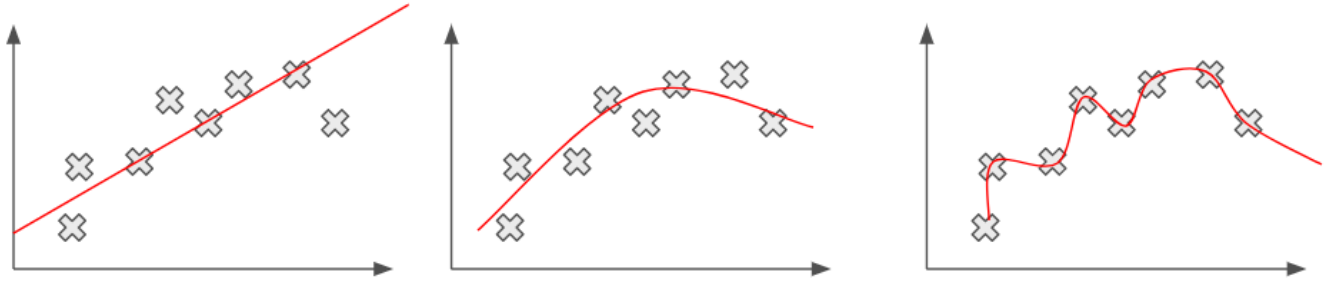
$$\Delta w_{ij} = -\eta \frac{\partial L}{\partial w_{ij}}$$



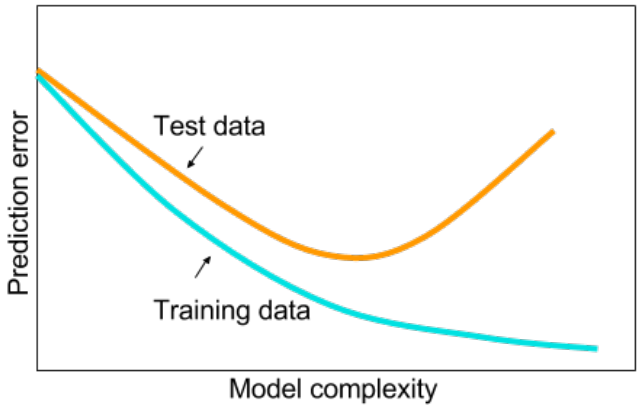
⇒ Itératif : Entraînement sur plusieurs *époques*

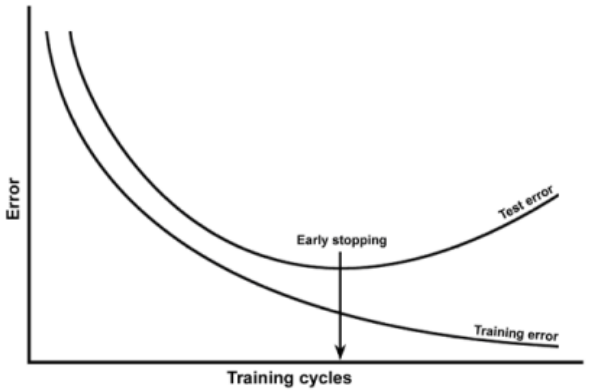
[2] https://arxiv.org/abs/1412.6980v8

Structure du RN



- Déterminer la bonne structure du RN et le nombre d'époques
  - Sur-apprentissage : Réseau trop complexe ou entraîné
  - Sous-apprentissage : Réseau pas assez complexe ou entraîné.
- *Regulizers* pour limiter le sur apprentissage : L1, L2, Dropout





⇒ Pertes sur un jeu de données de validation

Optimisation du RN

- Structure
  - Nombre de couche cachées
  - Nombre de neurones par couche
- Quelle fonctions d'activations pour notre cas
  - Couches cachées : ??
  - Sortie : linéaire (régression)
- Quelle Fonction de pertes : *MSE*
- Quel algo d'optimisation
- Quels régularisateurs
- Combien d'époques

