Résolution d'une équation différentielle pour la calibration du détecteur ATLAS via un réseau de neurones

Master Physique Recherche Fondamentale

Matthieu PÉLISSIER

Encadré par Pierre-Antoine DELSART

Abstract

La physique des particules étudiée au sein du LHC par le détecteur ATLAS requiert des mesures d'une grande précision. Les grandeurs énergétiques liées aux objets "jet hadroniques" ne peuvent être utilisées directement à partir des mesures expérimentales et doivent être corrigées pour pouvoir être comparées aux prédictions théoriques. Cette étude aborde ce problème sous la forme d'une équation différentielle non linéaire du second ordre sur la fonction de calibration, qui ne possède pas de solution analytique. Un réseau de neurones est utilisé pour la résoudre. L'objectif est d'obtenir une fonction de calibration précise en exploitant les capacités non linéaires des réseaux de neurones.

I Contextualisation

Ce stage portait majoritairement sur la création et utilisation du réseau de neurones. Il s'inscrit dans un contexte physique important : l'étude du modèle standard de la physique des particules et de potentielles traces de nouvelle physique non prédite par ce modèle. Cette étude a lieu au collisionneur de protons LHC (Large Hadronic Collider), avec le détecteur ATLAS.

I.1 Nécessité de calibration du détecteur ATLAS

Le détecteur ATLAS mesure la position et l'énergie déposée par les particules. L'acquisition de ces données permet une étude précise des jets de particules hadroniques, phénomène QCD résultant de l'émission de quarks ou de gluons lors d'une collision au sein du détecteur. L'analyse de ces données peut amener à la découverte de nouvelles particules, comme le boson de Higgs, présenté dans [col12]. Pour cela, il est nécessaire d'effectuer des mesures précises en énergie pour estimer au mieux la masse de la particule étudiée. Cependant, en raison du caractère intrinsèquement probabiliste des phénomènes quantiques lors de la formation des jets et de leurs interaction avec la matière du détecteur, il n'est pas possible d'atteindre une précision parfaite dans les mesures.

L'énergie mesurée $E_{recovered} = E_r$ ne correspond pas nécessairement à l'énergie réelle du processus. Pour étudier les processus à leur énergie réelle, les énergies mesurées sont calibrées avec la fonction de calibration $u(E_r) = E_c$. Il est primordial d'effectuer cette calibration avec précision pour pouvoir effectuer correctement l'analyse statistique des résultats.

Plus de détails sont donnés dans Ap. A.

I.2 Formulation mathématique

Pour un processus, les énergies E_r mesurées suivent une densité de probabilité $f(E_r, E_t)$, maximale en E_*. Ces mesures correspondent à un processus se déroulant à une énergie réelle la plus probable $E_{true} = E_t$, pouvant atteindre des énergies E_c différentes de ce maximum. Celles-ci suivent une fonction de densité de probabilité $g(E_c, E_t)$, maximale en E_t.

L'objectif est alors d'établir une relation entre les énergies mesurées et les énergies calibrées en utilisant la fonction de calibration $u(E_r) = E_c$. Aussi, l'égalité $u(E_*) = E_t$ n'est pas nécessairement satisfaite.

Il est possible de relier les deux distributions de probabilité et d'utiliser la définition de E_t pour obtenir le système d'équations suivant :

$$\begin{cases} g(E_c, E_t) = f(u^{-1}(E_c), E_t)(u^{-1})'(E_c) \\ \frac{dg(E_c)}{dE_c}(E_t, E_c = E_t) = 0 \end{cases} \tag{1}$$

Pour un grand nombre de mesures, la densité de probabilité des énergies mesurées peut être approximée à une gaussienne :

$$f(E_r, E_t) = C_{norm} e^{-\frac{1}{2\sigma(E_t)^2}(u^{-1}(E_c) - E_t R(E_t))^2}. \tag{2}$$

Dans Eq. 2, $R(E_t) = \frac{E_*}{E_t}$ est appelée réponse, et relie le maximum des distributions des énergies mesurées et calibrées. Re-injecter Eq. 2 dans le système Eq. 1 permet

d'obtenir une équation différentielle non linéaire du second ordre sur la fonction de calibration :

$$u''(E)\sigma(E)^2 + u'(E)(x - u(E)R(u(E))) = 0 \qquad (3)$$

Il n'existe pas de solution analytique à cette équation, mais sa résolution permettrait d'obtenir une fonction de calibration exacte, et donc une excellente précision. Sa démonstration est donnée dans Ap. B.

