DE LA RECHERCHE À L'INDUSTRIE

Formation EJN – IA
Incertitudes et robustesse

Geoffrey DANIEL CEA/DES/ISAS/DM2S/STMF/LGLS

Incertitudes en deep learning

1. Différents types d'incertitude
2. Estimation de ces incertitudes en Deep Learning
3. Validation des incertitudes

Travaux effectués en collaboration avec Jean-Marc MARTINEZ (DES/ISAS/DM2S/STMF/LGLS)
Ainsi que deux stagiaires : Mohamed Bahi YAHIAOUI (Mines St Etienne) et Clément RIBES (IMT Atlantique)

Incertitudes en deep learning

1. **Différents types d'incertitude**
2. Estimation de ces incertitudes en Deep Learning
3. Validation des incertitudes

Travaux effectués en collaboration avec Jean-Marc MARTINEZ (DES/ISAS/DM2S/STMF/LGLS)
Ainsi que deux stagiaires : Mohamed Bahi YAHIAOUI (Mines St Etienne) et Clément RIBES (IMT Atlantique)

Incertitudes de prédiction, aléatoire et épistémique

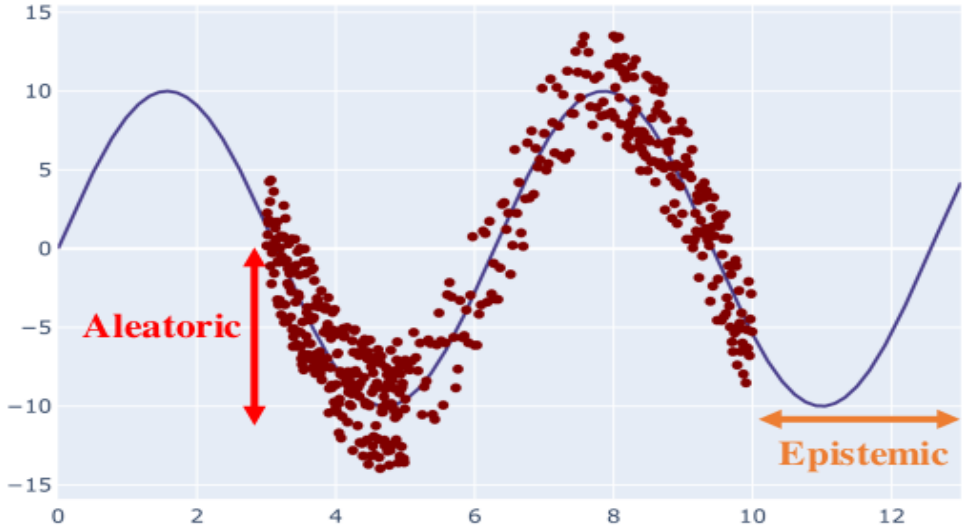
Soit x une donnée d'entrée, f_ω un modèle prédictif de paramètres ω

Objectif : Quantifier l'incertitude sur la prédiction $f_\omega(x)$

→ **Incertitude de prédiction**

Incertitude aléatoire (aleatoric uncertainty)
→ Incertitude liée aux données et au phénomène

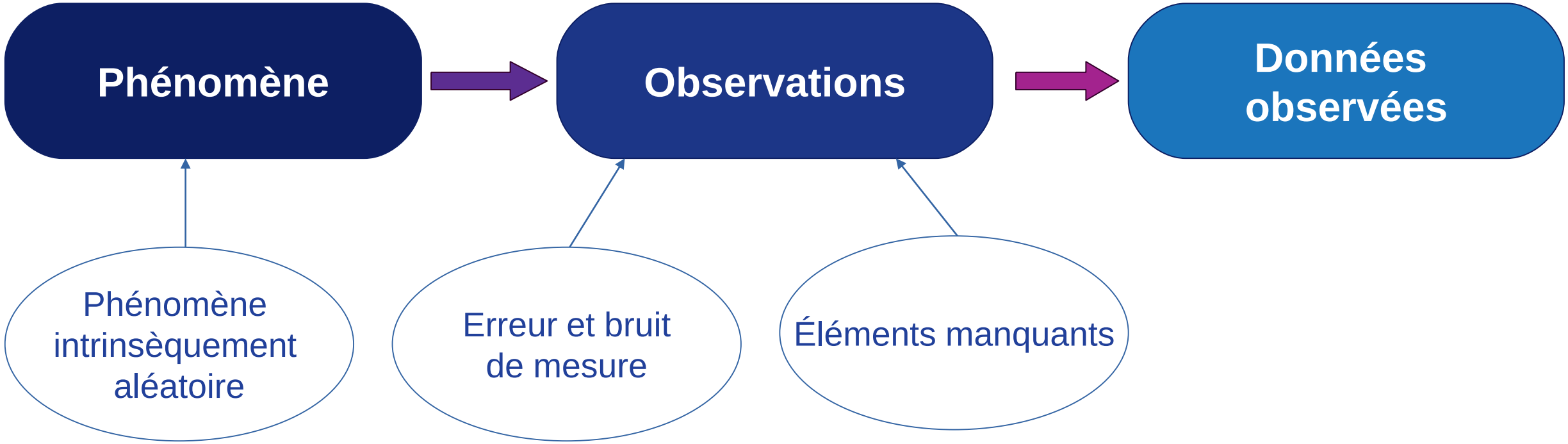
Incertitude épistémique (epistemic uncertainty)
→ Incertitude liée au modèle



Visualisation des deux types d'incertitude

A review of uncertainty quantification in deep learning: Techniques, applications and challenges, M. Abdar et al., https://doi.org/10.1016/j.inffus.2021.05.008

Incertitude aléatoire



Incertitude intrinsèque aux données, irréductible → **Nombre plus important de données ne réduit pas cette incertitude mais permet de mieux l'estimer !**

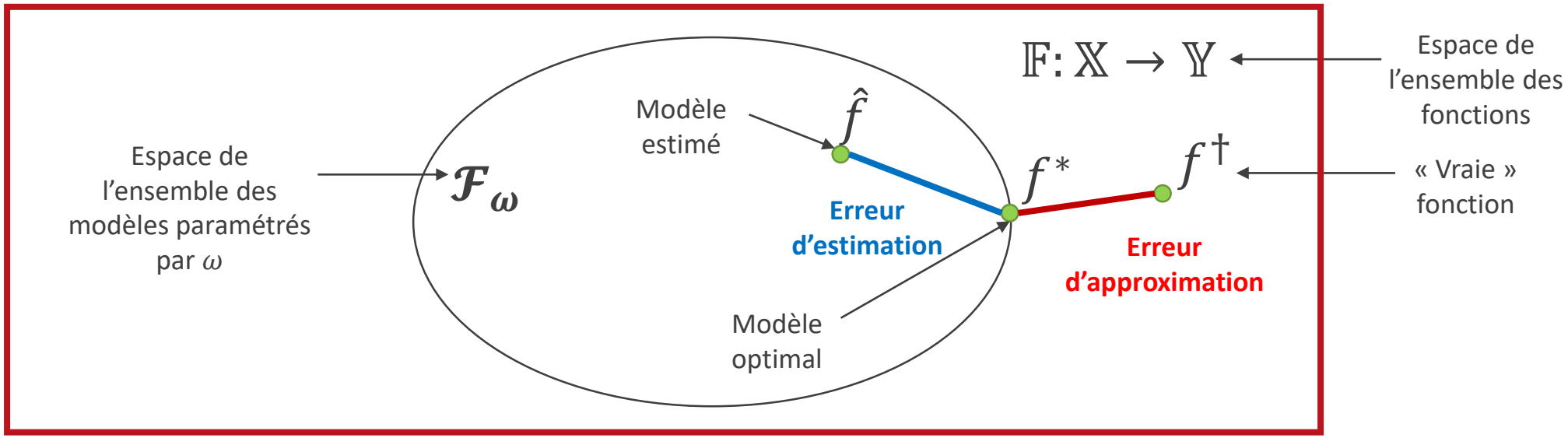
On peut la réduire en améliorant la qualité des données si le problème provient de l'erreur ou du bruit de mesure, mais pas si elle provient du caractère aléatoire du phénomène

Incertitude épistémique

Représente le manque de « connaissance » ou de « compréhension » d'un modèle pour une donnée d'entrée spécifique

Origines de l'incertitude épistémique pour un algorithme de machine learning :

- **Erreur d'estimation** : les données d'entraînement ne sont qu'un échantillon de l'ensemble des données possibles
- **Erreur d'approximation** : le modèle ne peut pas représenter la « vraie » fonction



Incertitude réductible avec un nombre plus important de données et un modèle suffisamment complexe

Modélisation probabiliste

x : une donnée d'entrée
ω : paramètres du modèle
y^* : une sortie possible
D : le jeu de données d'apprentissage

Incertitude de prédiction totale modélisée par une loi de probabilité

$$\boldsymbol{p(y^*|x, D) = \int_{\omega} p(y^*|x, \omega) p(\omega|D) d\omega}$$

Part aléatoire **Part épistémique**

En pratique : incalculable analytiquement → Quelles méthodes en Deep Learning ?

Incertitudes en deep learning

1. Différents types d'incertitude
2. **Estimation de ces incertitudes en Deep Learning**
3. Validation des incertitudes

Travaux effectués en collaboration avec Jean-Marc MARTINEZ (DES/ISAS/DM2S/STMF/LGLS)
Ainsi que deux stagiaires : Mohamed Bahi YAHIAOUI (Mines St Etienne) et Clément RIBES (IMT Atlantique)

Problème de classification (rappel)

Prédire à quelle classe une donnée x appartient parmi c classes, représentées par un vecteur binaire $\left(y^{(i)}\right)_{i=1}^{c}$

Représentation classique : un modèle f_ω prédit pour chaque classe i la probabilité d'appartenir à cette classe :

$$\boldsymbol{f_\omega^{(i)}(x) = p\left(y^{(i)} = 1 \middle| x\right)}$$

Vraisemblance associée : loi multinoulli (Bernoulli généralisée)

$$L(\omega|x) = \prod_{i=1}^{c} p\left(y^{(i)}|x;\omega\right)^{y^{(i)}} = \prod_{i=1}^{c} f_\omega^{(i)}(x)^{y^{(i)}}$$

Négative log-vraisemblance moyenne sur l'ensemble de la base de donnée : **categorical cross-entropy**

$$l(\omega) = -\frac{1}{N}\sum_j \sum_i y_j^{(i)} \log\left(f_\omega^{(i)}\left(x_j\right)\right) \rightarrow \text{Apprentissage : } \omega^* = \underset{\omega}{\operatorname{argmin}}\, l(\omega)$$

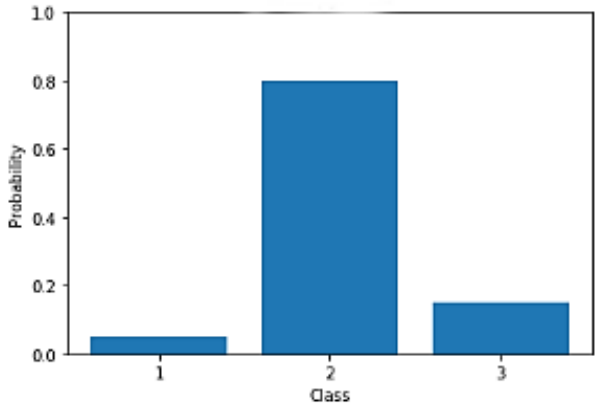
Problème de classification : incertitude aléatoire

Soit f_ω, un modèle donné de paramètres ω, x une donnée d'entrée

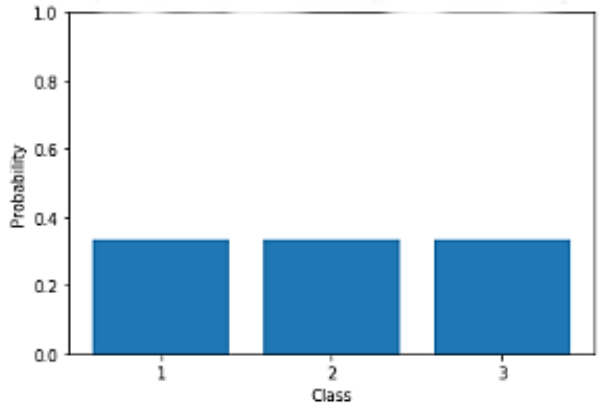
L'incertitude aléatoire modélise l'incertitude de décision entre chaque classe.

Exemple à 3 classes :

Faible incertitude



Forte incertitude
(complètement aléatoire)



Quantité pour résumer cette information ?

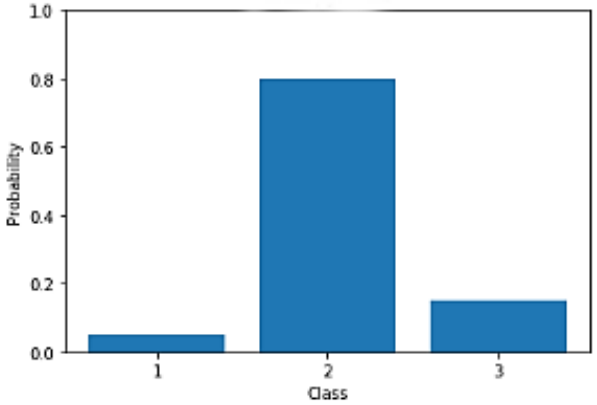
Problème de classification : incertitude aléatoire

Quantification par la théorie de l'information : **entropie de Shannon**

$$\mathcal{H}\left(p(y|x;\omega)\right) = -\sum_i p\left(y^{(i)}|x;\omega\right)\log\left(p\left(y^{(i)}|x;\omega\right)\right) = -\sum_i f_\omega^{(i)}(x)\log\left(f_\omega^{(i)}(x)\right)$$

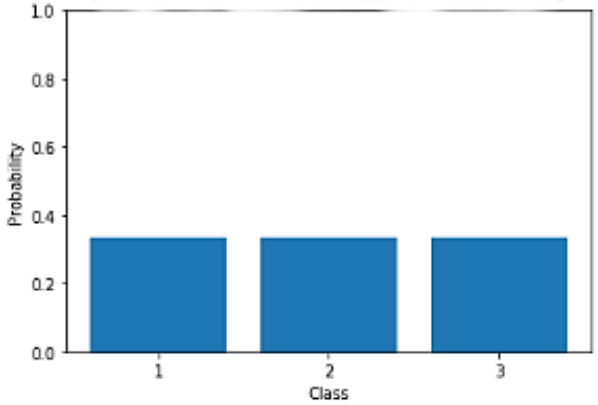
Exemple à 3 classes :

Faible incertitude



Entropie de Shannon : $0{,}61$

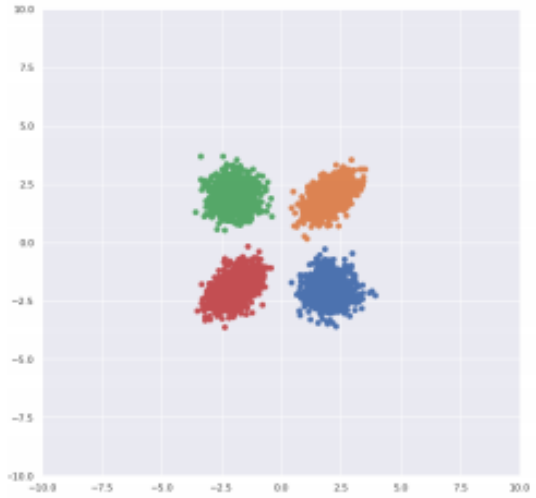
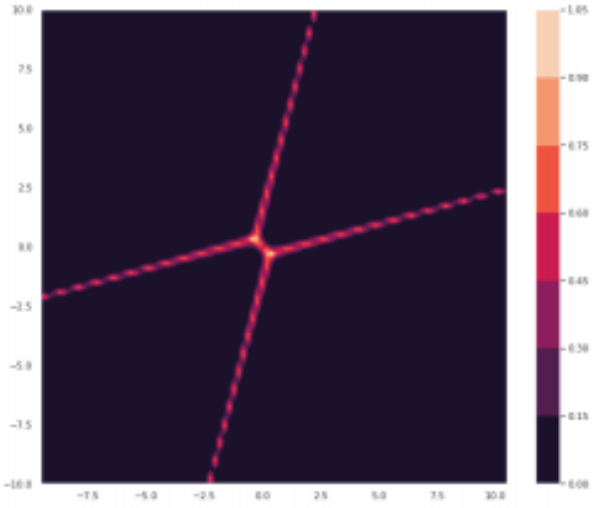
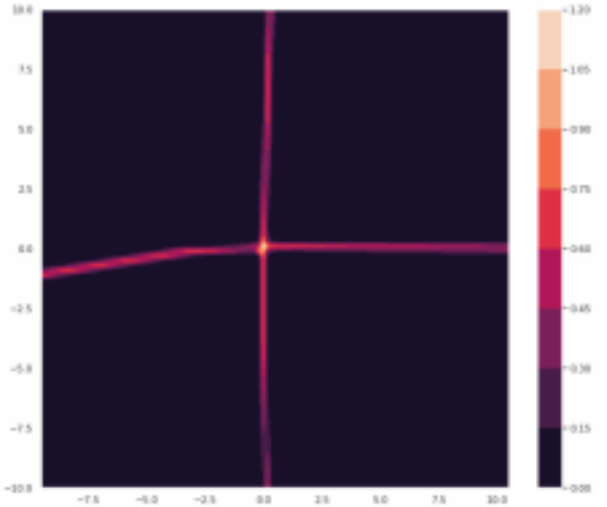
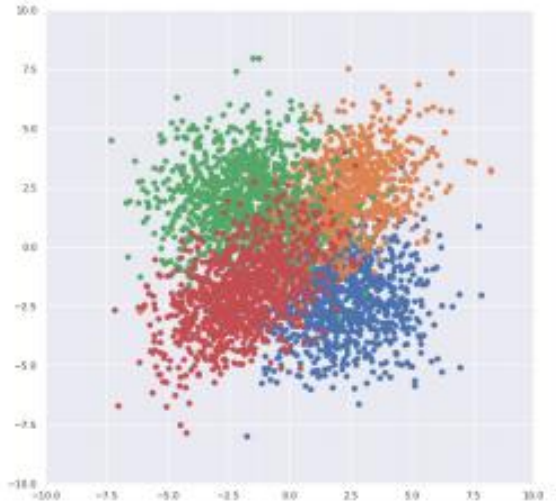
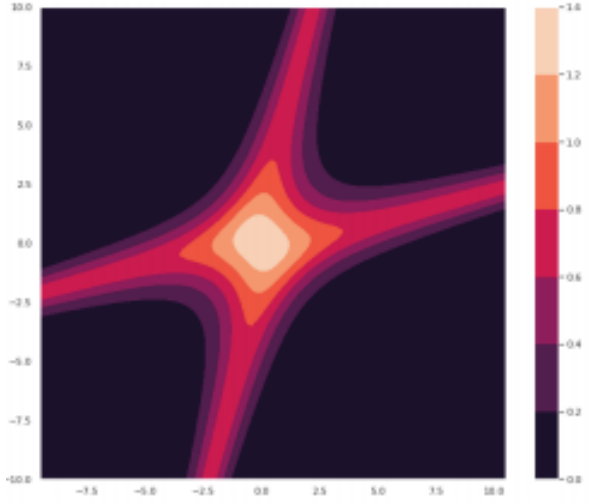
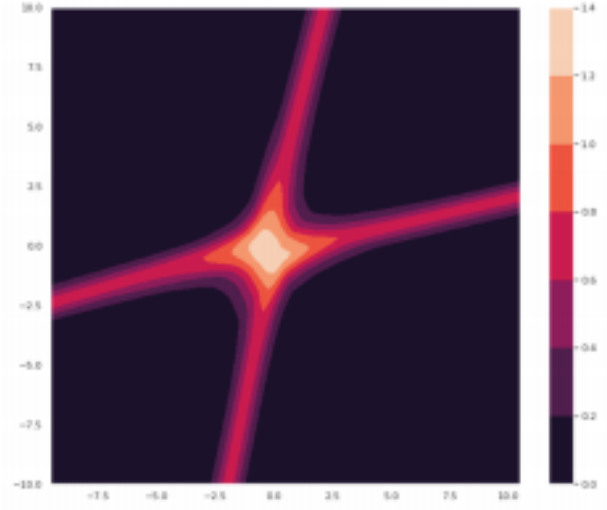
Forte incertitude
(complètement aléatoire)