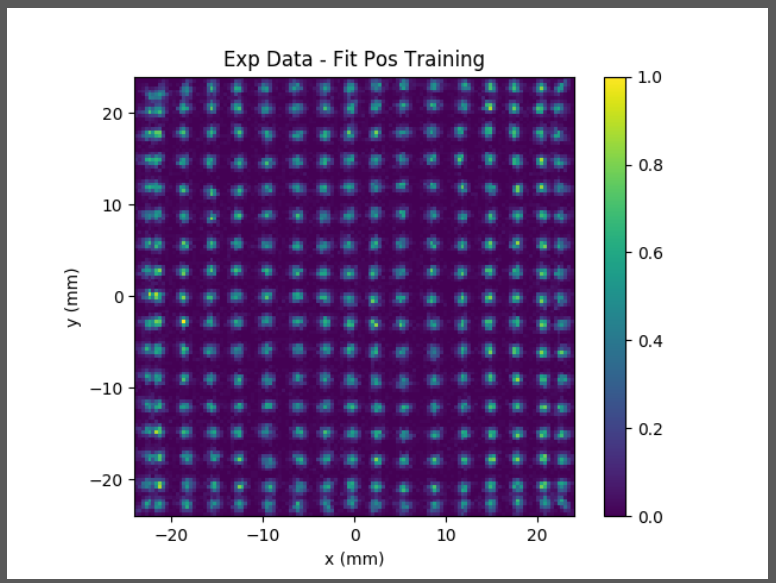
$\Rightarrow$ Énormément de paramètres à ajuster !

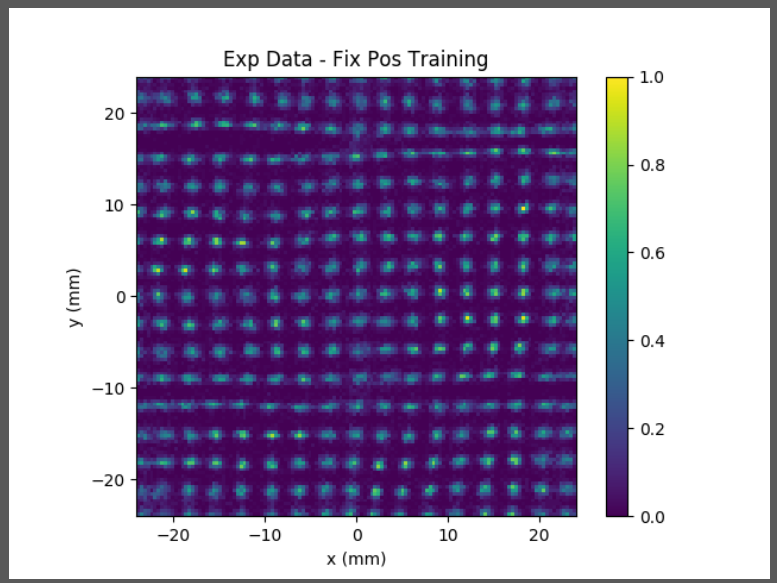
$\Rightarrow$ Quelles données d'entraînement ?

Quelles données pour l'Optimisation ?

- Scan de ^{57}Co avec collimation ϕ=500 m et un pas de 3 mm.
- Positions réelles inconnues.
  - Ajustement initial des données par une autre méthode ?
  - Approximation par la position mécanique du collimateur ?

Données expérimentales





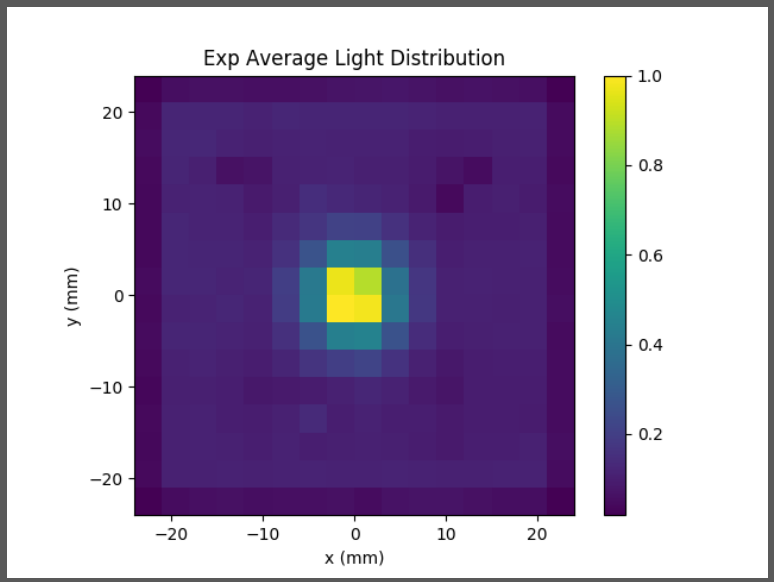
⇒ Forte influence de l'imprécision des données d'entraînement !