II Utilisation d'un réseau de neurones

Jusqu'à présent, la collaboration ATLAS utilise des méthodes d'inversion numériques pour obtenir l'énergie calibrée E_c. Cependant, ces méthodes sont valides pour des fonctions de calibration $u(E_r) = E_c$ linéaires, ce qui n'est pas le cas à basse énergie, et qui peut donc être problématique. Les réseaux de neurones offrent la possibilité d'obtenir une réponse non linéaire grâce aux fonctions d'activation, améliorant ainsi la précision de la calibration. Leur utilisation est également en phase avec les méthodes récemment développées pour divers aspects du détecteur ATLAS. L'implémentation de la calibration via ces outils numérique permettrait ainsi d'homogénéiser la chaine d'analyse des données. Une introduction sur le fonctionnement général d'un réseau de neurones est donné dans Ap. C.

Les réseaux de neurones sont utiles pour résoudre des équations différentielles de type $\psi(x, u, u', u'') = 0$. Le réseau prend un x en entrée, constitué de plusieurs couches de perceptrons interconnectés. Ce réseau fournit une prédiction de la valeur de la fonction recherchée $u(x)$ à ce point. La fonction *loss* à minimiser, utilisée pour ajuster les paramètres du réseau par descente de gradient, permet de chercher une prédiction satisfaisant à l'équation différentielle. La *loss* est composée de deux termes :

$$loss = (\psi(x, u, u', u''))^2 + \sum_i (u(x_i) - u_i)^2 \qquad (4)$$

Le premier terme est l'équation différentielle au carré, pris au point x étudié. Si ce terme parvient à son minimum, i.e. 0, alors u est solution de l'équation différentielle. Le second terme de l'équation permet d'obtenir une solution unique à l'équation différentielle, en utilisant des conditions initiales choisies au préalable (x_i, u_i). Ces dernières peuvent porter sur la fonction u ou sa dérivée. Il compare la prédiction du réseau $u(x_i)$ (ou $u'(x_i)$) en des point x_i, avec la valeur théorique u_i (ou u'_i) choisie au préalable. Lorsque le minimum de ce terme, de nouveau 0, est atteint, alors u satisfait aux conditions initiales. Lorsque la *loss* atteint son minimum, u est une solution unique à l'équation différentielle.

Pour implémenter cette méthode, la fonction `"AUTOGRAD"` de la librairie `PYTORCH` permet de calculer la dérivée de la sortie du réseau, ici $u(x)$, par rapport à son entrée x. Les valeurs de u' et u'', et donc la *loss* Eq. 4, sont calculées par ce procédé. Des exemples et détails sont fournis dans Ap. D. Le sujet de cette étude est d'appliquer cette méthode à Eq. 3, pour obtenir la fonction de calibration.

III Résultats

Une partie majeure de ce projet fut de coder l'implémentation de cette méthode, puis de tester son efficacité pour des équations différentielles linéaires et non linéaires du premier et second ordre. La sensibilité aux conditions initiales a également constitué un sujet d'étude important. Il a ensuite été nécessaire de chercher à améliorer la convergence de la *loss* vers son minimum, en améliorant divers aspects du réseau. Nous avons pu appliquer notre méthode à plusieurs cas simplifiés de Eq. 3 en choisissant des fonctions $\sigma(E)$ et $R(E)$ appropriées. Les résultats présentés dans la suite constituent l'aboutissement de ces recherches, appliquées à Eq. 3 pour des cas plus complexes.

III.1 Réponse en énergie

Expérimentalement, la réponse en énergie et l'écart type suivent les formes suivantes :

$$R(x) = \alpha\left(1 - \frac{1}{(\beta + x)^9}\right). \qquad (5)$$

$$\sigma_{norm} = Ax + b \qquad (6)$$

où α, β, A et b sont des constantes déterminées expérimentalement. x correspond à l'énergie normalisée E/E_{norm}. Son utilisation facilite le travail au réseau de neurones. Les définitions de ces fonctions permettent de définir l'équation différentielle à résoudre pour trouver u.

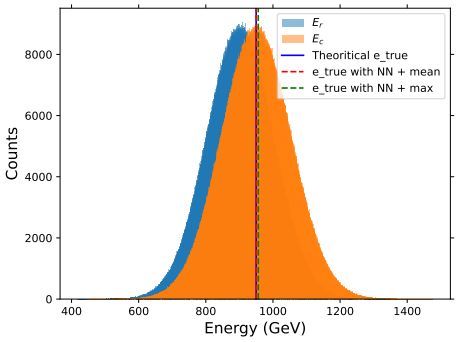
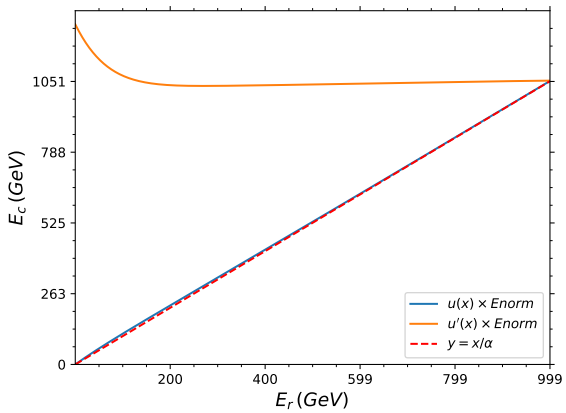