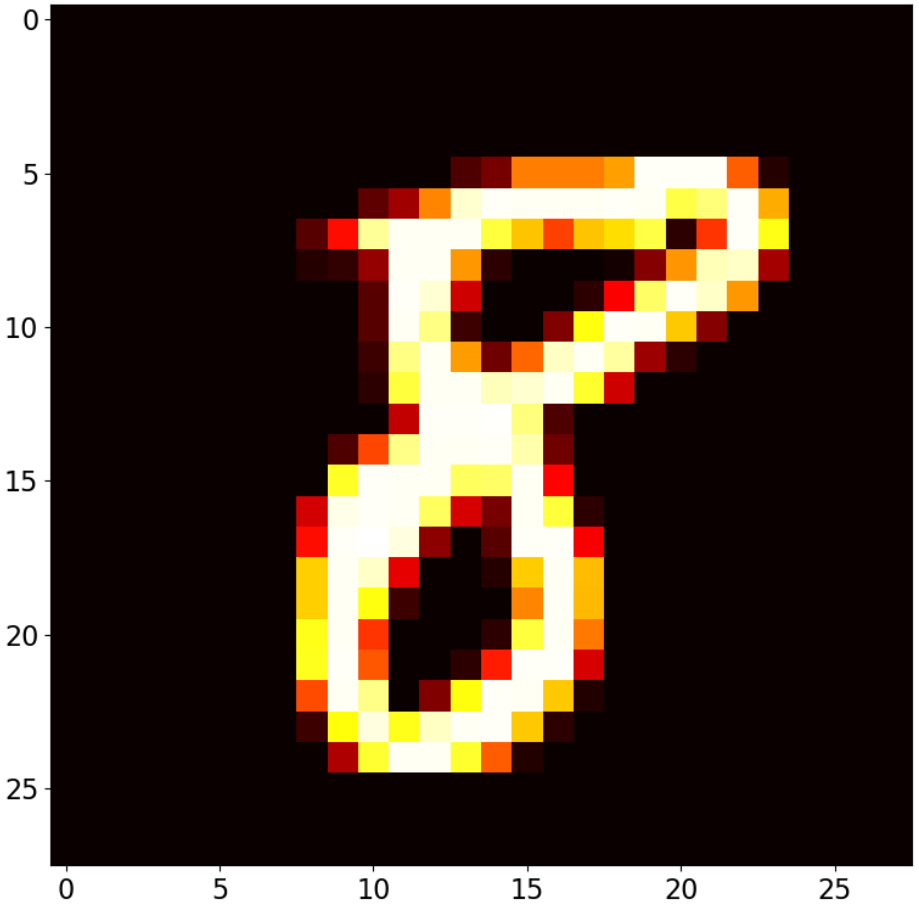
$1{,}1 = \log(3)$
(entropie maximale)

Exemple

Données générées à partir de 4 gaussiennes

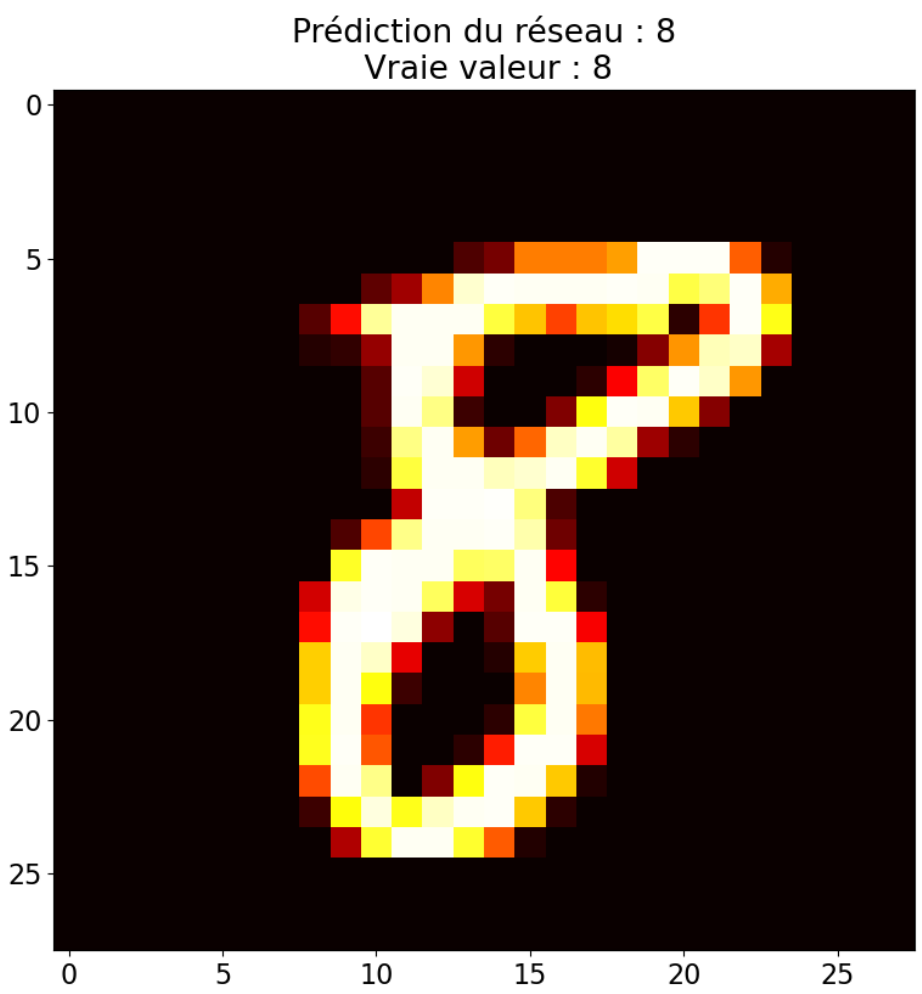
	Jeu de données généré	Entropie attendue (formule de Bayes)	Entropie obtenue avec un réseau de neurones
Classes bien séparées			
Classes superposées			

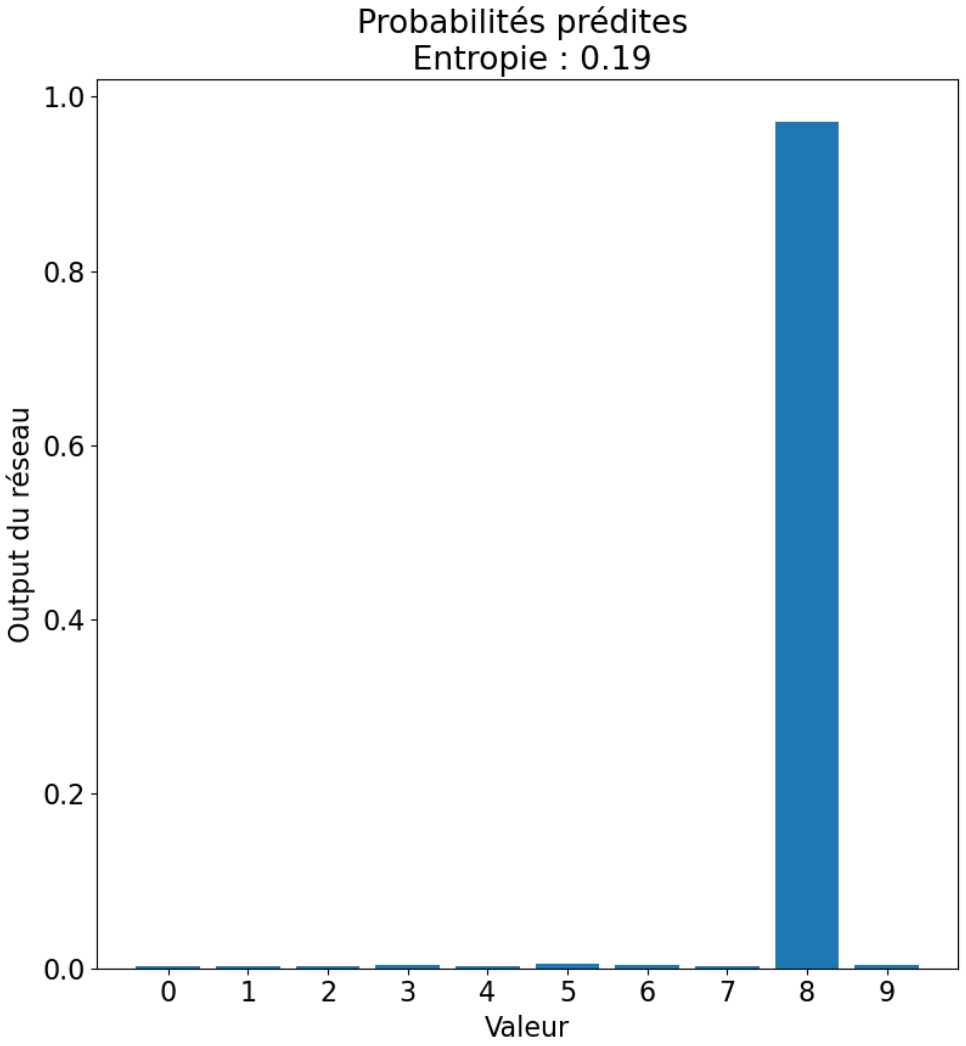
Exemple MNIST



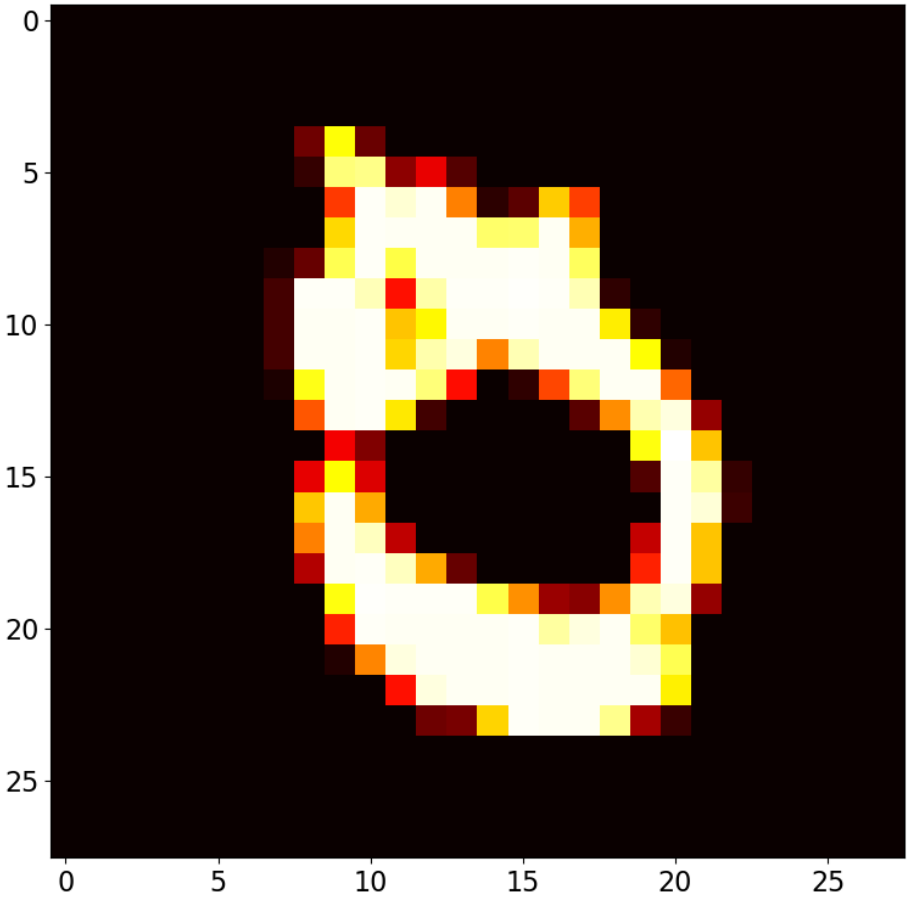
?

Exemple MNIST



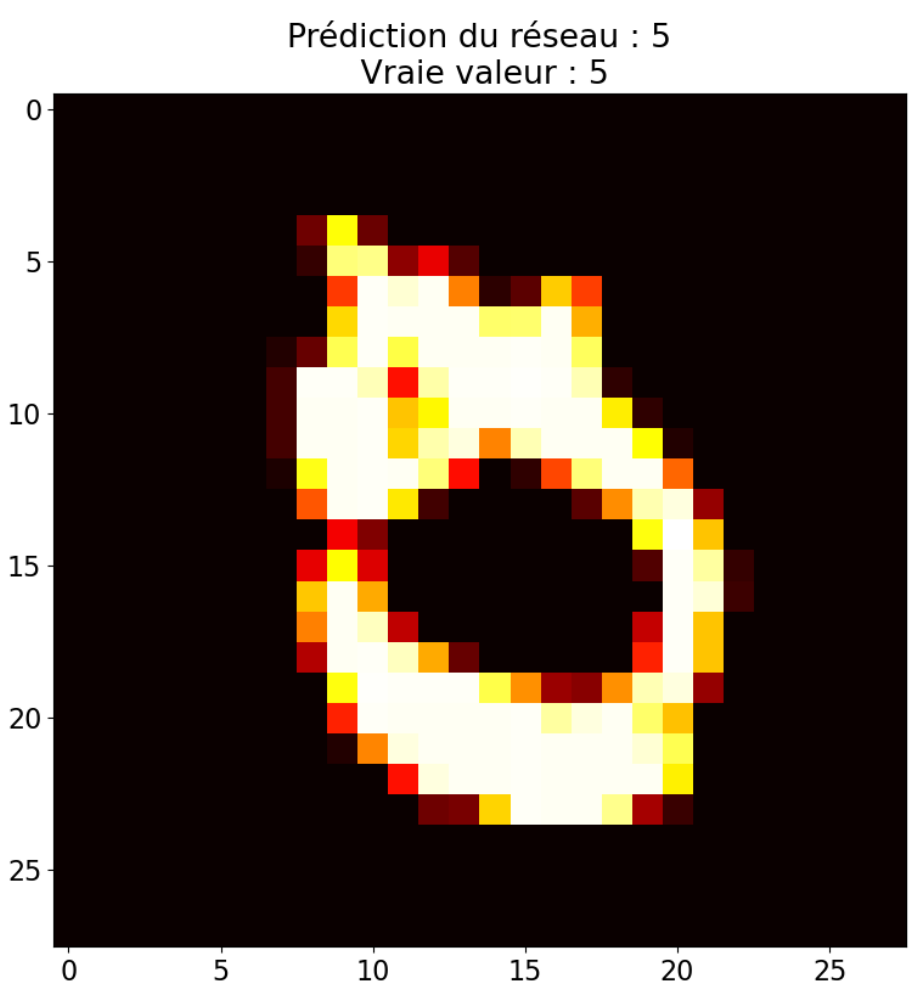


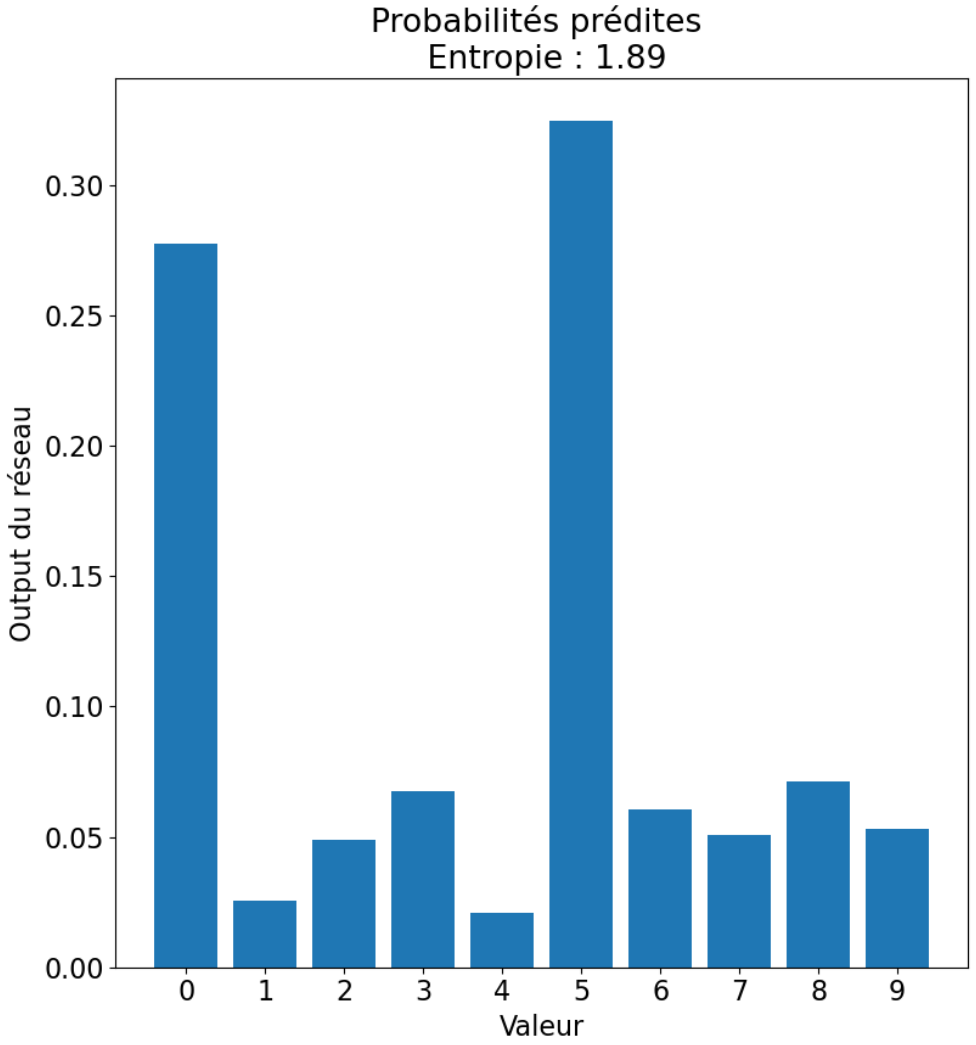
Exemple MNIST



?

Exemple MNIST





Problème de classification : incertitude épistémique

L'incertitude épistémique représente la part d'incertitude liée aux paramètres du réseau.

Les poids du réseau sont représentés par une variable aléatoire, de densité de probabilité a priori inconnue $\boldsymbol{p(\omega|D)}$, où D sont les données d'entraînement.