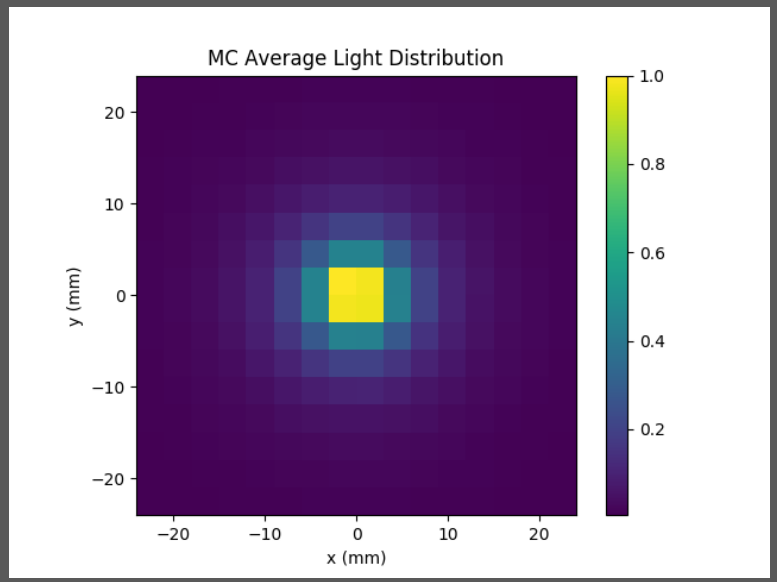
⇒ Simulation MC pour optimiser le RN (pas l'entraîner).

Données Monte Carlo (1)

- Génération d'événements de scintillation (trames) dont la position (x_0, y_0, z_0) ainsi que le nombre N_0 de photons généré est connu
  - Position d'interaction (x_0, y_0, z_0) est connue.
  - Nombre de photons généré N_0 de photoélectrons sont connus.

Trame moyenne au centre du scintillateur



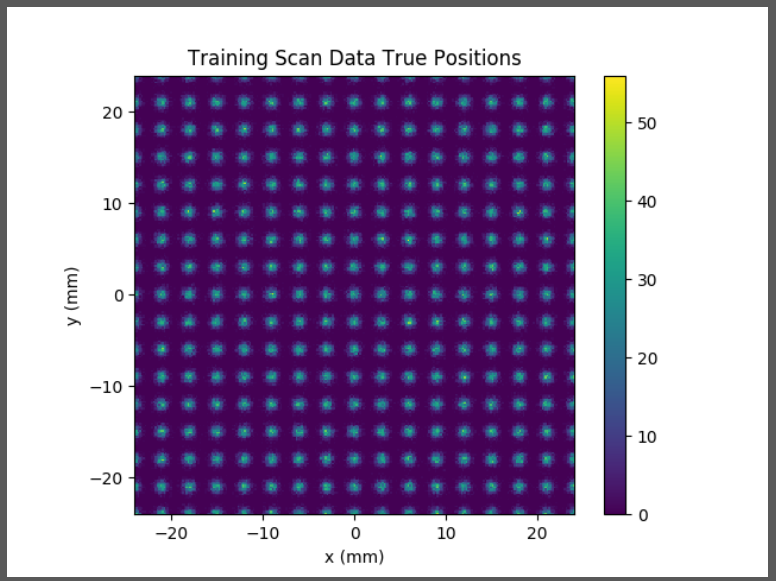


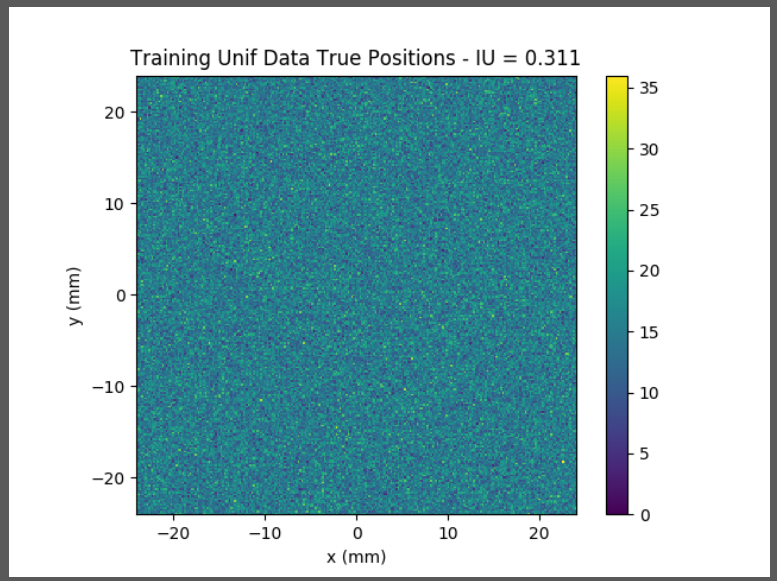
⇒ Jeux de données MC

Données Monte Carlo (2)