Quantification par l'in**formation mutuelle** entre les sorties y et les poids du réseau ω, conditionnellement à l'entrée considérée x et aux données d'entraînement D :

$$\underbrace{\mathcal{I}(y;\omega|x;D)}_{\text{Incertitude épistémique}} = \underbrace{\mathcal{H}\left(\mathbb{E}_{\omega \backsim p(\omega|D)}\left(p(y|x;\omega)\right)\right)}_{\text{Incertitude totale}} - \underbrace{\mathbb{E}_{\omega \backsim p(\omega|D)}\left(\mathcal{H}\left(p(y|x;\omega)\right)\right)}_{\text{Incertitude aléatoire}}$$

Information mutuelle	=	Entropie de la moyenne des prédictions	–	Moyenne des entropies des prédictions

Problème de classification : incertitude épistémique, en pratique

Principale difficulté : estimation de $p(\omega|D)$ → problème toujours ouvert en Deep Learning, plusieurs méthodes proposées, nous en parlerons dans la suite de la présentation (réseaux de neurones bayésiens)

Supposons $p(\omega|D)$ tout de même connue (ou du moins estimée)
Information mutuelle reste incalculable analytiquement → utilisation d'un échantillonnage Monte-Carlo

Tirage de T échantillons $\{\omega_1, \omega_2, \dots, \omega_T\}$ selon $p(\omega|D)$

Calcul de la prédiction moyenne : $$\mathbb{E}_{\omega \backsim p(\omega|D)}\left(p\left(y^{(i)}|x;\omega\right)\right) \cong \hat{p}\left(y^{(i)}|x\right) = \frac{1}{T}\sum_t p\left(y^{(i)}|x;\omega_t\right)$$

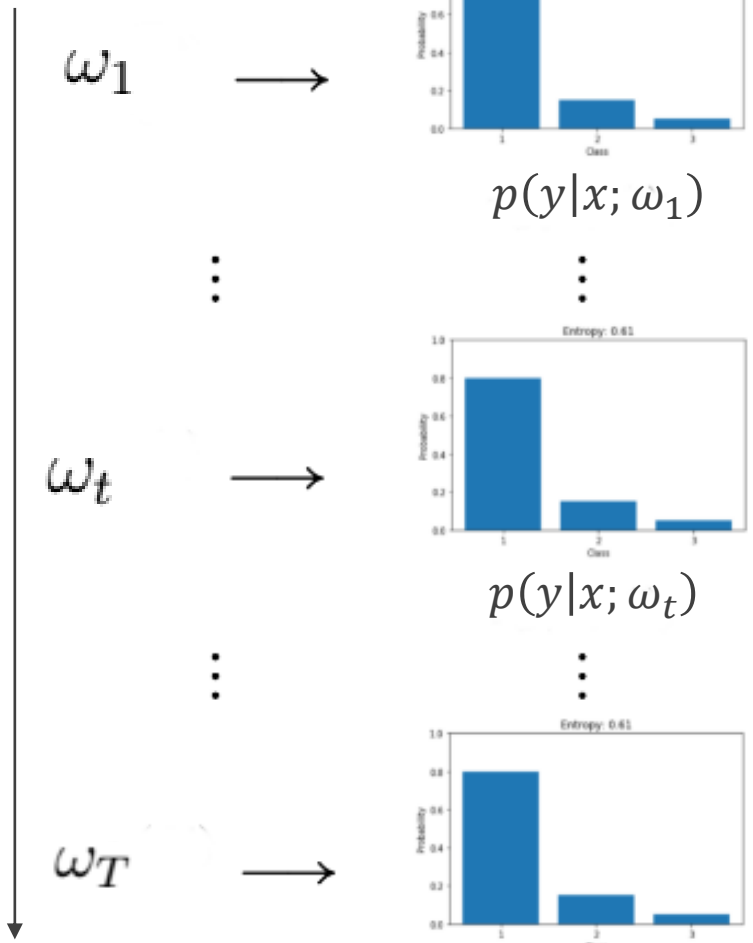
Calcul de l'entropie totale : $$\mathcal{H}\left(\hat{p}(y|x)\right) = -\sum_i \hat{p}\left(y^{(i)}|x\right)\log\left(\hat{p}\left(y^{(i)}|x\right)\right)$$

Calcul de l'entropie aléatoire : $$\widehat{\mathcal{H}}\left(p(y|x;\omega)\right) = -\frac{1}{T}\sum_t\sum_i p\left(y^{(i)}|x;\omega_t\right)\log\left(p\left(y^{(i)}|x;\omega_t\right)\right)$$

Calcul de l'information mutuelle : $$\hat{\mathcal{I}}(y;\omega|x,D) = \mathcal{H}\left(\hat{p}(y|x)\right) - \widehat{\mathcal{H}}\left(p(y|x;\omega)\right)$$

Intuitivement

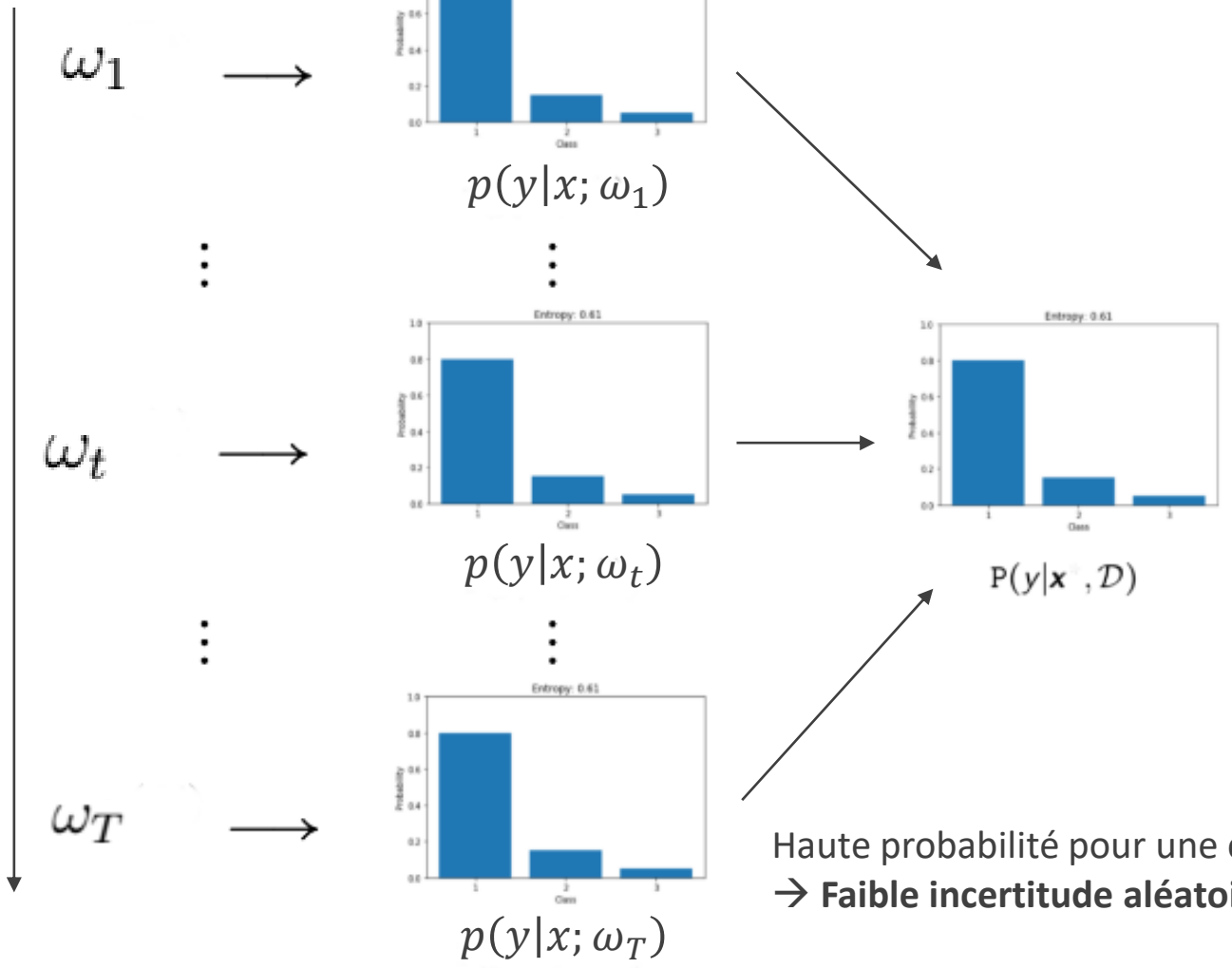
Différents paramètres tirés



?

Intuitivement

Différents paramètres tirés



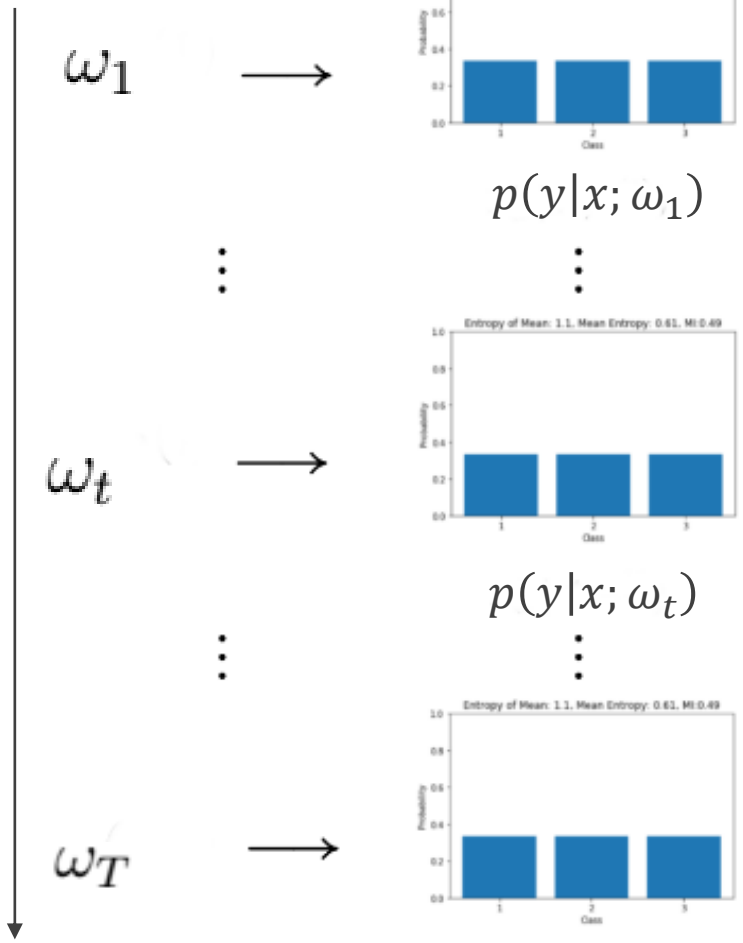
Faible variabilité entre chaque modèle
→ **Faible incertitude épistémique**

Haute probabilité moyenne pour une classe
→ **Faible incertitude totale**

Haute probabilité pour une classe dans chaque modèle
→ **Faible incertitude aléatoire**

Intuitivement

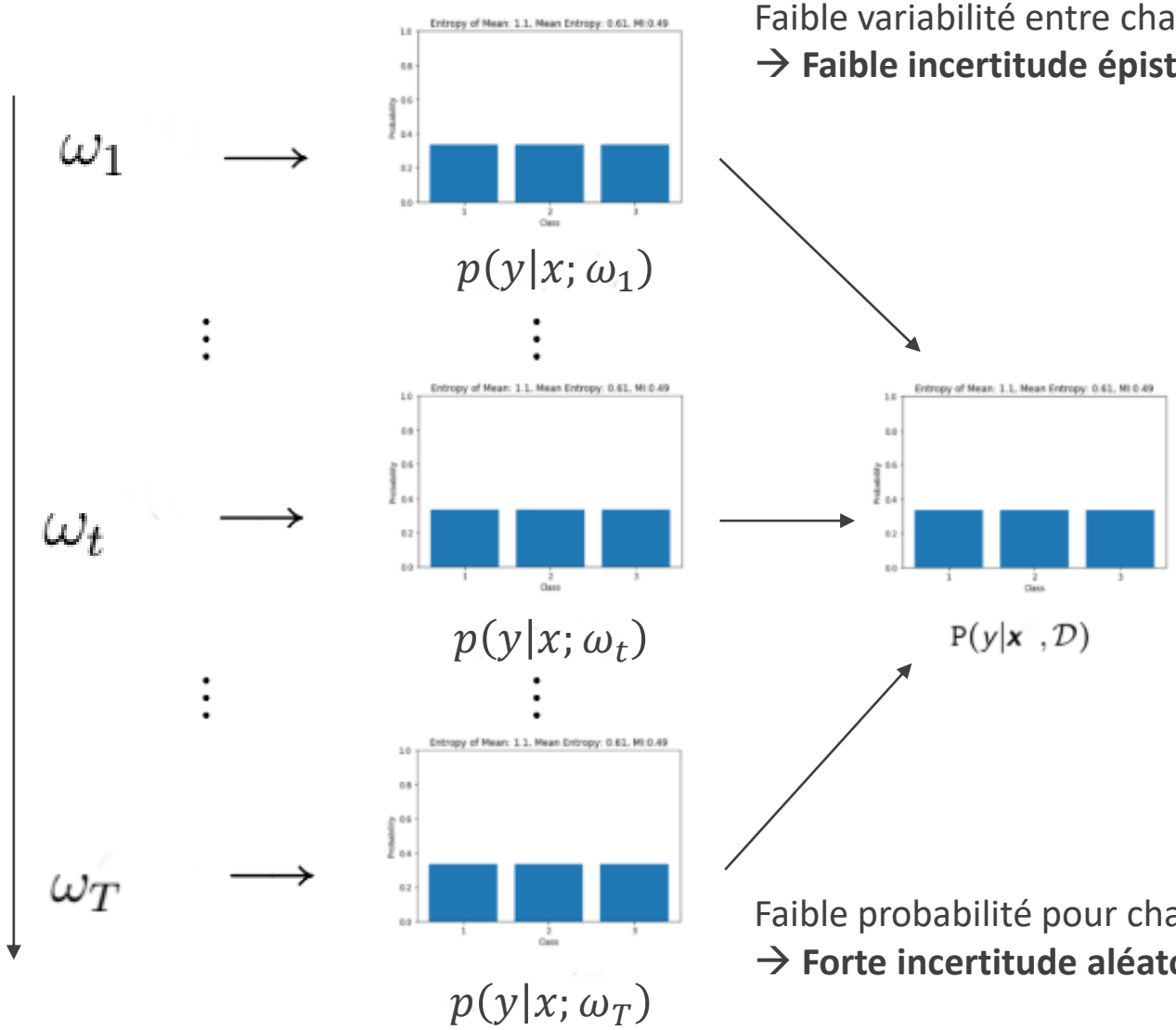
Différents paramètres tirés



?

Intuitivement

Différents paramètres tirés



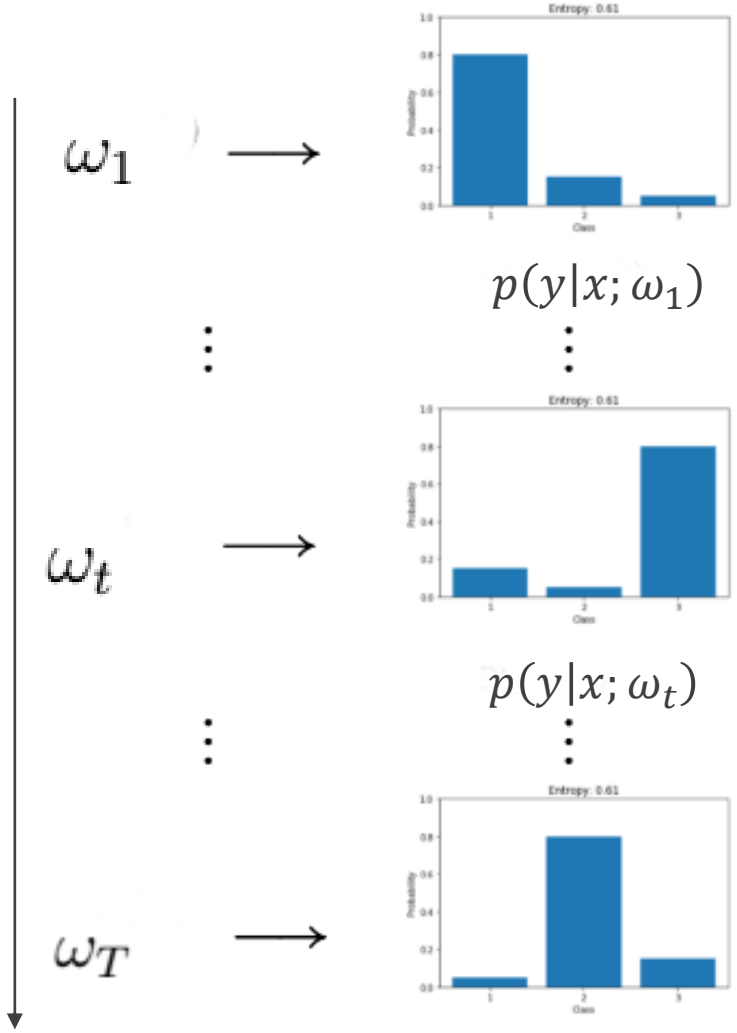
Faible variabilité entre chaque modèle
→ **Faible incertitude épistémique**

Faible probabilité moyenne pour chaque classe
→ **Forte incertitude totale**

Faible probabilité pour chaque classe dans tous les modèles
→ **Forte incertitude aléatoire**

Intuitivement

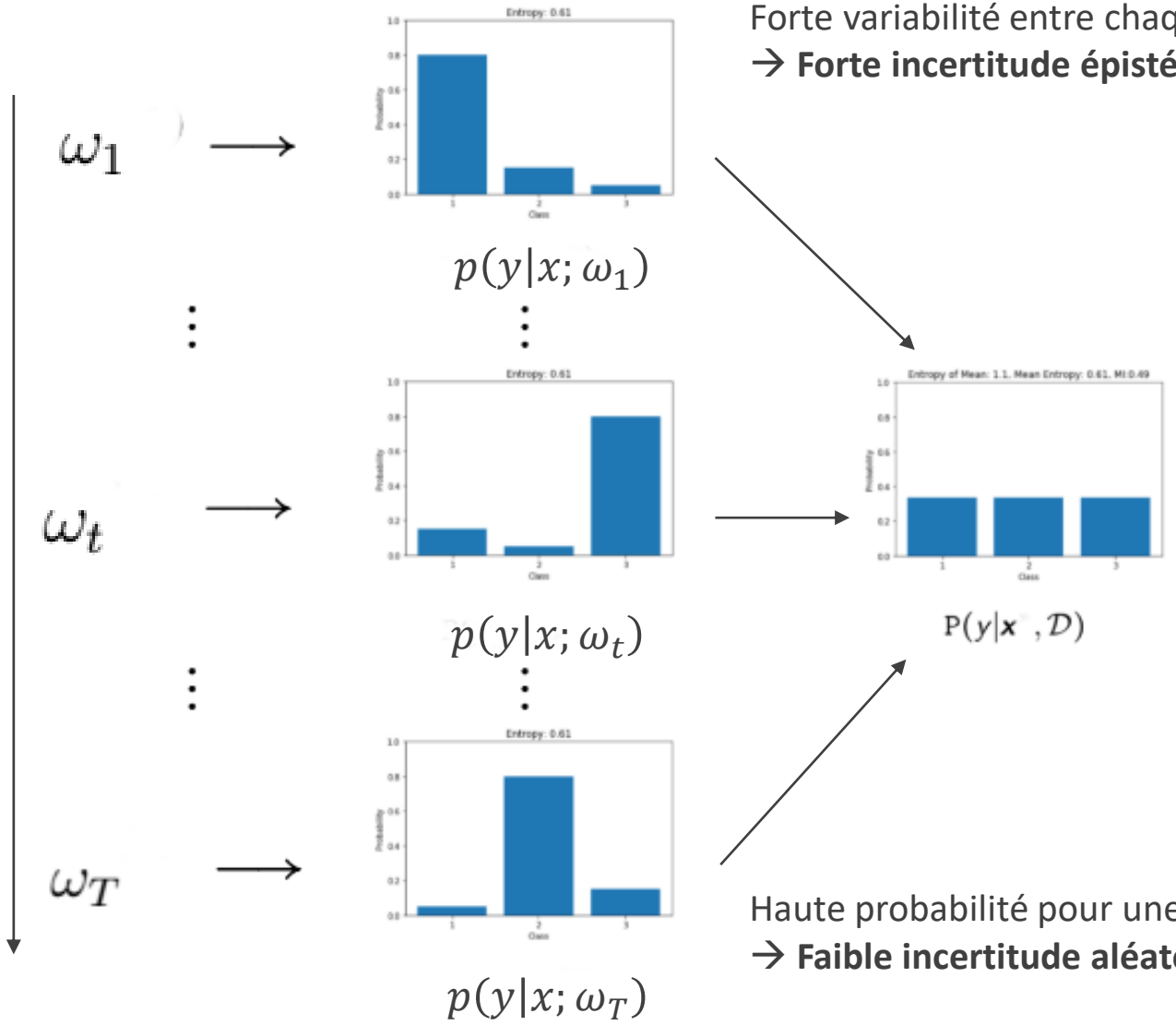
Différents paramètres tirés



?

Intuitivement

Différents paramètres tirés



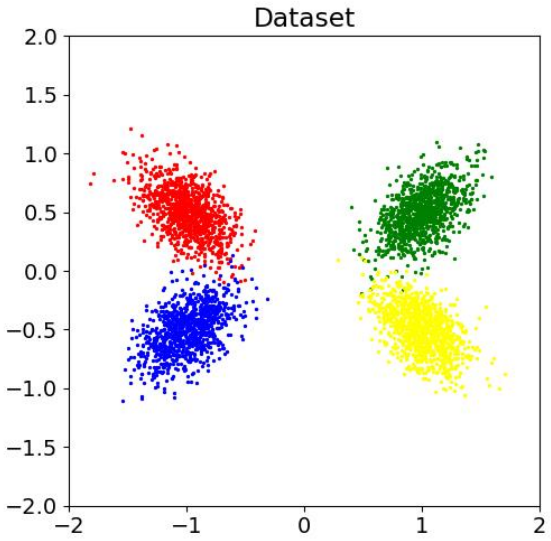
Forte variabilité entre chaque modèle
→ **Forte incertitude épistémique**

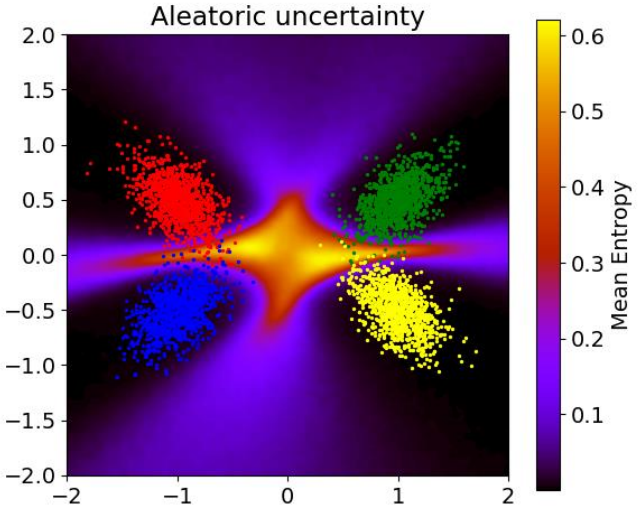
Faible probabilité moyenne pour chaque classe
→ **Forte incertitude totale**

Haute probabilité pour une classe dans chaque modèle
→ **Faible incertitude aléatoire**

Exemple

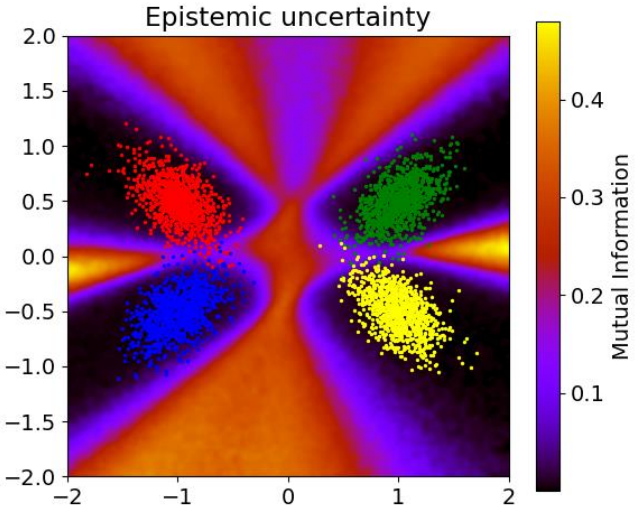
Retour sur les données jouet (4 gaussiennes 2D)



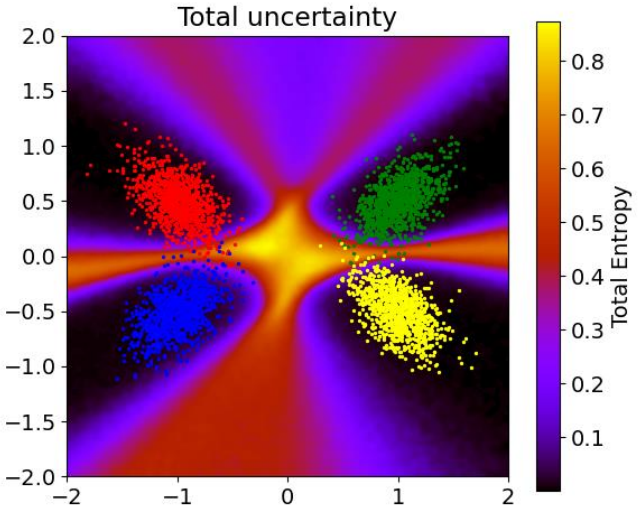


Incertitude aléatoire prédite par un réseau de neurones bayésien

Utilisation du Monte-Carlo Dropout



Incertitude épistémique prédite par un réseau de neurones bayésien



Incertitude totale prédite par un réseau de neurones bayésien

Problème de régression : approche classique

Soit un modèle f_ω de paramètres ω, $(x_j, y_j)_j$ des données d'apprentissage.

Objectif de l'apprentissage classique : régression moindres carrés

$$\boldsymbol{\omega^* = \underset{\omega}{\operatorname{argmin}} \frac{1}{N} \left\| y_j - f_\omega(x_j) \right\|_2^2}$$

Hypothèses sous-jacentes :

- les données suivent une loi normale $y_j \sim \mathcal{N}(\mu_j, \sigma_j^2 \mathbb{I})$
- les données sont homoscédastiques, σ_j est identique pour tous les exemples

Sous ces hypothèses, la vraisemblance s'écrit (d, dimension des données de sortie) :

$$L(\omega) = \prod_j \frac{1}{\left(2\pi\sigma_j^2\right)^{\frac{d}{2}}} \exp\left(- \frac{\left\| y_j - f_\omega(x_j) \right\|_2^2}{2\sigma_j^2} \right)$$

Négative log vraisemblance + σ_j constant

Problème de régression : incertitude aléatoire

Density Neural Networks

Modélisons l'incertitude aléatoire par une loi de probabilité $p(y|\theta(x))$ de paramètres $\theta(x)$.
L'objectif du réseau de neurones est de prédire ces paramètres θ : $f_\omega(x) = \theta(x)$
Remarque : il peut y avoir un modèle spécifique pour chaque paramètre.

$$\boldsymbol{\omega^* = \underset{\omega}{\operatorname{argmin}} - \sum_{j} \log(p(y_j|f_\omega(x_j))}$$

Exemple : cas d'une gaussienne 1D, $\theta(x) = (\mu(x), \sigma(x))$.
On ne fait plus l'hypothèse d'homoscédasticité, σ peut dépendre de l'entrée x !

$$p(y|f_\omega(x)) = \frac{1}{\sqrt{2\pi} f_\omega^{(2)}(x)} \exp\left(-\frac{1}{2} \frac{\left(y - f_\omega^{(1)}(x)\right)^2}{f_\omega^{(2)}(x)^2} \right)$$

Négative log vraisemblance : $$\boldsymbol{l(\omega) = \sum_j \log\left(\sqrt{2\pi} f_\omega^{(2)}(x_j)\right) + \frac{1}{2} \frac{\left(y_j - f_\omega^{(1)}(x_j)\right)^2}{f_\omega^{(2)}(x_j)^2}}$$

Remarques :

- il n'est pas nécessaire d'avoir une connaissance du « vrai » écart-type dans les données d'entraînement
- Fonctionne pour d'autres lois, mais conditions de convergence plus difficiles à garantir parfois (divergence de la vraisemblance par exemple)

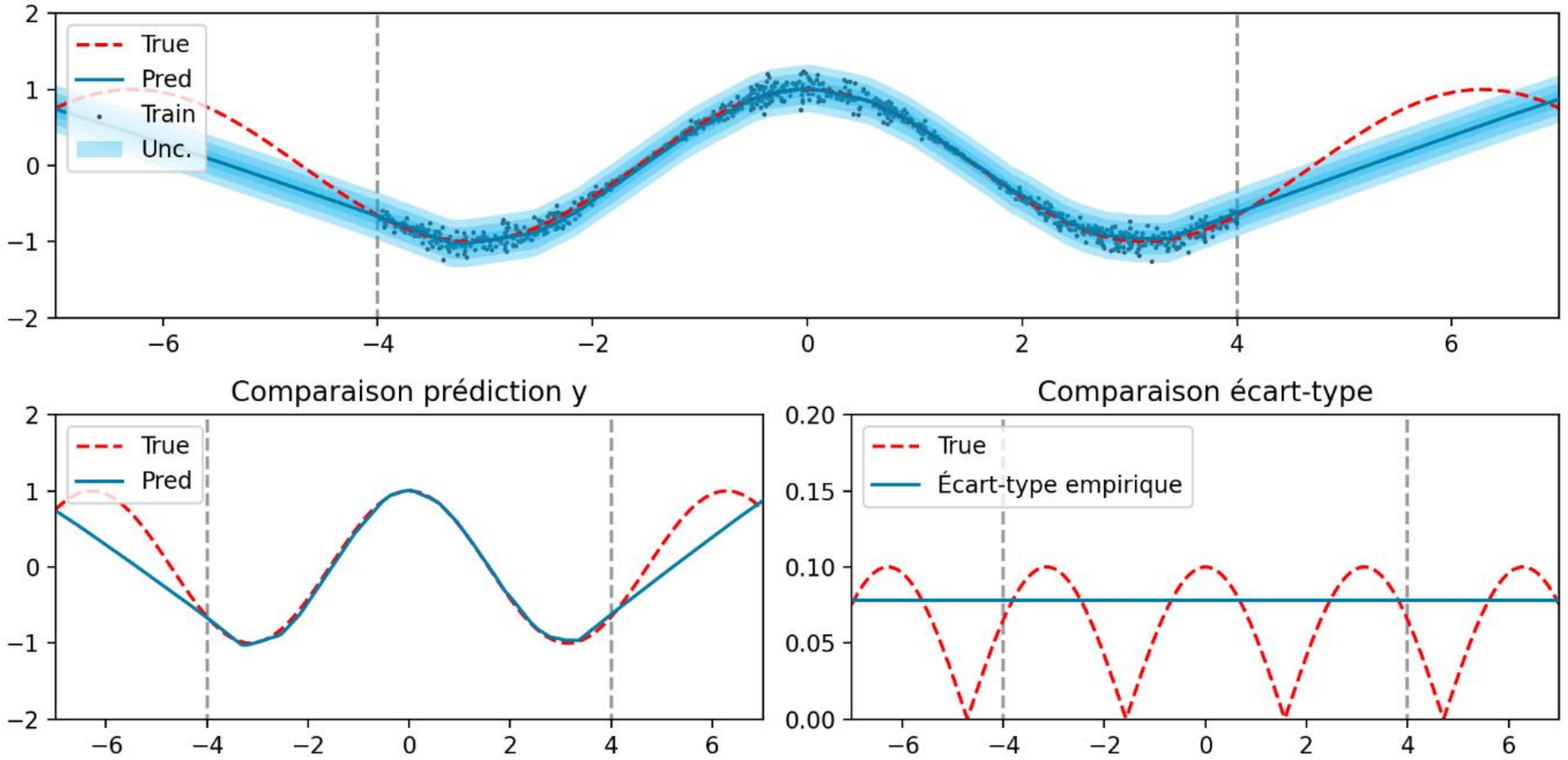
Problème de régression : incertitude aléatoire

Density Neural Networks : un exemple

Données générées avec bruit gaussien hétéroscédastique

Approche classique :

- Estimation par moindres carrés
- Écart-type estimé empiriquement a posteriori sur les exemples

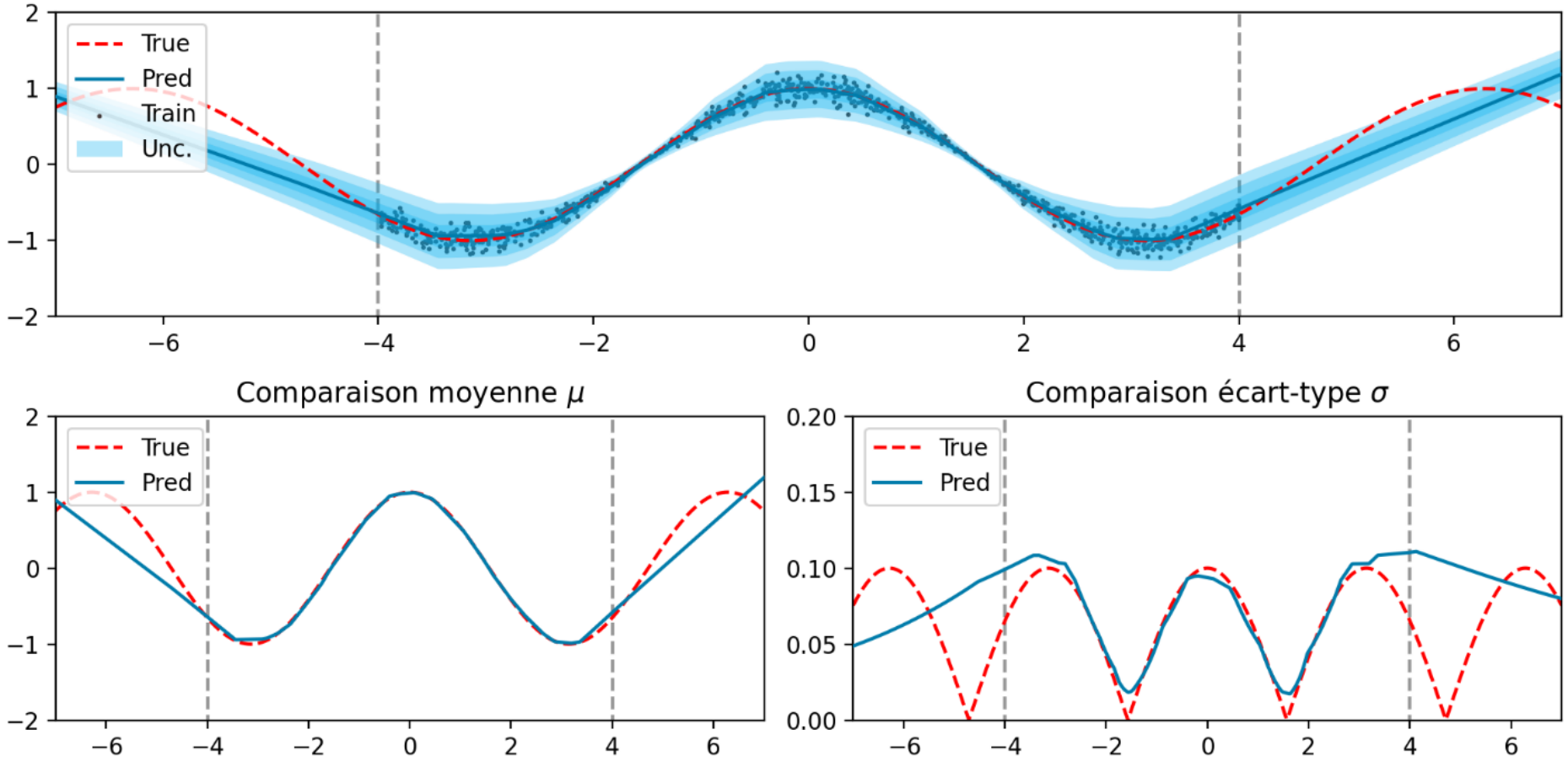


Problème de régression : incertitude aléatoire

Density Neural Networks : un exemple

Données générées avec bruit gaussien hétéroscédastique

Approche par Mixture Density Networks

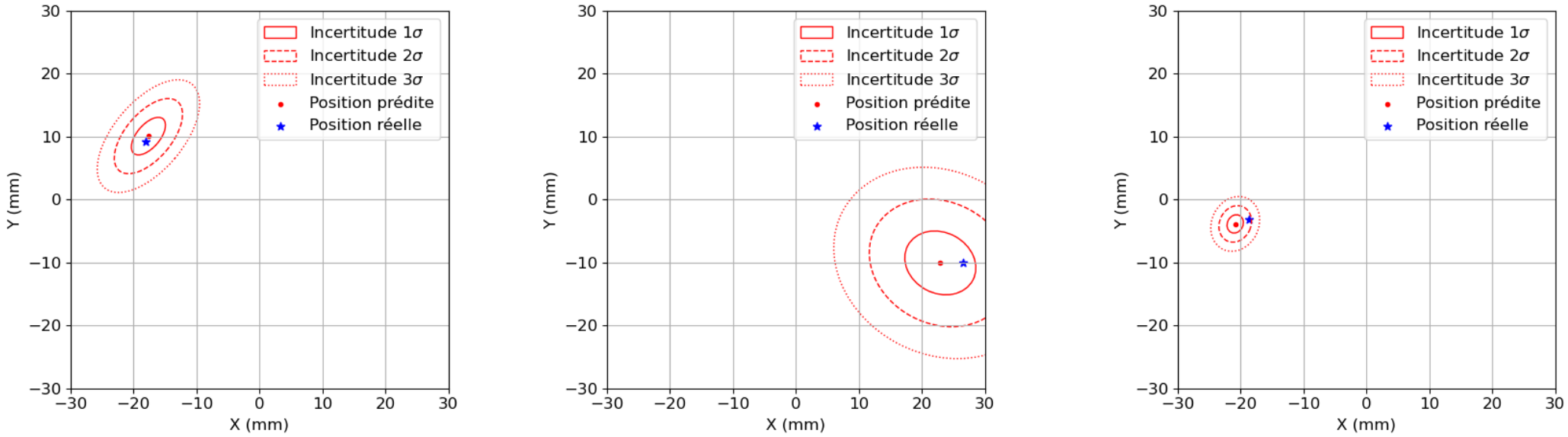


Problème de régression : incertitude aléatoire

Density Neural Networks : application à des données d'imageur médical
PTC AAIMME (DRF/Irfu, DRF/SHFJ, DES/LGLS) au sein du Projet ClearMind

Entrée : formes d'impulsions électriques enregistrées par un détecteur
Sortie : position de l'interaction gamma

Modélisation de l'erreur sous forme d'une gaussienne 2D



Problème de régression : incertitude épistémique

Même interprétation qu'en classification : dépendance des sorties du réseau par rapport aux poids du réseau qui suivent une distribution $p(\omega|D)$

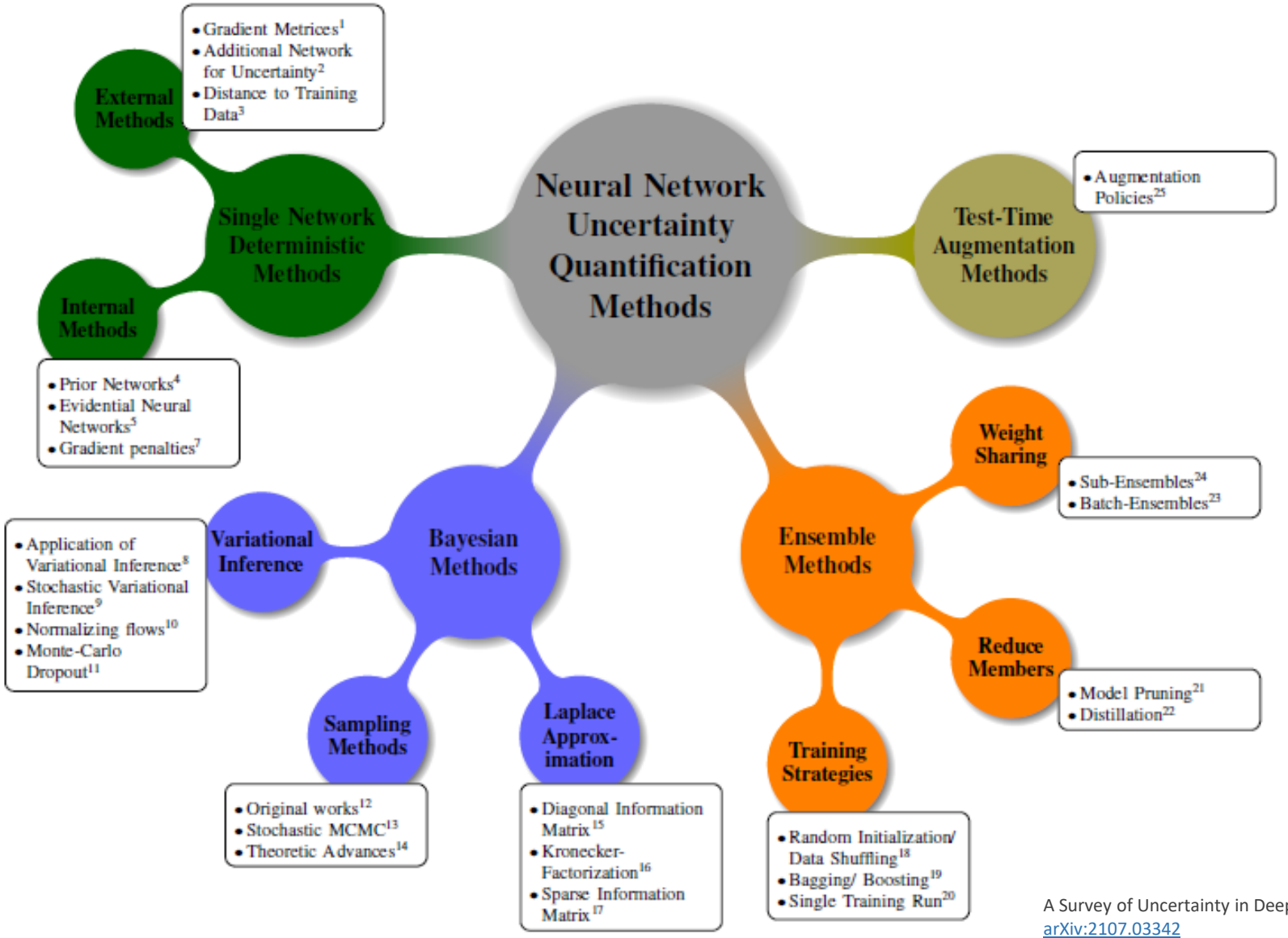
Même problématique principale qu'en classification : estimation de $p(\omega|D)$