- Création de jeux de données MC
  - 2 *Scans* de (17×17) positions de 1000 événements.
    $\Rightarrow$ Entraînement + validation
  - 1 *Unifs* d'1 million d'événements. $\Rightarrow$ Validation

Scan et *Unif* simulés



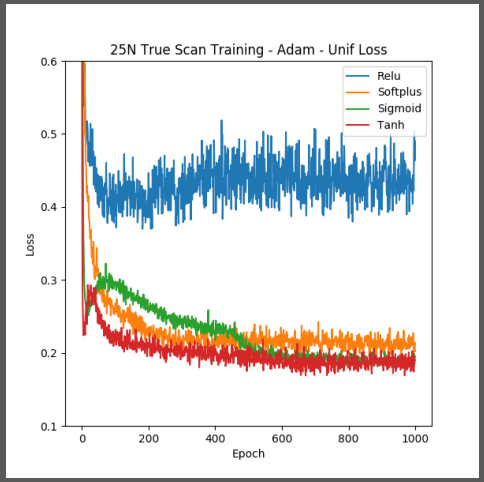


$\Rightarrow$ Paramétrisation du RN.

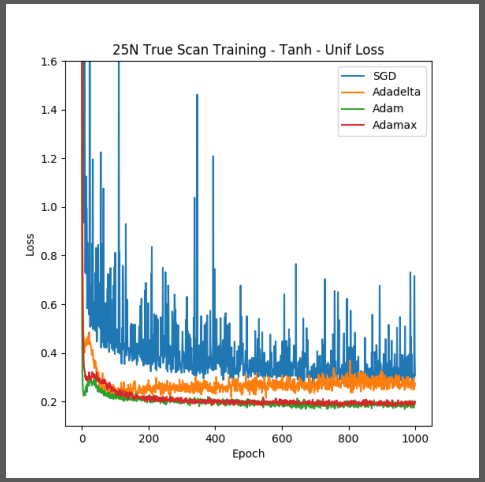
$\Rightarrow$ Entraînement fait avec le *Scan*, le calcul d'erreur, sur l'*Unif*.