Quantité d'intérêt pour mesurer l'incertitude épistémique : la variance

$$\underbrace{\mathbb{V}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{D})}_{\text{Incertitude totale}} = \underbrace{\mathbb{V}_{\boldsymbol{\omega}\sim p(\boldsymbol{\omega}|\boldsymbol{D})}\left(\mathbb{E}_{\boldsymbol{y}\sim p\left(\boldsymbol{y}|\boldsymbol{f}_{\boldsymbol{\omega}}(\boldsymbol{x})\right)}(\boldsymbol{y})\right)}_{\text{Incertitude épistémique}} + \underbrace{\mathbb{E}_{\boldsymbol{\omega}\sim p(\boldsymbol{\omega}|\boldsymbol{D})}\left(\mathbb{V}_{\boldsymbol{y}\sim p\left(\boldsymbol{y}|\boldsymbol{f}_{\boldsymbol{\omega}}(\boldsymbol{x})\right)}(\boldsymbol{y})\right)}_{\text{Incertitude aléatoire}}$$

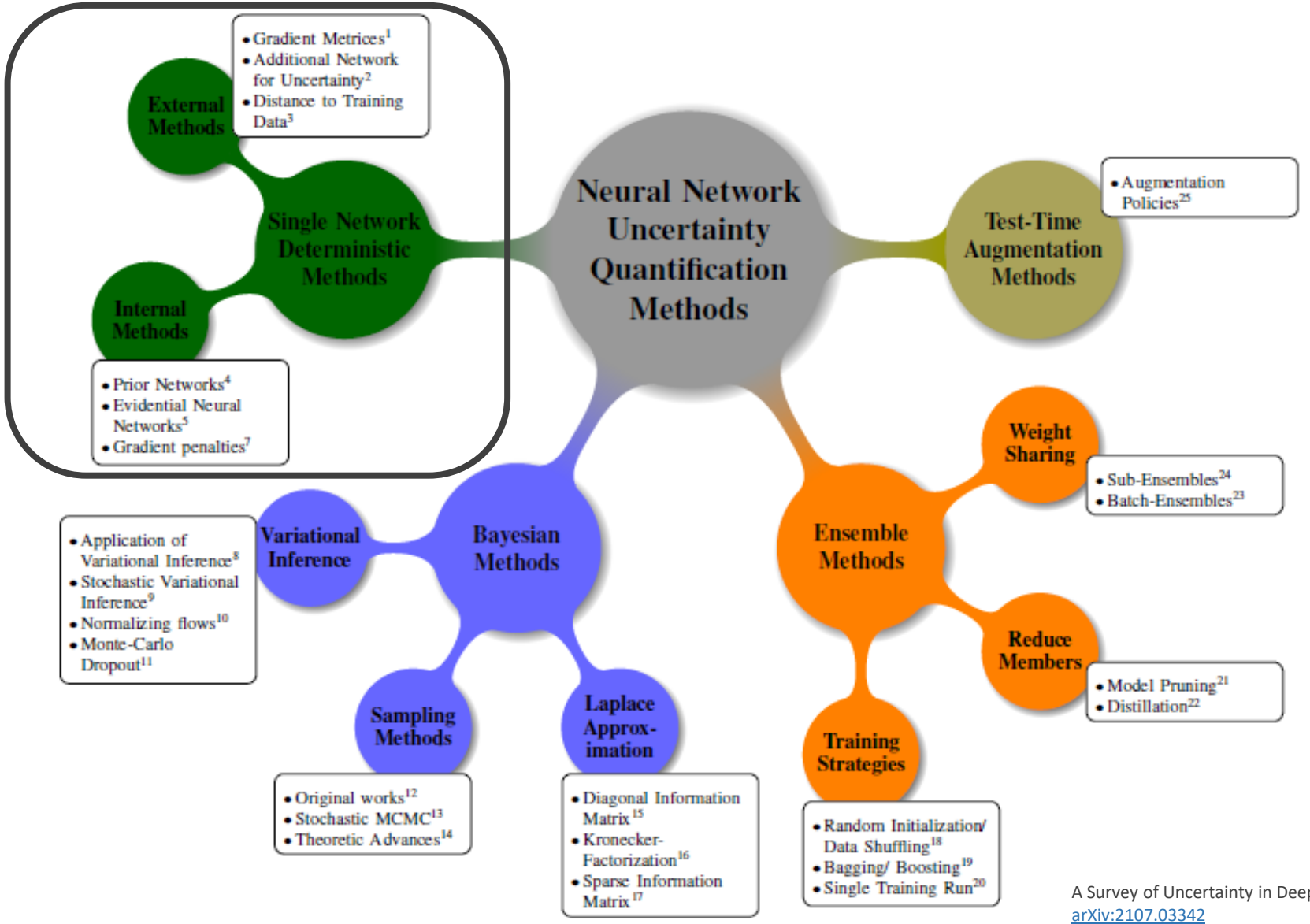
Utilisation de l'entropie différentielle (au lieu de l'entropie de Shannon) ?

Différentes approches Deep Learning



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Différentes approches Deep Learning



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Single Network Deterministic methods :

Utilisation d'un réseau de neurones unique et déterministe pour estimer les incertitudes

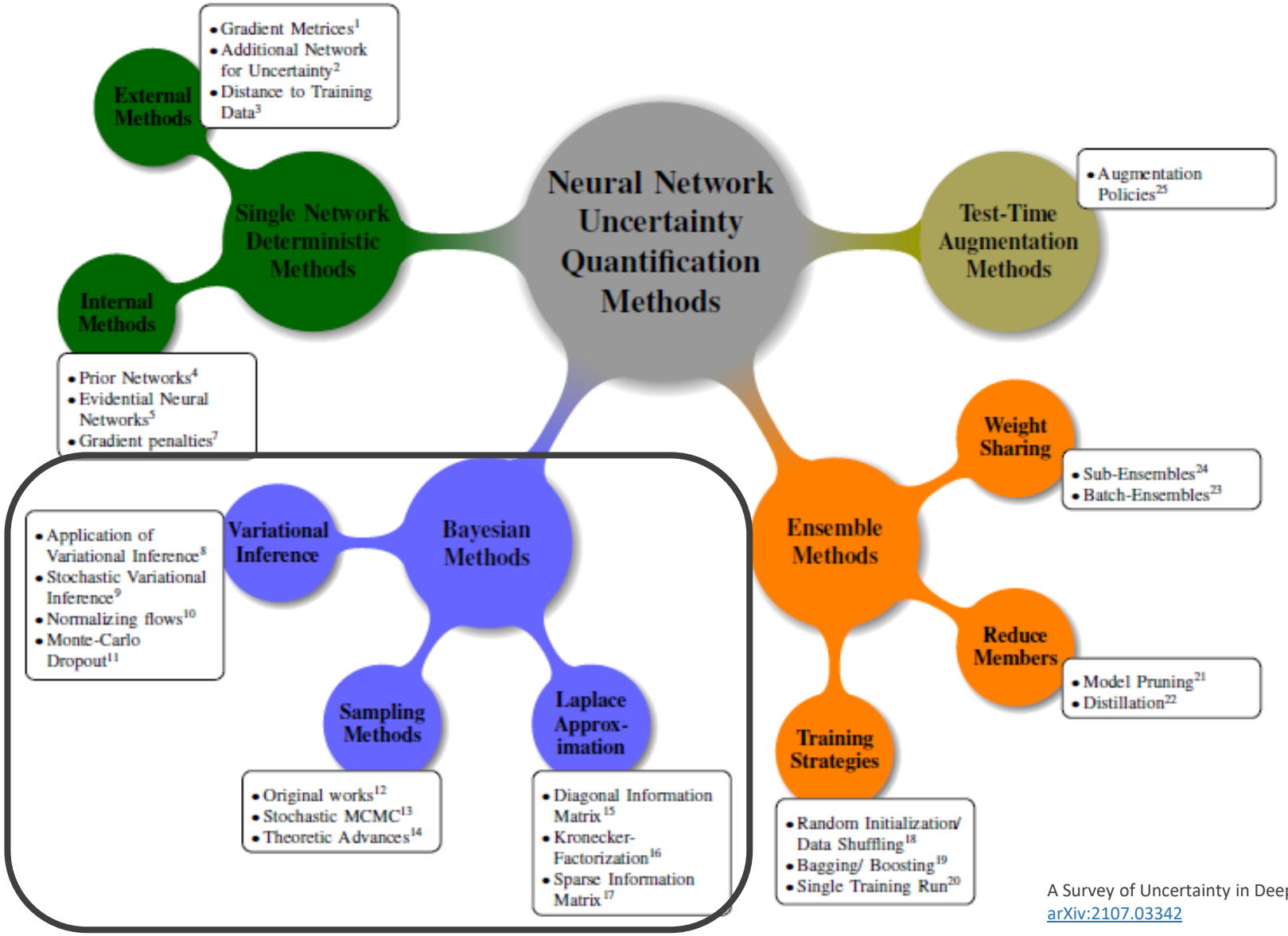
Exemples :

- **Additionnal Network for Uncertainty** : pour des problèmes de classification, on a plusieurs réponses pour une même donnée d'entrée. Par exemple, le diagnostic de plusieurs médecins sur une même image. Un réseau apprend les fréquences sur chaque image.

- **Evidential Neural Network** : étend l'idée des Mixture Density Networks en introduisant une loi sur les paramètres des lois.

  Exemple :
  - loi normale sur les sorties : $y \sim \mathcal{N}(\mu, \sigma^2)$
  - les paramètres de la loi normale suivent eux-mêmes une loi modélisée par :
    - $\mu \sim \mathcal{N}(\gamma, v^{-1}\sigma)$
    - $\sigma \sim \Gamma^{-1}(\alpha, \beta)$
  - le réseau estime les 4 paramètres $(\gamma, v, \alpha, \beta)$
  - la prédiction directe est représentée par $\mathbb{E}(\mu) = \gamma$
  - l'incertitude aléatoire est donnée par $\mathbb{E}(\sigma^2) = \frac{\beta}{\alpha-1}$ et l'incertitude épistémique par $\mathbb{V}(\mu) = \frac{\beta}{v(\alpha-1)}$

Différentes approches Deep Learning



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Approches bayésiennes :

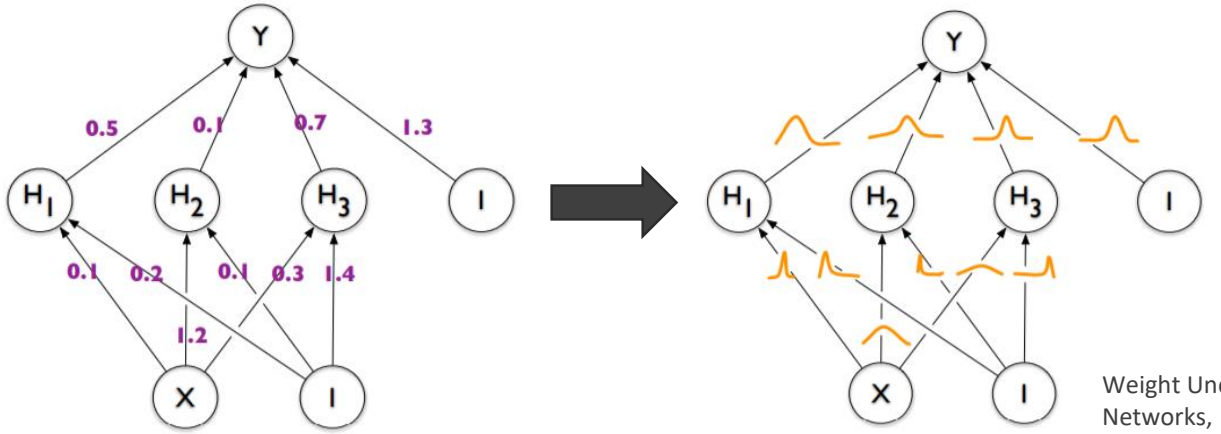
Idée principale : être capable de réaliser un tirage de ω selon la loi de densité $p(\omega|D)$ inconnue

Plusieurs approches :

Bayes by backprop : inférence variationnelle

$p(\omega|D)$ toujours impossible à calculer
→ On l'approche par une distribution $q(\omega|\Theta)$

Par exemple $\Theta_l = (\mu_l, \sigma_l)$ (loi normale)



Weight Uncertainty in Neural Networks, Blundell et al.

Objectif :

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \, \mathrm{KL}[q(\omega|\Theta)||p(\omega|D)] \qquad \text{KL : Divergence de Kullback-Leibler}$$

$$\Leftrightarrow \boldsymbol{\Theta^* = \underset{\Theta}{\operatorname{argmin}} \, \mathrm{KL}[q(\omega|\Theta)||p(\omega)] - \mathbb{E}_{q(\omega|\Theta)}[\log(p(D|\omega))]} \qquad \text{ELBO : Evidence Lower Bound}$$

Terme de régularisation
Prior $p(\omega)$ à définir

Minimisation de la vraisemblance
Calculée par échantillonnage Monte-Carlo
→ Inconvénient : apprentissage alourdi

Approches bayésiennes :

Idée principale : être capable de réaliser un tirage de ω selon la loi de densité $p(\omega|D)$ inconnue

Plusieurs approches :

Monte-Carlo Dropout

Extinction des neurones ou des poids / Bruit sur les neurones ou les poids
→ Méthode classique de régularisation

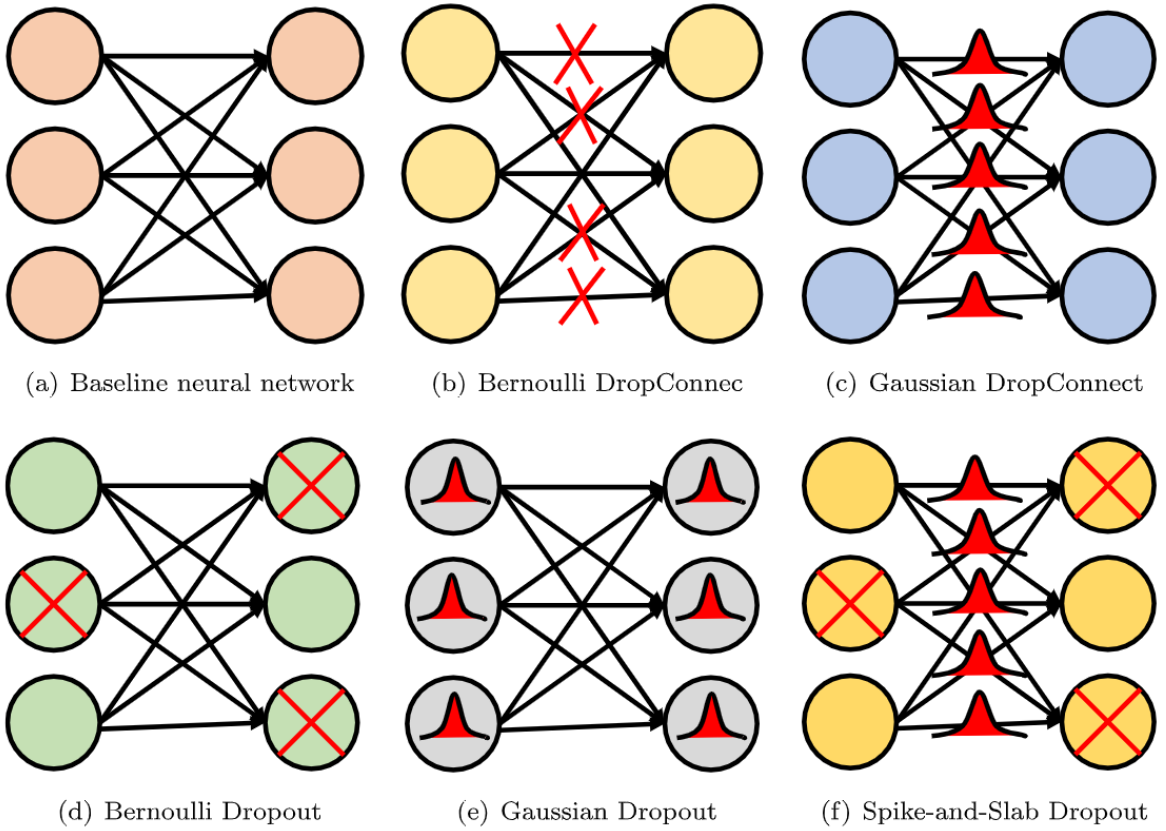
→ Ici appliqué en phase de test

Extinction des poids : chaque composante de ω suit une loi de Bernoulli

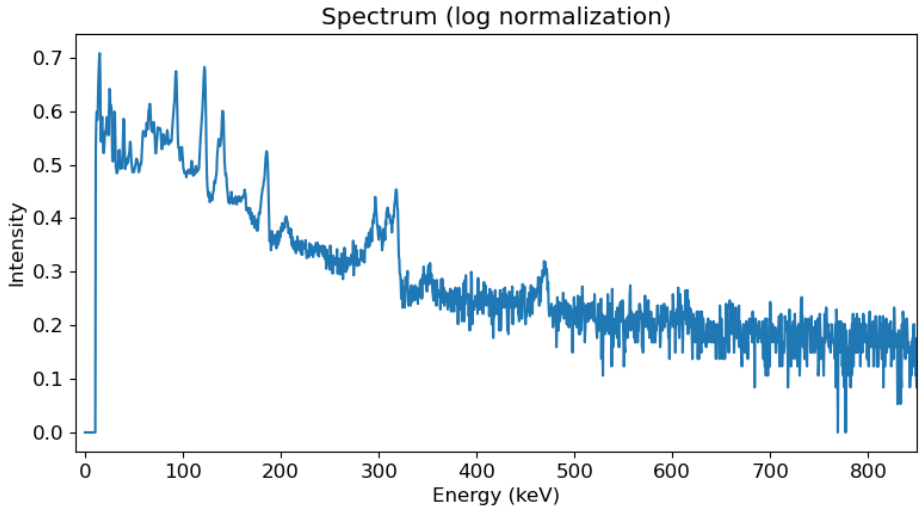
Bruit gaussien : chaque composante de ω suit une loi gaussienne dont on cherche à optimiser la moyenne (variance fixe)

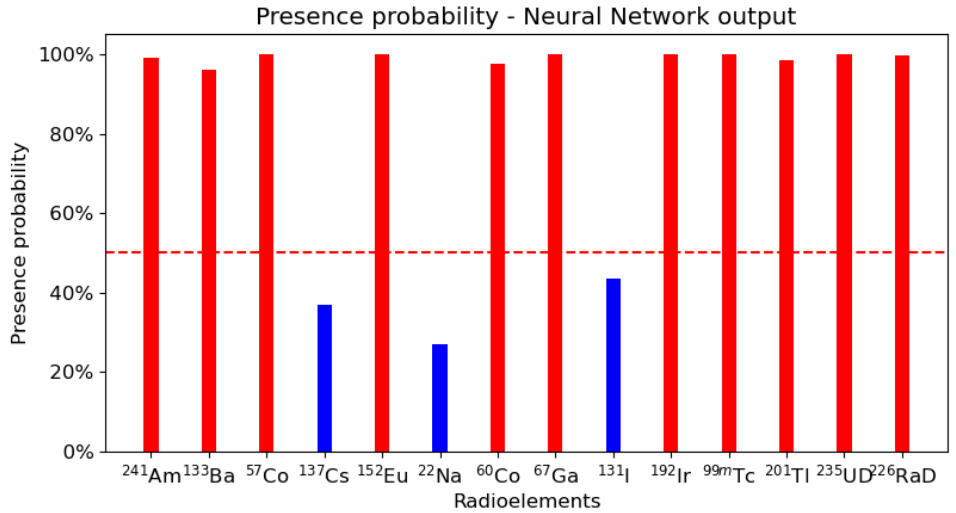
Prior $p(\omega)$ souvent choisi $\mathcal{N}(0, l)$ → terme de régularisation L_2

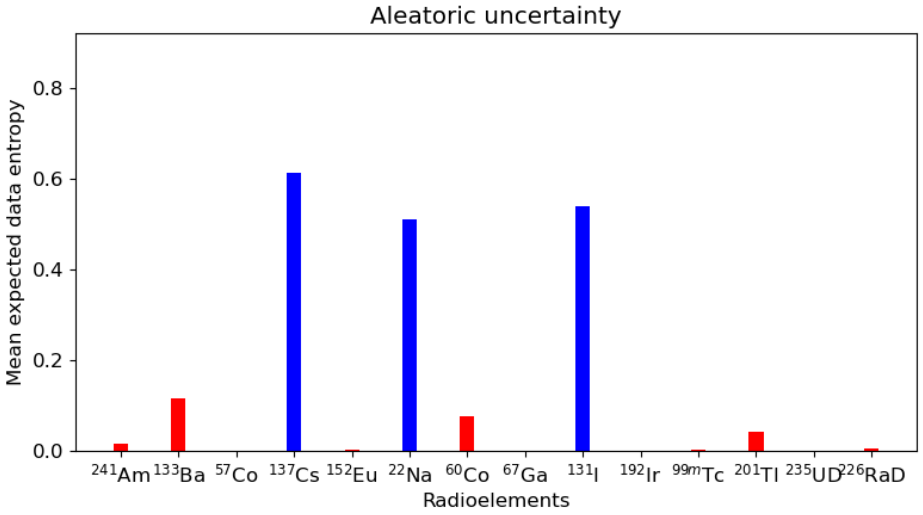
Inconvénients :
- Garanties théoriques faibles
- Taux de Dropout / Variance du bruit gaussien ?

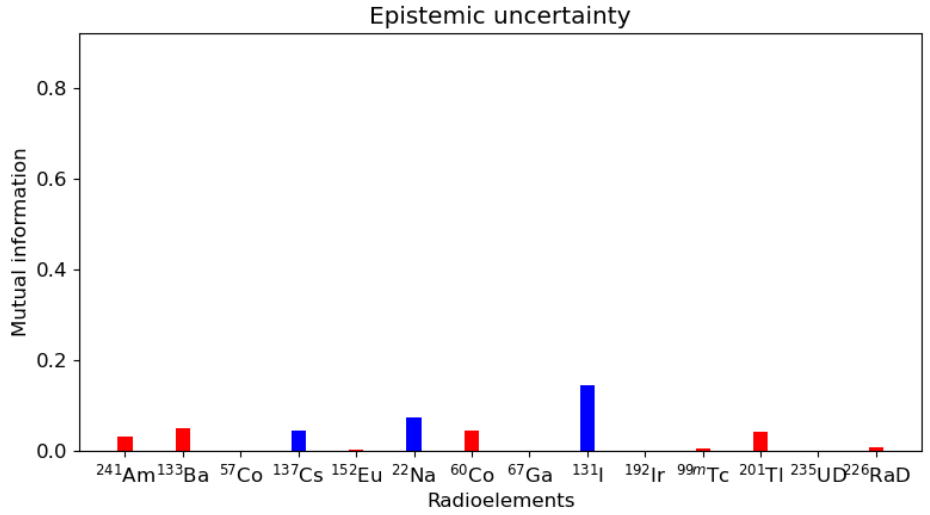

(a) Baseline neural network (b) Bernoulli DropConnec (c) Gaussian DropConnect
(d) Bernoulli Dropout (e) Gaussian Dropout (f) Spike-and-Slab Dropout

Exemple : application spectrométrie gamma





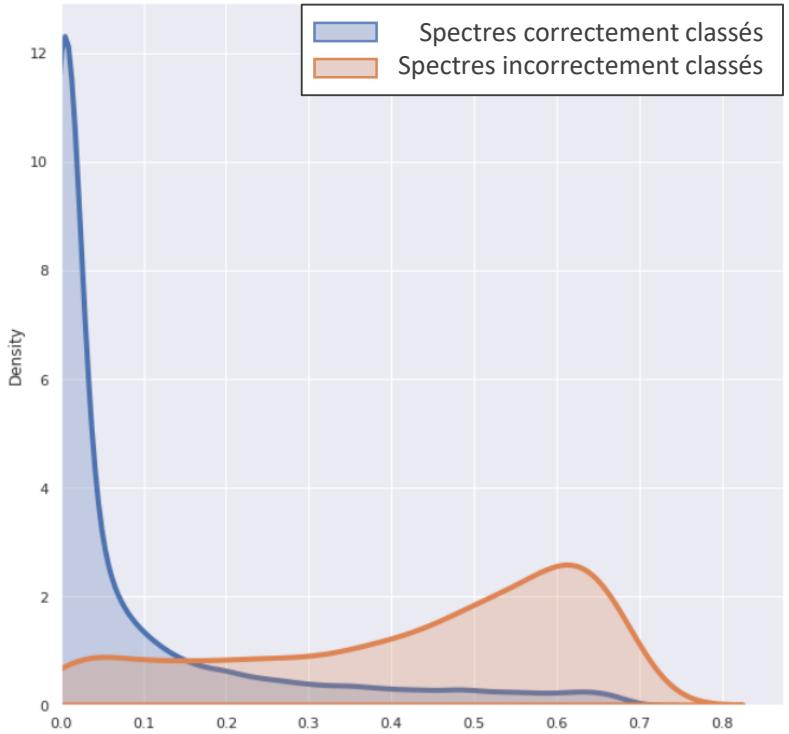




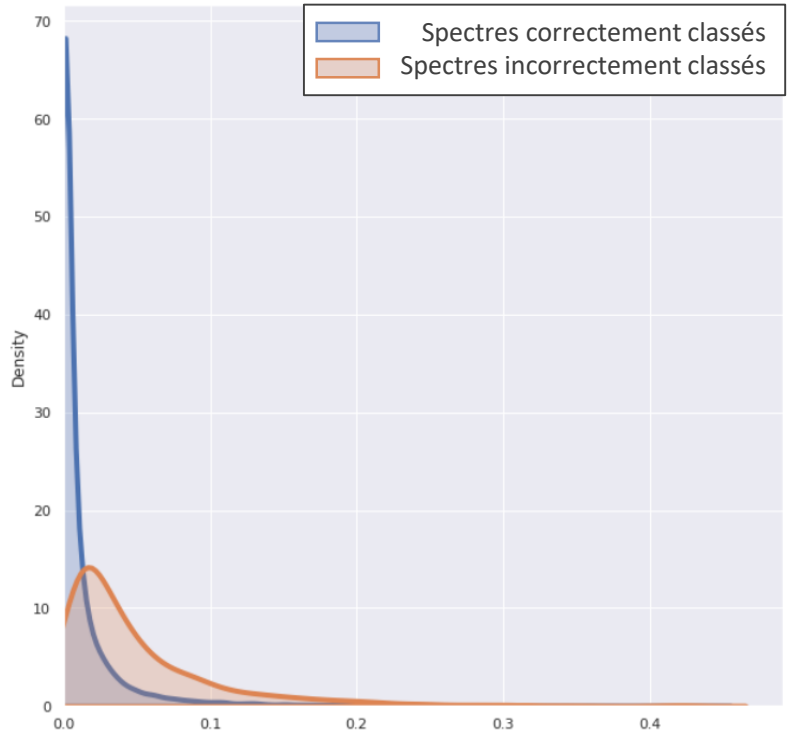
Éléments réellement présents

Éléments réellement absents

Utilisation pour « identifier » les bonnes et les mauvaises prédictions



Moyenne des entropies
= Incertitude aléatoire

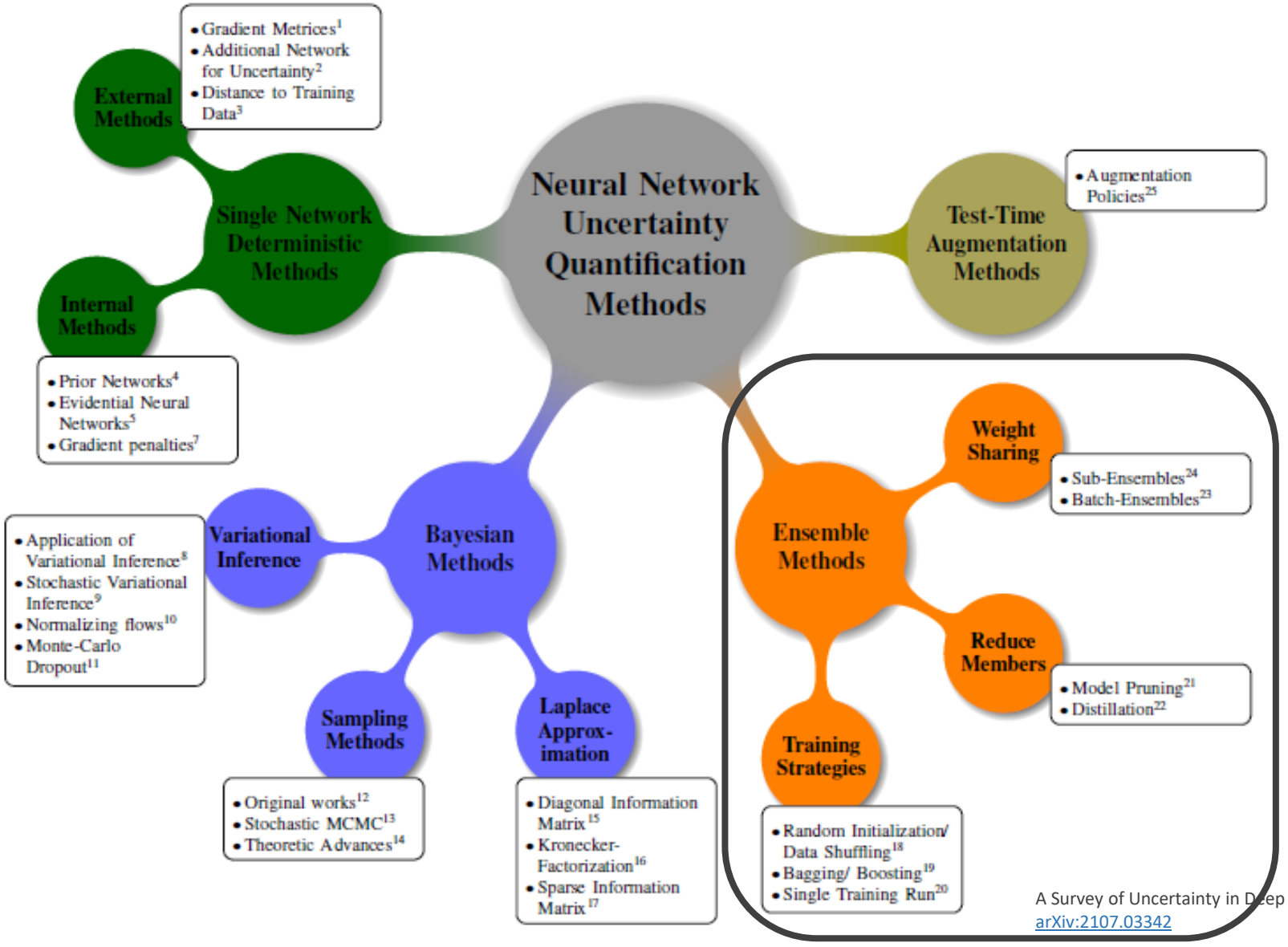


Information mutuelle
= Incertitude épistémique

Mauvaises prédictions : grande incertitude aléatoire
Bonnes prédictions : faible incertitude aléatoire

Mauvaises prédictions : incertitude épistémique légèrement plus élevée que pour les bonnes prédictions

Différentes approches Deep Learning



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Différentes approches Deep Learning

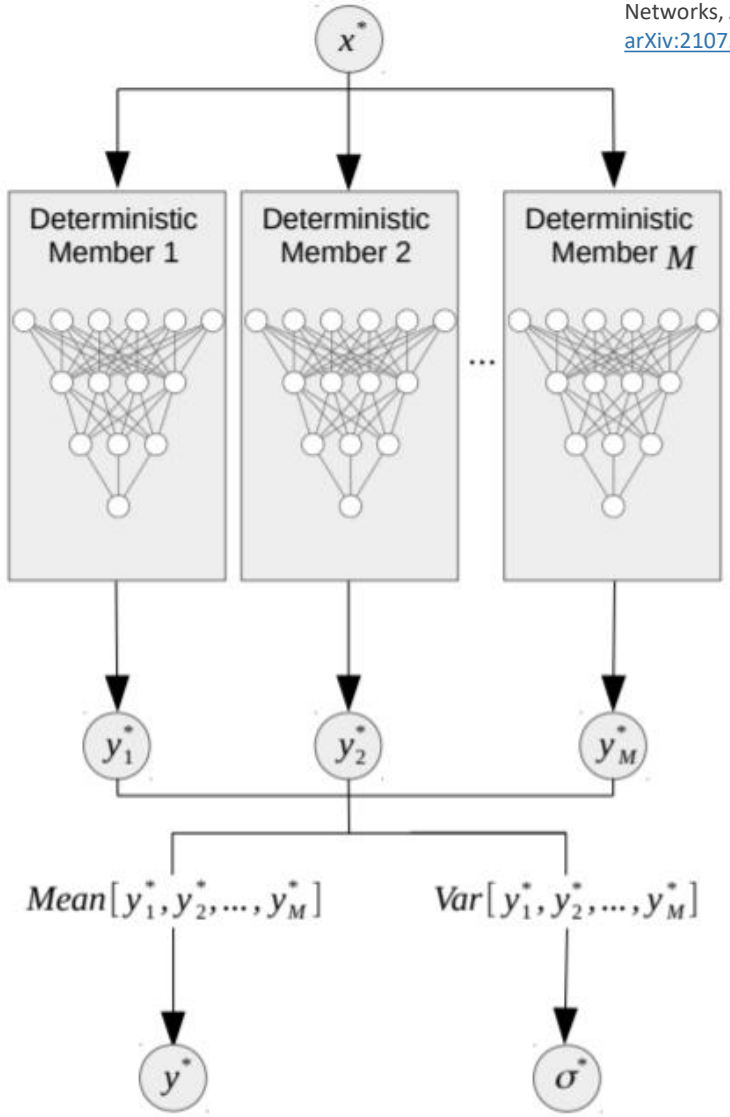
Ensemble methods :

Idée principale : entraîner plusieurs réseaux de neurones pour obtenir une variabilité sur les prédictions

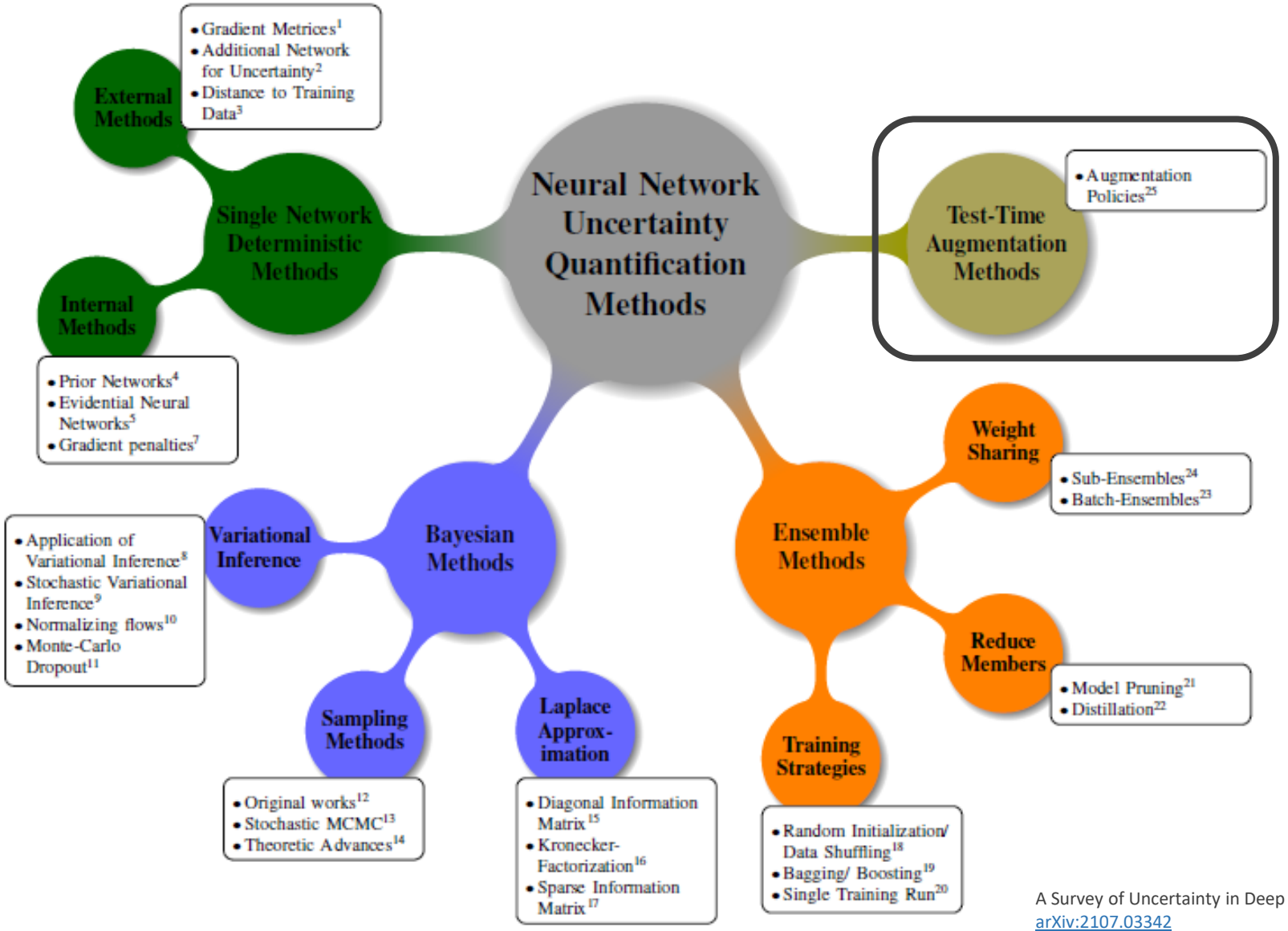
Plusieurs approches :
- Nouvelles initialisations du réseau
- Brassage de la base de données
- Perturbations aléatoires lors de la descente de gradient,

Inconvénients :
- Nécessite l'apprentissage de plusieurs réseaux de neurones
- Stockage de plusieurs modèles

A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342



Différentes approches Deep Learning



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

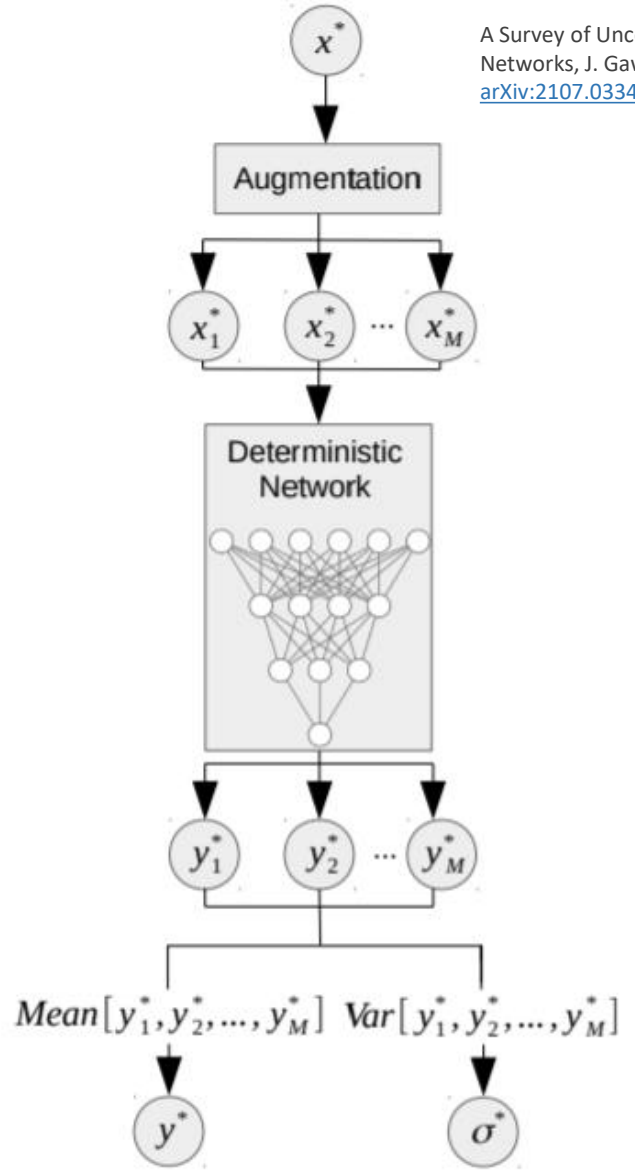
Différentes approches Deep Learning

Test-time augmentation methods:

Idée principale : Déformer et perturber aléatoirement les données de test pour obtenir une variabilité des réponses

Plusieurs types de déformation :
- Bruit gaussien (ou non)
- Pour des images : zoom, rotations, tronquer une partie de l'image…

Inconvénient :
- Choix du niveau de bruit, des déformations
- Réelle représentation des incertitudes ?