Fonction d'activation

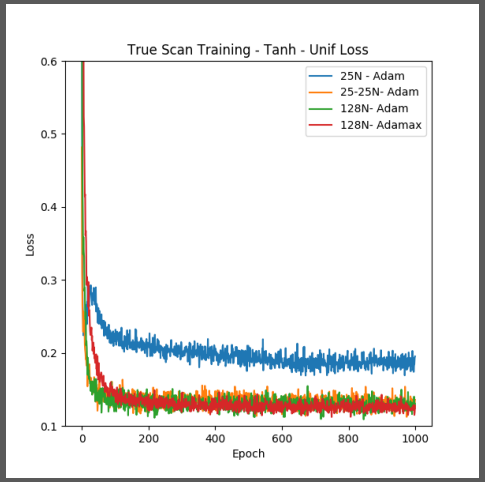
Fonction d'activation



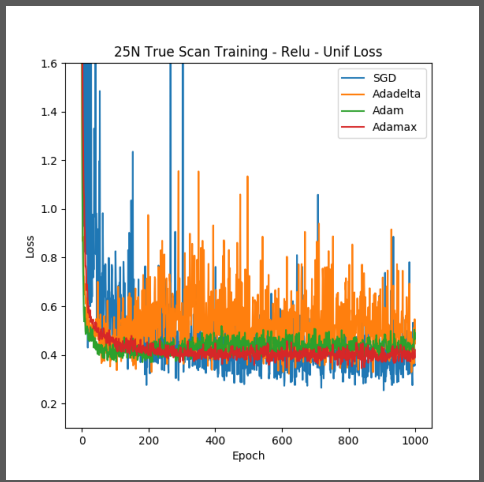
Fonction d'activation



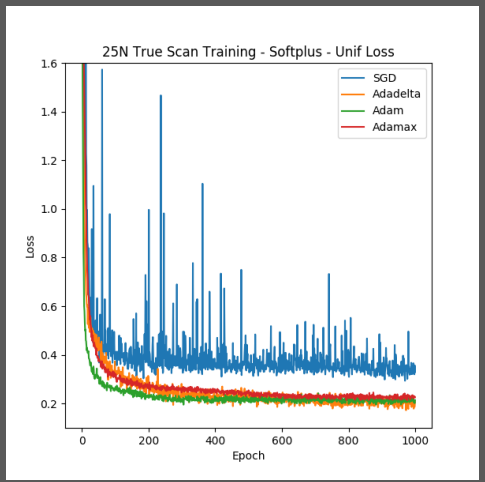
Fonction d'activation



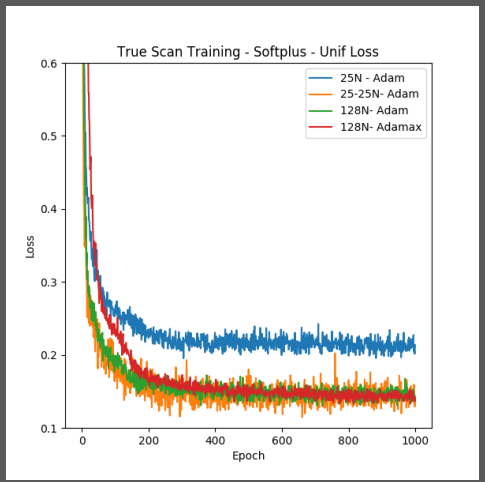
Fonction d'activation