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Incertitudes en deep learning

1. Différents types d'incertitude
2. Estimation de ces incertitudes en Deep Learning
3. **Validation des incertitudes**

Questions interactives lors de la présentation : répondre dans l'onglet Sondages

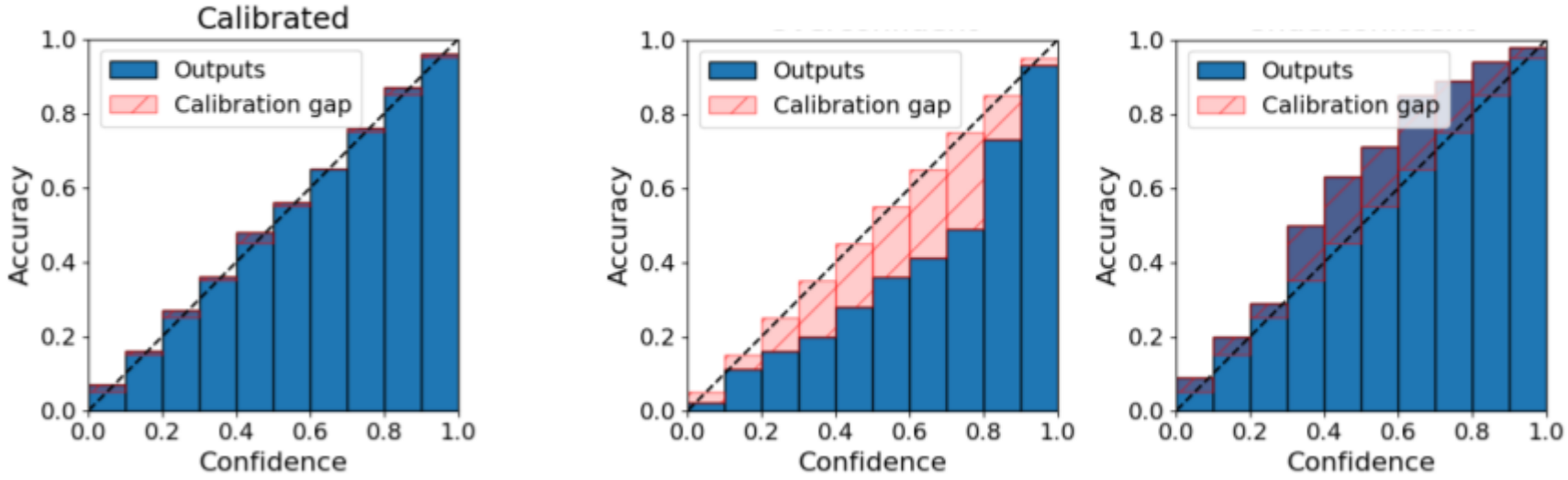
Travaux effectués en collaboration avec Jean-Marc MARTINEZ (DES/ISAS/DM2S/STMF/LGLS)
Ainsi que deux stagiaires : Mohamed Bahi YAHIAOUI (Mines St Etienne) et Clément RIBES (IMT Atlantique)

Courbe de calibration en classification

Idée : Utiliser l'interprétation probabiliste, un réseau de neurones prédit la probabilité qu'une donnée x appartienne à une certaine classe y_i :

$$f_\omega^{(i)} = p\left(y^{(i)}|x\right)$$

Comparaison dans la base de test de la fréquence des réponses correctes aux probabilités prédites :
Exemple : la fréquence des réponses correctes avec une probabilité de 40 % devrait être de 40 %



?

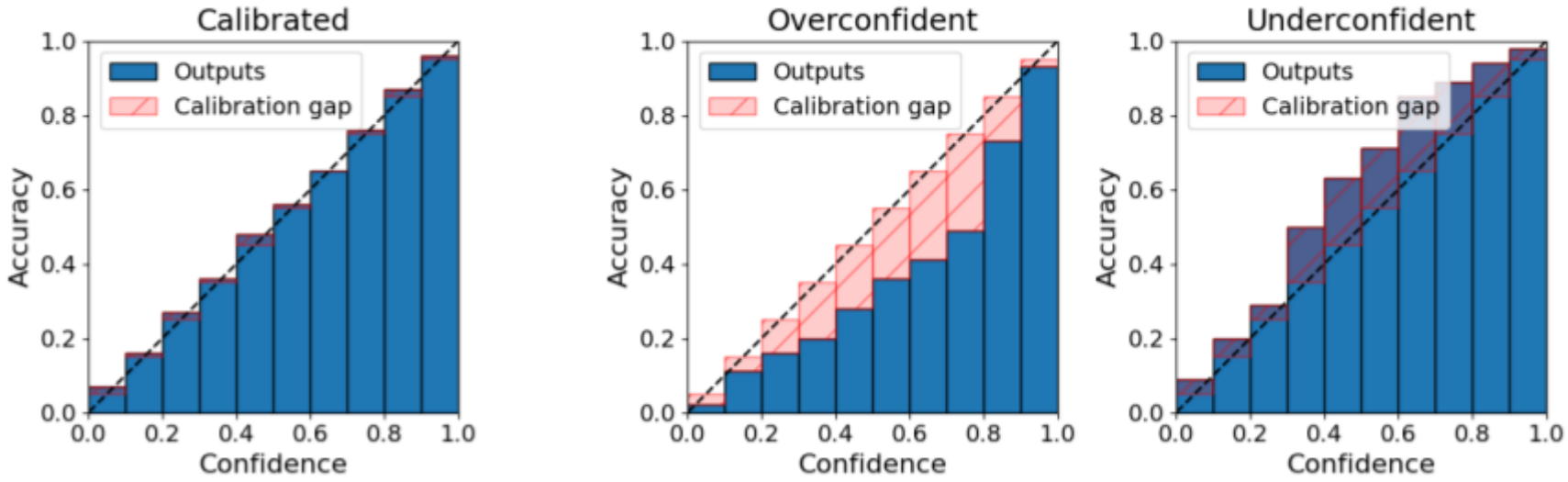
A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Courbe de calibration en classification

Idée : Utiliser l'interprétation probabiliste, un réseau de neurones prédit la probabilité qu'une donnée x appartienne à une certaine classe y_i :

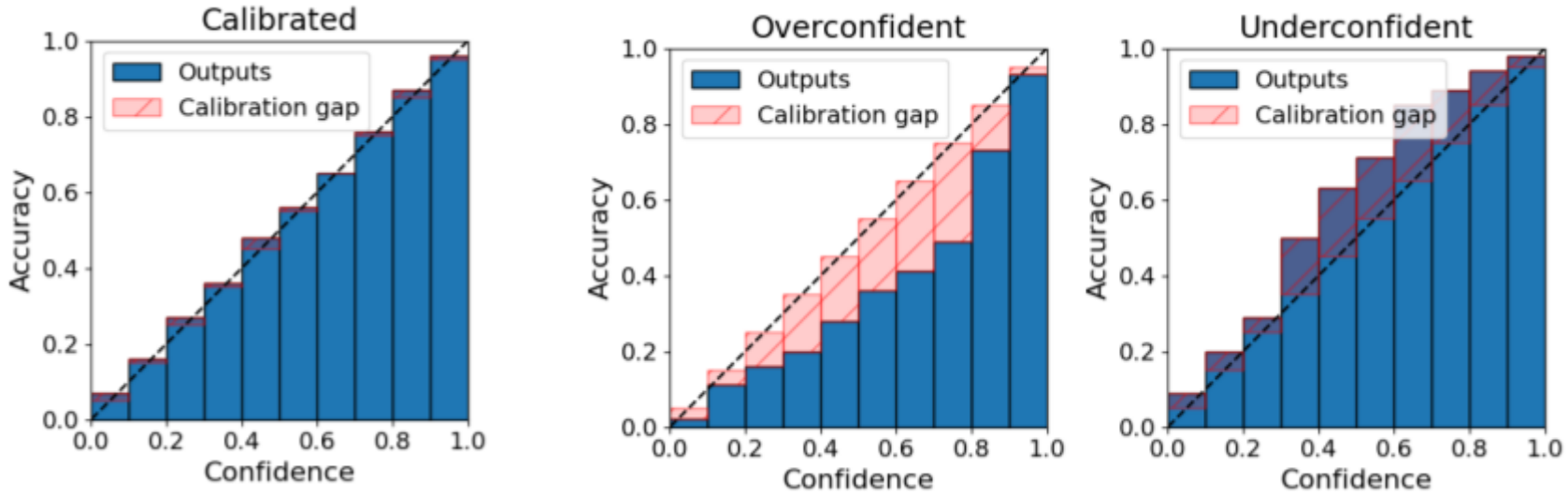
$$f_\omega^{(i)} = p\left(y^{(i)}|x\right)$$

Comparaison dans la base de test de la fréquence des réponses correctes aux probabilités prédites :
Exemple : la fréquence des réponses correctes avec une probabilité de 40 % devrait être de 40 %



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Courbe de calibration en classification



A Survey of Uncertainty in Deep Neural Networks, J. Gawlikowski et al,, arXiv:2107.03342

Techniques de recalibration :

- Méthodes de régularisation lors de la phase d'apprentissage
- Post-processing à l'aide d'un facteur dit de température

$$f_{\omega}^{(i)}(x) = p\left(y^{(i)}|x\right) = \sigma\left(\frac{z^{(i)}}{T}\right)$$

- Calibration de la méthode d'évaluation des incertitudes : taux de Dropout par exemple

Courbe de calibration en régression

Intervalle de prédiction $I_\alpha(x)$ pour une donnée x et une probabilité α :

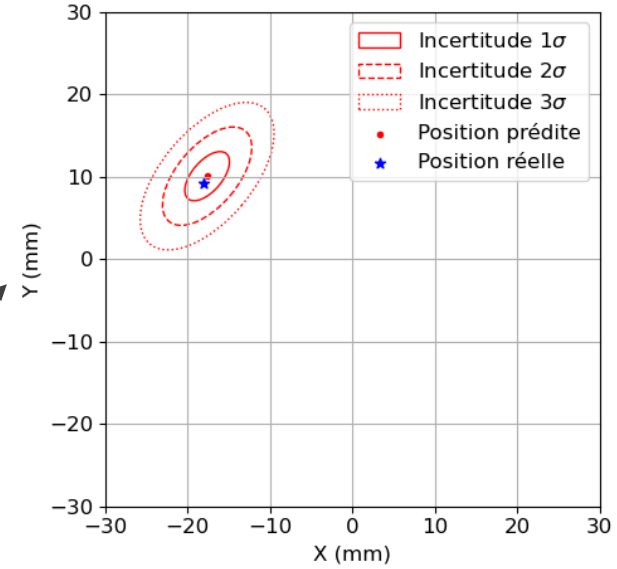
$P\big(I_\alpha(x) \ni y(x)\big) = \alpha$, où $y(x)$ est la valeur de sortie réelle associée à x

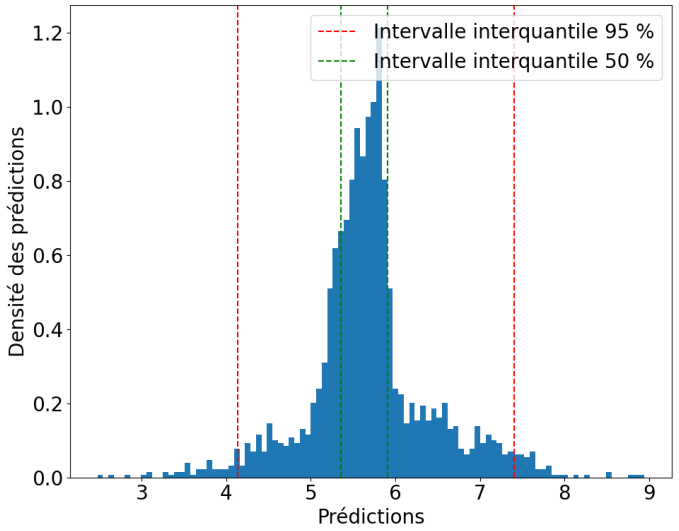
Avec les différentes approches de quantification d'incertitudes, il est possible d'estimer des intervalles de prédiction $\hat{I}_\alpha$:

- Analytiquement dans le cas des Mixture Density Network :

  Estimation de la densité de probabilité $p(y|\theta)$ où $\theta = f_\omega(x)$

- En calculant les intervalles interquantiles sur un échantillon Monte-Carlo des prédictions pour chaque exemple





Courbe de calibration en régression

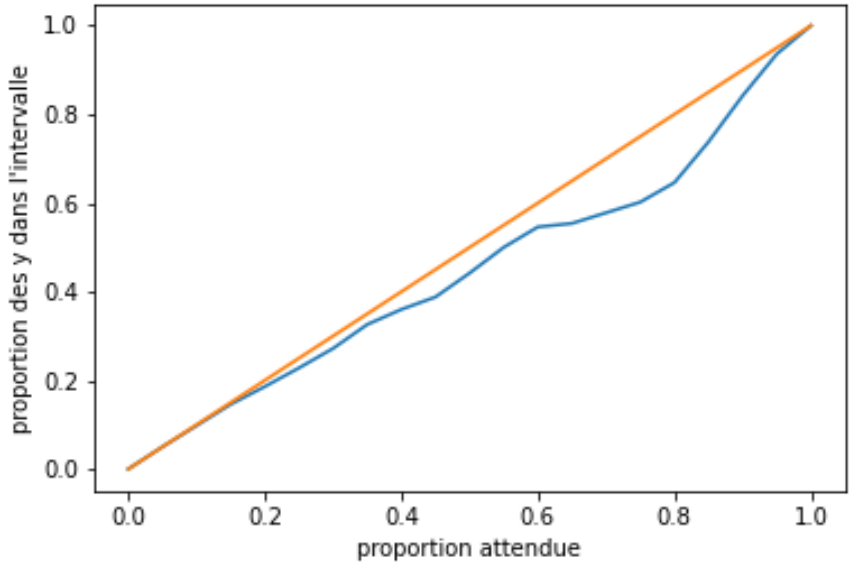
Les intervalles interquantiles $\hat{I}_\alpha$ de probabilité α obtenus sont-ils des intervalles de prédiction I_α de probabilité α ?

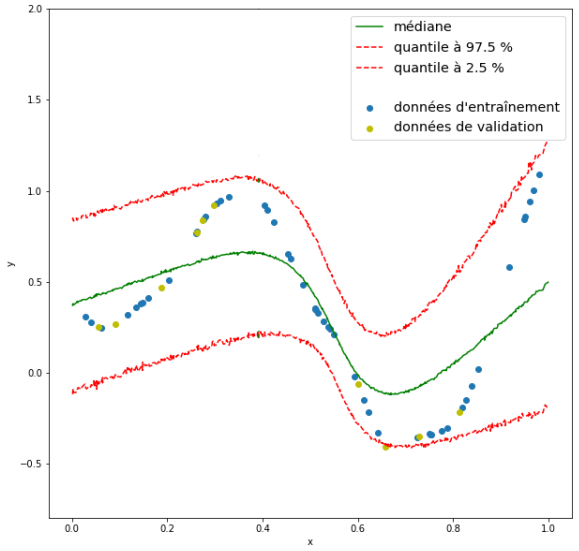
Méthode de vérification : évaluation moyenne sur la base de test :

$$\mathbb{E}_x\left[P\left(\hat{I}_\alpha(x) \ni y(x)\right)\right] = \int_x P\left(\hat{I}_\alpha(x) \ni y(x)\right) p(x)dx \underset{?}{=} \alpha$$

Des incertitudes bien calibrées ne signifient pas qu'un modèle est performant

Courbe de calibration : proportion des données de test dans leur intervalle interquantile $\hat{I}_\alpha$ prédit comparée avec la probabilité attendue α





Recalibration :

- Changement de modèle de Mixture Density Network
- Adaptation des hyperparamètres (taux de Dropout en MC-Dropout, prior sur les réseaux bayésiens…)

Courbe de calibration en régression

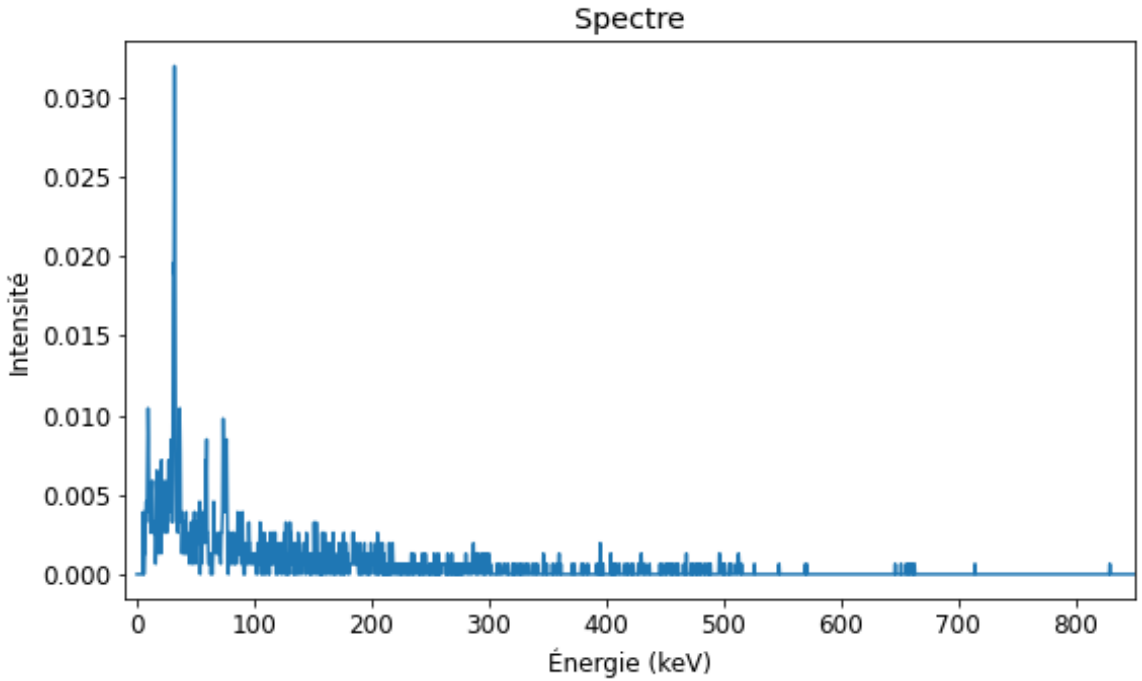
Application : Exemple en spectrométrie gamma

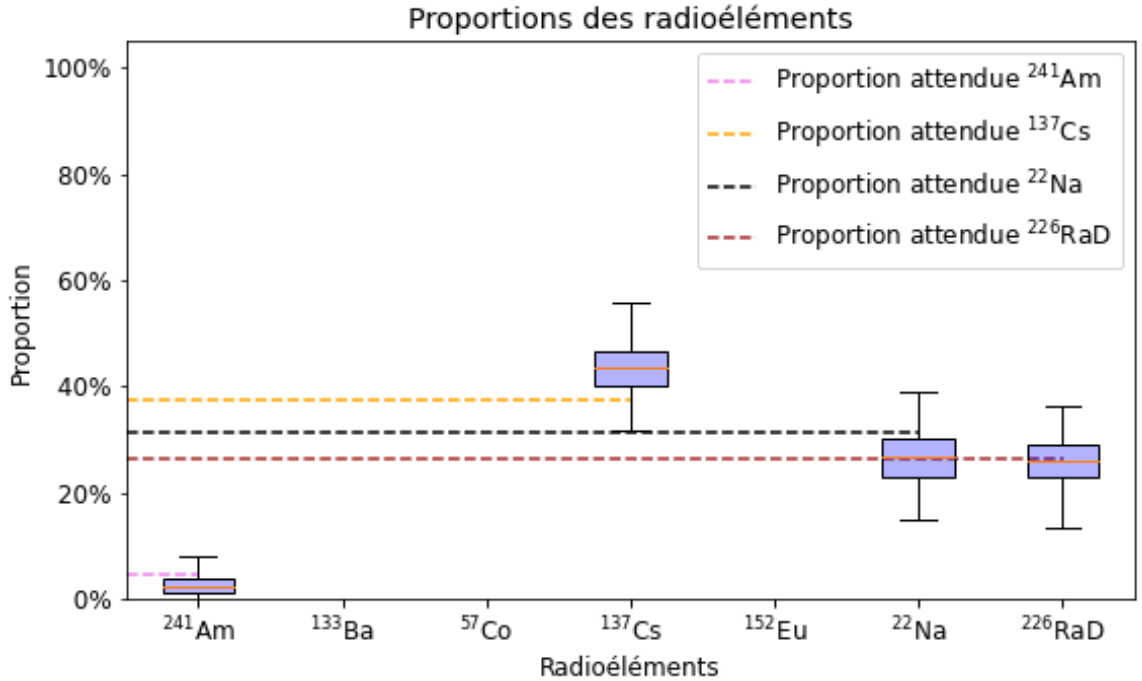
Attention : Ici prédiction des proportions des éléments (et non leur présence)

→ Problème de régression

→ Prédiction avec incertitude par MC Dropout

Exemple :