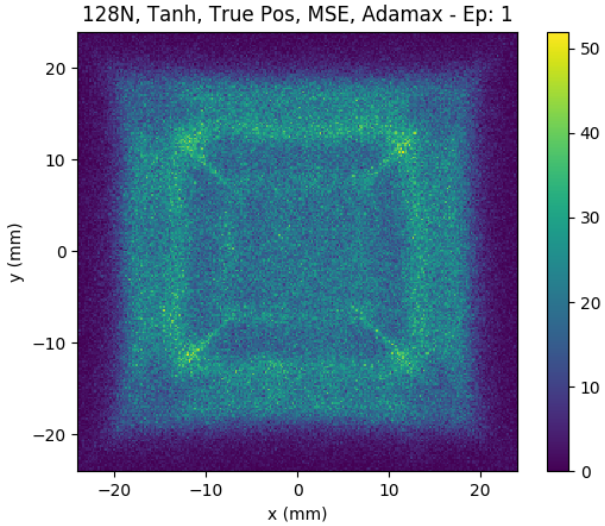
Fonction d'activation

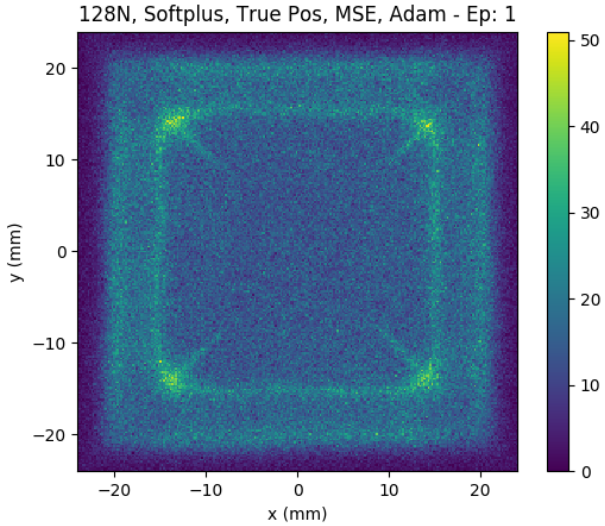


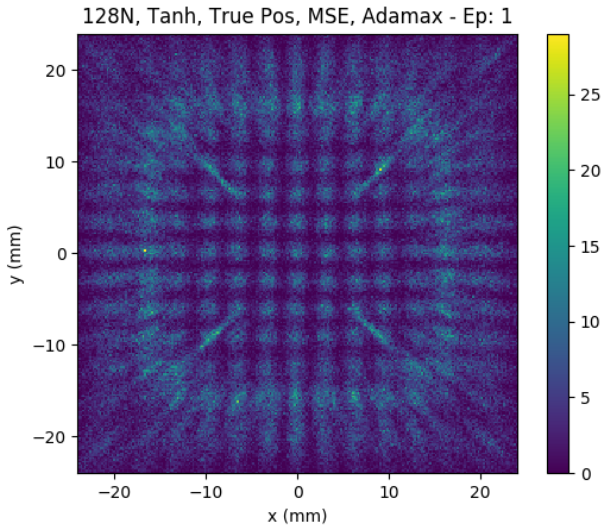
Fonction d'activation

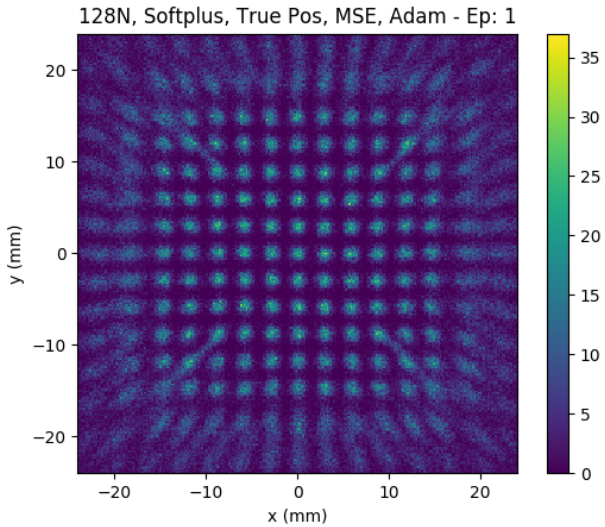


128N+(*Tanh*+*Adamax*) ou (*Softplus*+*Adam*)



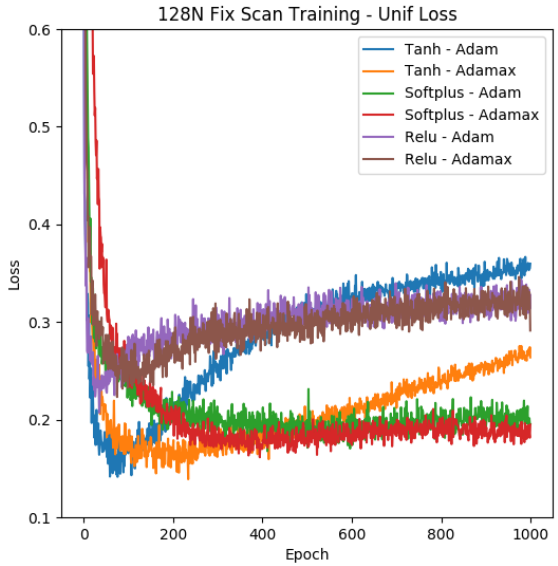


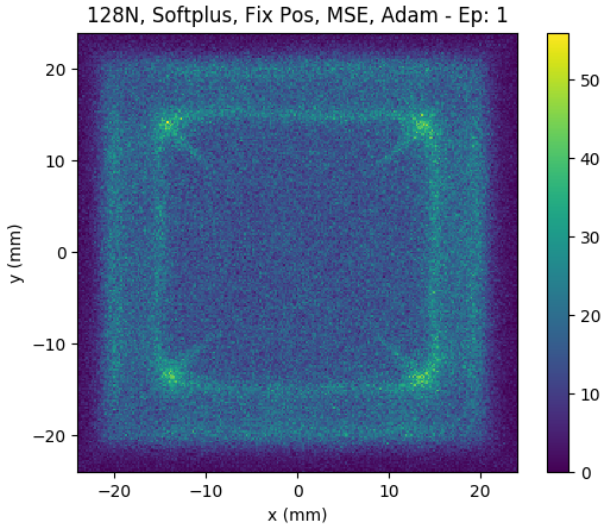


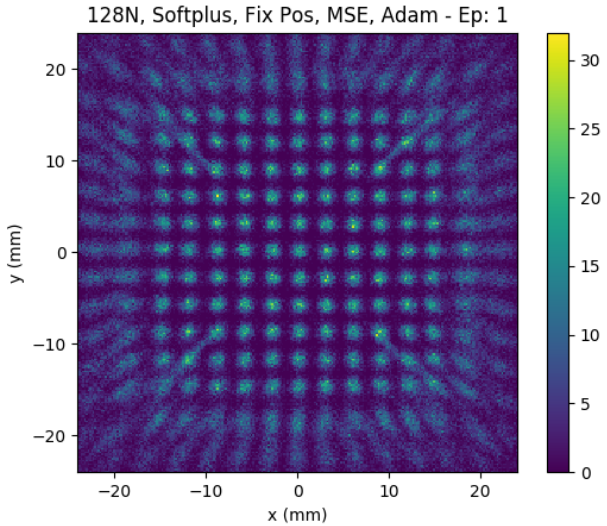


Entraînement avec positions mécaniques

- Réseau idéal ?
  - 128 Neurones
  - *Softplus*
  - *Adam*





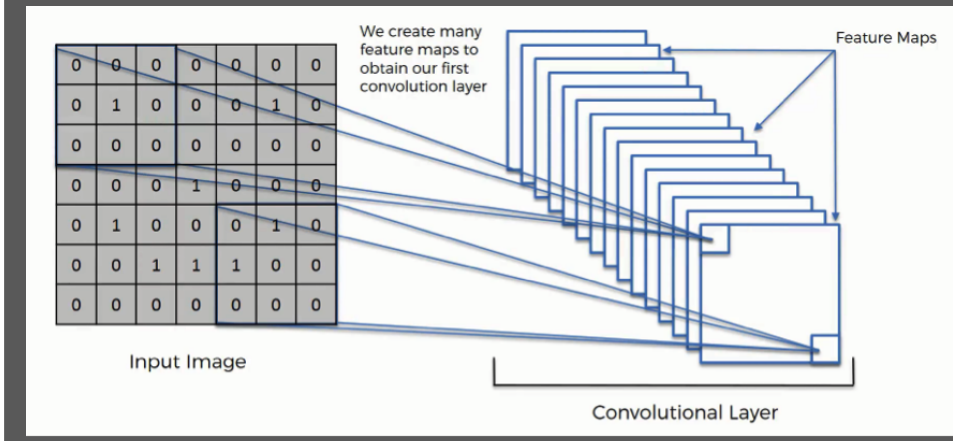


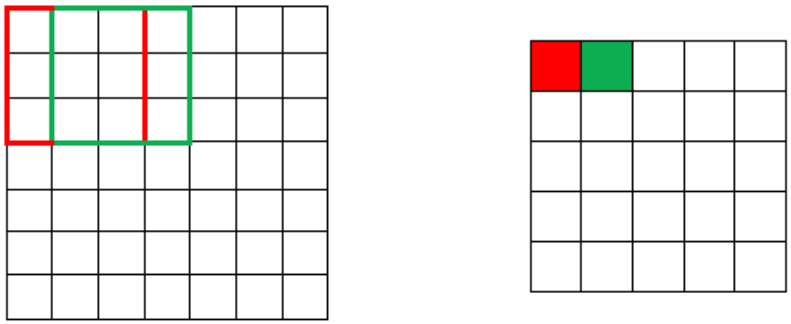
Méthode alternative : RN Convolué (2D)

Tire partie du fait que chaque trame est une image 2D

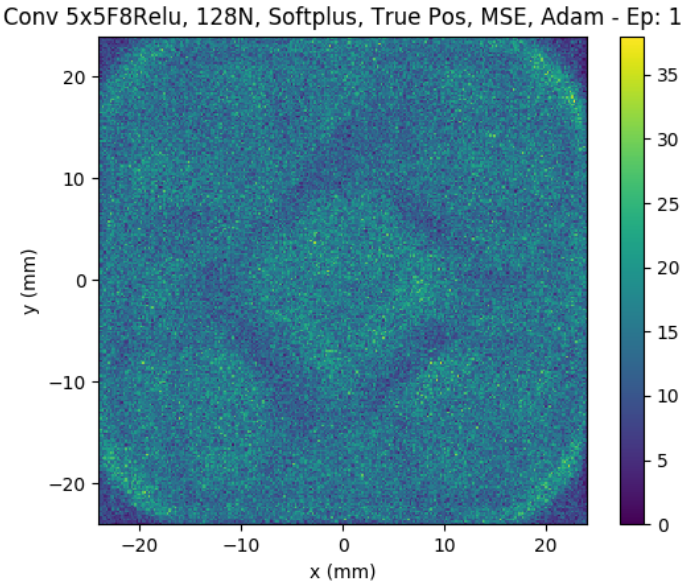
- Kernel de 5x5 neurones et foulée (stride) de 1
  $\Rightarrow$ 144 Sous-réseaux
- Couche cachée :
  N filtres de 144 kernels en parallèle