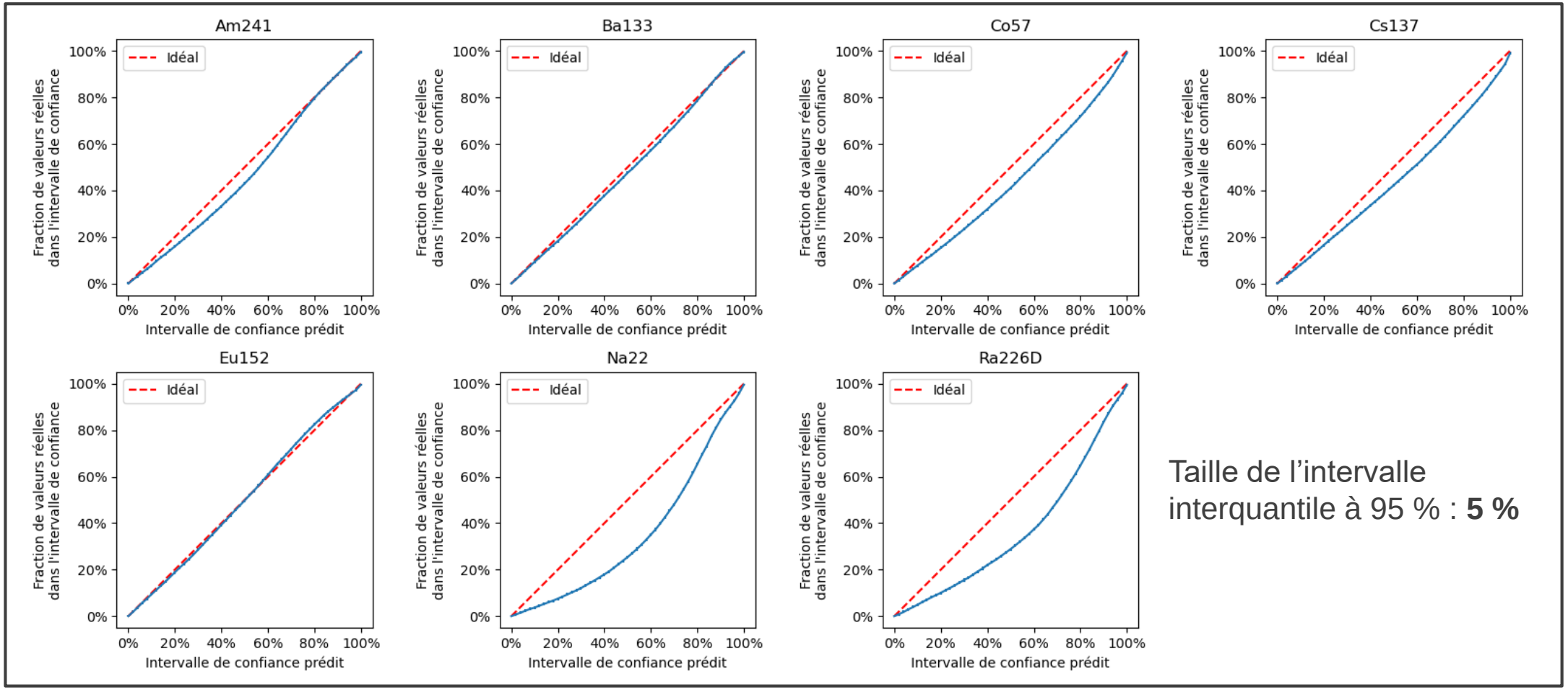




Courbe de calibration en régression

Application : Exemple en spectrométrie gamma

Courbes de calibration avec un taux de Dropout de 0,25



Taille de l'intervalle interquantile à 95 % : **5 %**

Conclusion sur les incertitudes

Méthodes de quantification des incertitudes en Deep Learning :

- Des outils existants et nouveaux dans la littérature
- État de l'art en pleine évolution

Interprétation avec précaution → Nécessité de valider les incertitudes

Mise en œuvre simplifiée avec les bibliothèques actuelles : Tensorflow Probabilities

Sujet d'étude au LGLS

→ Importance à prendre en compte dans les applications scientifiques utilisant les modèles d'apprentissage profond

→ Lien avec les aspects de robustesse (sensibilité des réseaux de neurones à des petites perturbations, travaux avec le PTC ICARE)

Quelques références

Reviews sur les incertitudes en Deep Learning

- *A Survey of Uncertainty in Deep Neural Networks*, J. Gawlikowski et al. (2021)
  arXiv:2107.03342

- *A review of uncertainty quantification in deep learning: Techniques, applications and challenges*, M. Abdar et al. (2020), Information Fusion
  https://doi.org/10.1016/j.inffus.2021.05.008

Mixture Density Network

- *Mixture Density Networks*, Christopher M. Bishop (1994), NCRG/94/004
  https://publications.aston.ac.uk/id/eprint/373/1/NCRG_94_004.pdf

Bayes by backprop

- *Weight Uncertainty in Neural Networks*, Blundell et al. (2015), Proceedings of the 32 nd International Conference on Machine Learning
  arXiv:1505.05424v2

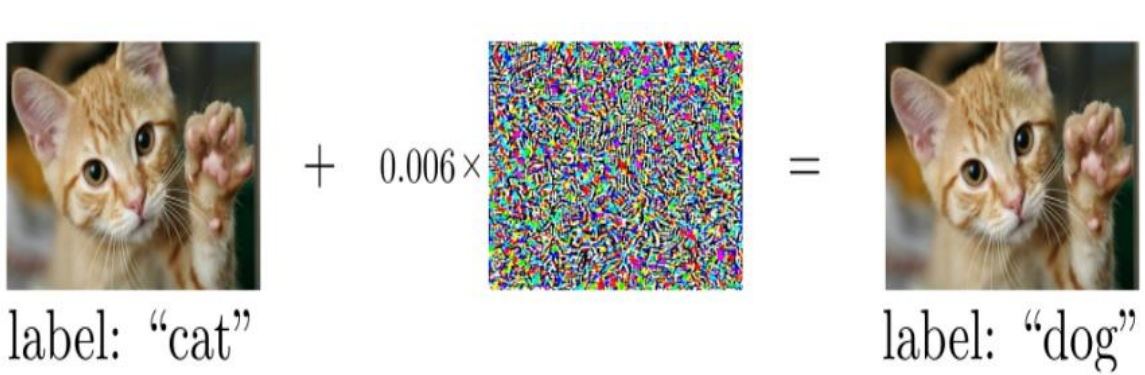
Monte-Carlo Dropout

- *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,* Gal & Ghahramani (2016), Proceedings of the 33 rd International Conference on Machine Learning
  arXiv:1506.02142v6

Robustesse d'un réseau : caractérise l'insensibilité d'un réseau à des petites perturbations calculées spécifiquement pour le tromper

Dans la littérature : grande sensibilité des réseaux « classiques » à ces petites perturbations → principalement lié à la « grande dimension »

Exemples mignons

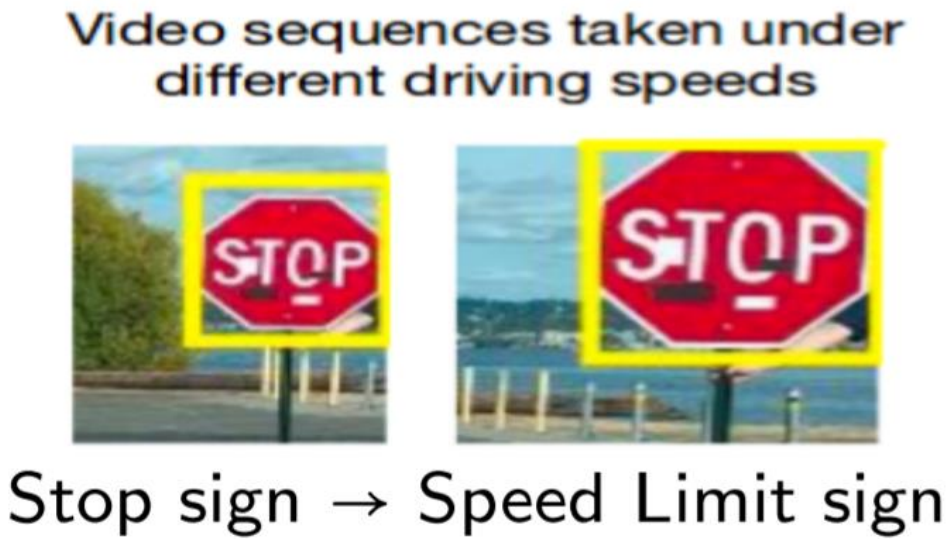


classified as pig (91% confidence) | classified as airliner (99% confidence) | (difference between the two images, /0.005)

Robustesse d'un réseau : caractérise l'insensibilité d'un réseau à des petites perturbations calculées spécifiquement pour le tromper

Dans la littérature : grande sensibilité des réseaux « classiques » à ces petites perturbations → principalement lié à la « grande dimension »

Exemple moins mignon



→ Nécessité d'étudier ce sujet
→ Mettre en place des méthodes de défense

Comment calcule-t-on ces perturbations adverses ?

Prenons un réseau de neurones déjà entraîné, f_ω, une donnée d'entrée x, son label associé y et une fonction de coût l

Trouver une perturbation optimale δ^* qui trompe le réseau revient à résoudre le problème suivant :

$$\delta^* = \underset{\delta \in B}{\operatorname{argmax}}\, l(f_\omega(x+\delta), y)$$

B représente une certaine contrainte que la perturbation doit respecter, de sorte à ce qu'elle soit considérée comme petite ou imperceptible.
Exemples : $\|\delta\|_2 \leq \varepsilon$, $\|\delta\|_\infty \leq \varepsilon$, ou encore d'autres contraintes spécifiques au problème

Approche « boîte blanche »

Supposons que la structure du réseau est connue ainsi que ses paramètres ω. On peut résoudre le problème d'optimisation ci-dessous par remontée de gradient respectivement aux entrées :

$$\delta \propto \nabla_x\, l(f_\omega(x), y)$$

Gradient calculable par backpropagation en remontant jusqu'à l'entrée du réseau
Approche « one-shot » (FGSM), par projection pour respecter les contraintes (PGD), itérative (BIM)

Comment calcule-t-on ces perturbations adverses ?

Prenons un réseau de neurones déjà entraîné, f_ω, une donnée d'entrée x, son label associé y et une fonction de coût l

Trouver une perturbation optimale δ^* qui trompe le réseau revient à résoudre le problème suivant :

$$\delta^* = \underset{\delta \in B}{\operatorname{argmax}}\, l(f_\omega(x+\delta), y)$$

B représente une certaine contrainte que la perturbation doit respecter, de sorte à ce qu'elle soit considérée comme petite ou imperceptible.
Exemples : $\|\delta\|_2 \leq \varepsilon$, $\|\delta\|_\infty \leq \varepsilon$, ou encore d'autres contraintes spécifiques au problème

Approche « boîte noire »

On ne connaît pas la structure ou les paramètres du réseau :

- On peut entraîner un autre réseau connu sur le problème et calculer ses adversaires par une méthode boîte blanche : il y a une probabilité plus élevée que les adversaires d'un autre réseau soient aussi des adversaires pour le réseau cible
- Utilisation d'algorithmes de type évolutionnaire (génétique) pour la recherche de la perturbation optimale

Comment rendre le réseau plus robuste ?

Adversarial learning

Idée générale : au cours de l'apprentissage, on calcule les adversaires sur la base de données à chaque itération et on les intègre au jeu d'apprentissage → s'approche d'une procédure de data-augmentation

Mathématiquement, cela revient à résoudre le problème min-max suivant pour la recherche des paramètres optimaux ω du réseau :

$$\min_{\omega} \max_{\delta \in B} l(f_{\omega}(x+\delta), y)$$

Alourdit la phase d'apprentissage mais montre une efficacité à rendre le réseau plus robuste.
Reste a priori dépendant de la contrainte B choisie.

Approche par régularisation

Idée générale : le réseau est sensible aux petites perturbations car il est peu régulier, il existe une direction qui peut perturber grandement sa réponse, donnée par son gradient par rapport aux entrées. On va chercher à minimiser ce gradient.
Par exemple : ajout dans la fonction de coût d'un terme de pénalisation :

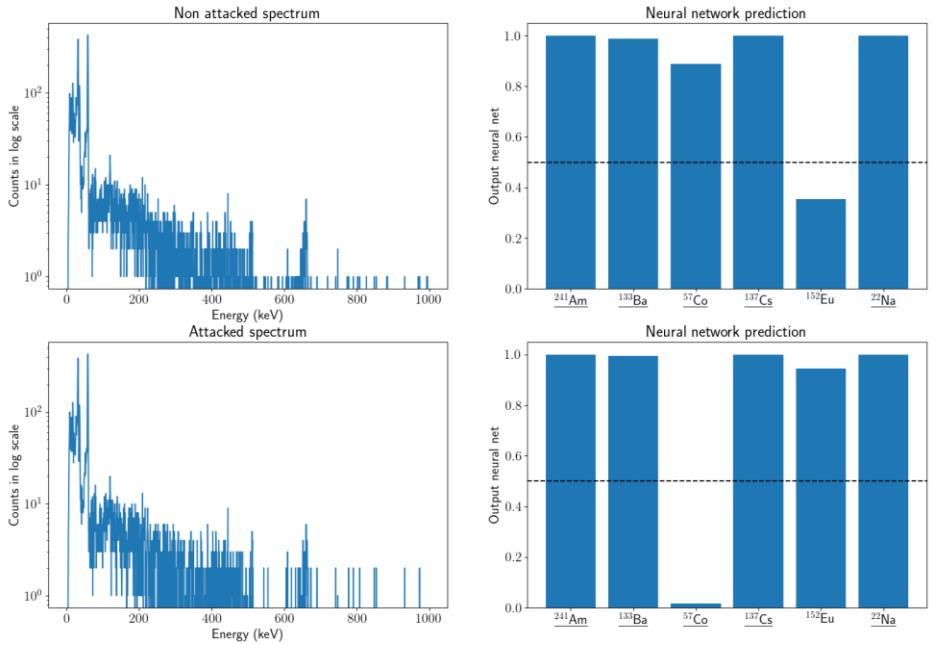
$$L_{tot}(f_{\omega}(x), y) = l(f_{\omega}(x), y) + \lambda \left\| \nabla_x \left(f_{\omega}(x) \right) \right\|$$

Généralisation aux dimensions multiples (> 1) sur les sorties en utilisant la jabobienne au lieu du gradient.

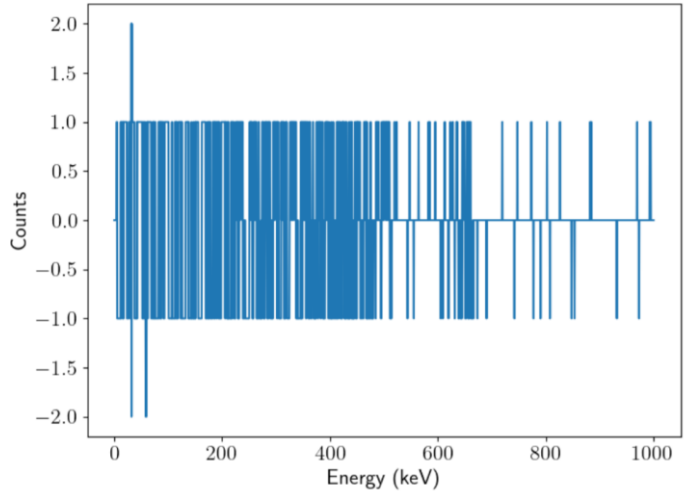
Application en spectrométrie gamma

Recherche des adversaires par descente de gradient.
Définition des contraintes spécifiques au problème : une perturbation est petite si elle peut être « confondue » avec des fluctuations statistiques de Poisson :

- Sur chaque bin i du spectre qui contient initialement N_i photons : $|\delta_i| \leq \sqrt{N_i}$ (1σ sur la loi de Poisson)
- Au total sur le spectre : $|\sum_i \delta_i| \leq \sqrt{\sum_i N_i}$ contrainte sur le nombre de photons détectés au total (1σ sur la loi de Poisson)
- La perturbation doit être un nombre entier (on parle d'un nombre de photons) : $\delta_i \in \mathbb{Z}$

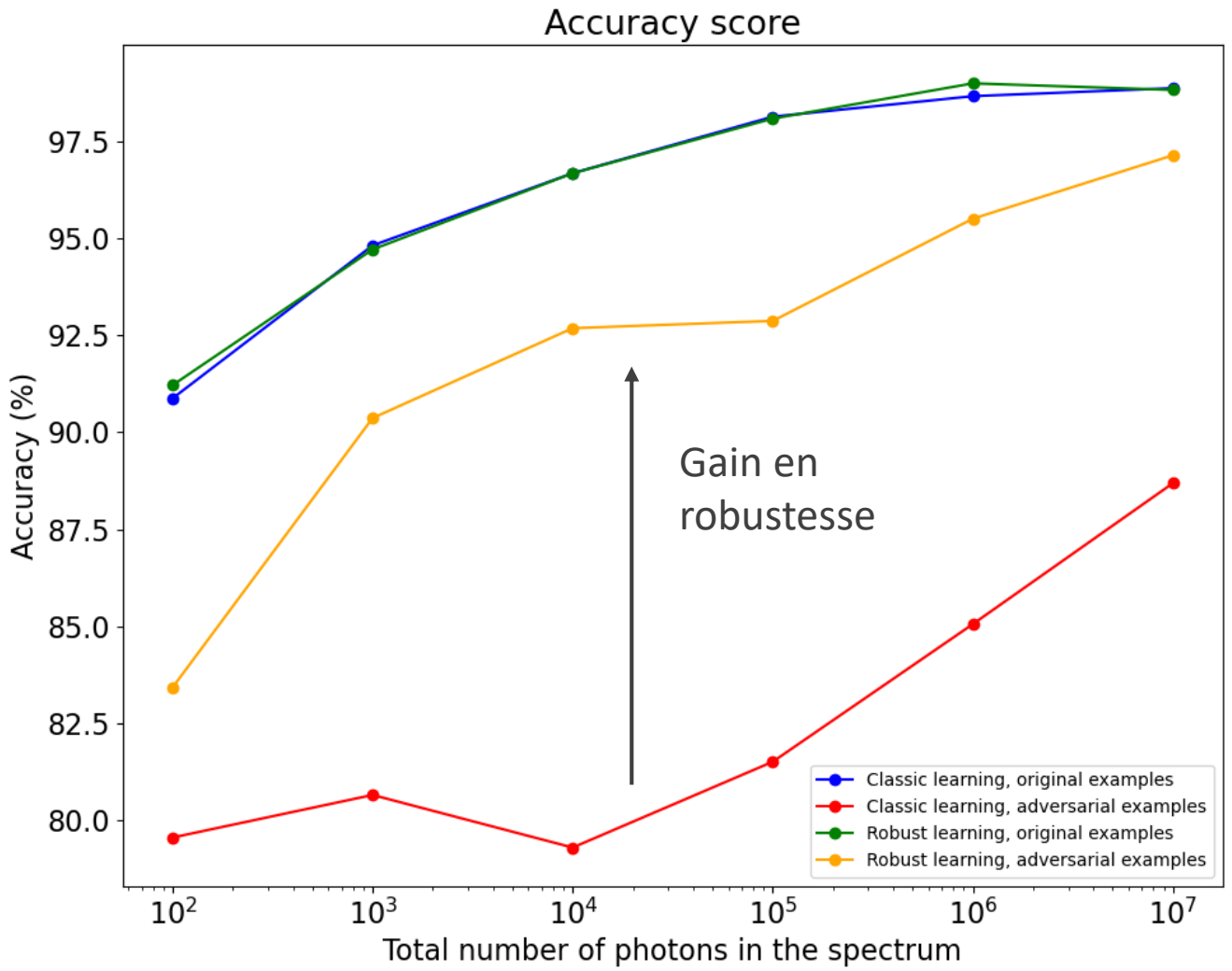


Différence entre les deux spectres = Perturbation δ



Application en spectrométrie gamma

Application de l'apprentissage robuste : impact sur l'accuracy du réseau



Conclusion

Robustesse des réseaux de neurones : problématique à considérer, notamment dans des applications critiques

Des méthodes existent pour rendre les réseaux de neurones plus robustes

Lien avec la régularité des réseaux de neurones et avec les données en grande dimension

Le problème existe aussi, sous une autre forme, en régression

→ Les réseaux de neurones se comportent comme des fonctions peu régulières dans certaines directions

→ Méthodes de **certification** pour être capable de borner ces « irrégularités »

Références sur la robustesse des réseaux

A. Kurakin, I. Goodfellow, and S. Bengio. *Adversarial examples in the physical world*. arXiv preprint arXiv:1607.02533, 2016.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, arXiv preprint arXiv:1706.06083, 2017.

S.M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, *Deepfool: A simple and accurate method to fool deep neural networks*. 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582, 2016.