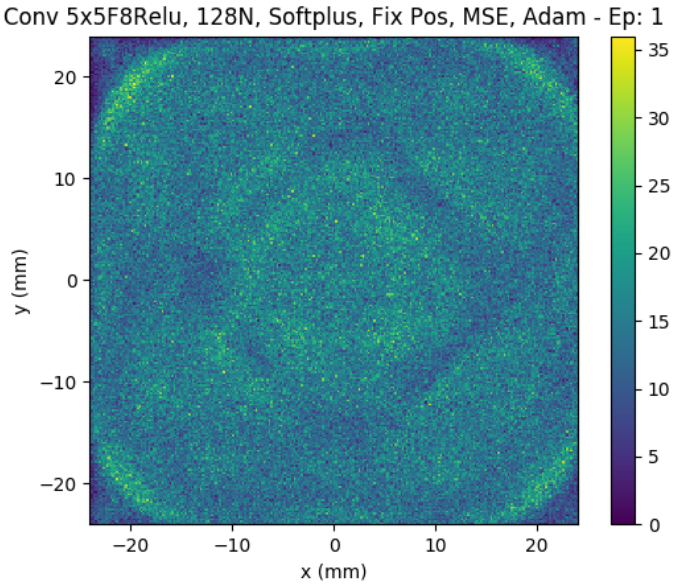
Convolution 2D





Résultats RN Convolué

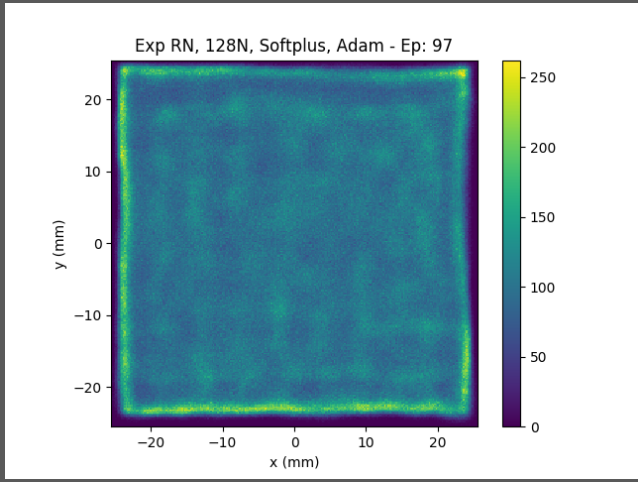




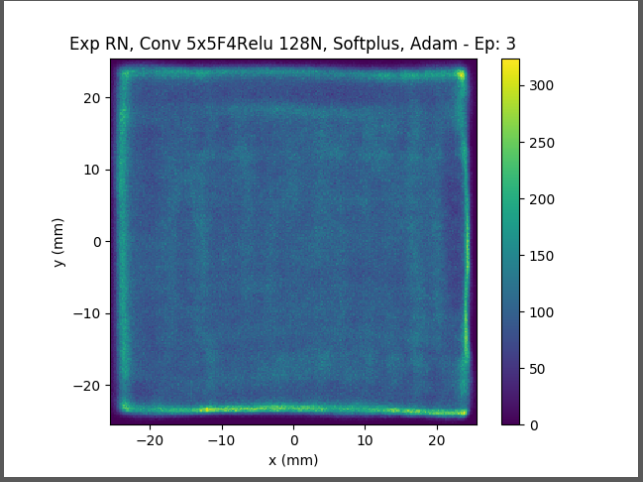
⇒ Très prometteur mais nécessite des réglages fins.

Données Expérimentales

Réseau *Fully Connected*



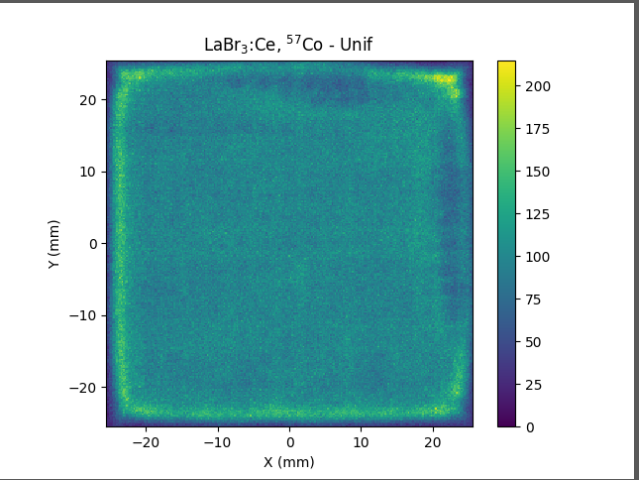
Réseau *Convolué*



⇒ Déformations supérieures à la méthode itérative mais gain en vitesse d'un ordre de grandeur.

⇒ Nécessite encore des ajustement.

Méthode itérative



Conclusion

- Optimisation du RN "fully connected"
  - Fonction d'activation : *Tanh* et *Softplus*
  - Optimisateurs : *Adam* et *Adamax*
  - Amélioration avec le nombre de neurones (au moins 128)
  - 1 seule couche cachée semble suffire
- RN convolué (2D) pour tirer partie de l'image
  - Converge vite mais très sensible au sur-apprentissage

- Perspectives (à faire)
  - Méthode de quantification
  - Ajustement fin des paramètres (Régulateurs notamment)
  - Ajuster les données en amon par une méthode précise pour l'entraînement.
  - Ajouter la profondeur d'interaction.
  - Passage sur GPU ?
  - Autres idées ...