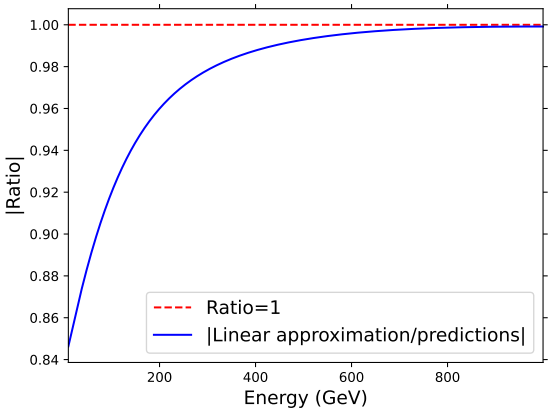
FIGURE 1 – Exemple de génération et calibration d'une gaussienne. La gaussienne est générée pour $E_t = 950GeV$. La distribution orange correspond à la distribution après calibration, où sa moyenne, son maximum et l'énergie E_t théorique sont repérés par les lignes verticales.

Il est nécessaire de vérifier si la solution obtenue par le réseau de neurones correspond bien à la fonction de calibration. Pour cela, il est possible de générer des énergies mesurées E_r suivant des gaussiennes par Monte-Carlo, pour un ensemble de E_t connu, centrées en $E_* = R \times E_t$, et d'écart type $\sigma(E_t)$. Les gaussiennes obtenues sont données en entrée du réseau pré-entrainé, pour calibrer les énergies mesurées, obtenant donc pour chaque E_t la distribution $g(E_c, E_t)$. Fig. 1 illustre un résultat obtenu par cette méthode pour $E_t = 950\ GeV$. La comparaison entre la valeur à laquelle la distribution est maximale E_{max}, obtenue par le réseau, avec celle théorique E_t, renseigne sur la qualité

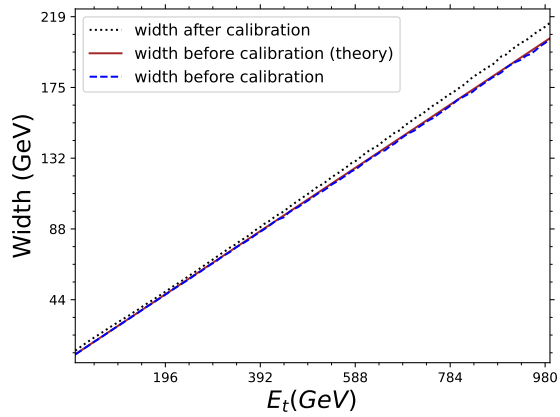
de la calibration. Si la réponse calibrée $R_{calib} = E_{max}/E_t$ est égale à 1, alors la fonction u obtenue par le réseau de neurones calibre correctement en énergie, et est donc la fonction attendue.



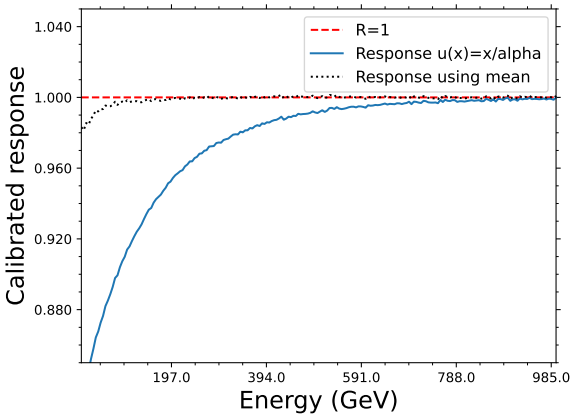
(a) Fonction de calibration



(b) $|u_{\text{linéaire}}/u_{\text{RN}}|$



(c) Largeur de distribution



(d) Réponse calibrée

FIGURE 2 – Vérification de la convergence du réseau de neurones. 20a présente la fonction de calibration par rapport à l'énergie normalisée. 15b est la valeur absolue du ratio de la fonction de calibration approximée linéaire $u(x) = x/\alpha$ par celle obtenue en sortie du réseau de neurones. 15c illustre l'évolution de la largeur de la distribution ($\simeq 2\sigma$), avant et après la calibration. 15d présente la réponse en énergie une fois calibrée, via $u(x) = x/\alpha$ et celle obtenue par le réseau de neurones. Les résultats sont présentés pour des énergies E_{true} comprises entre 10 et 1000 GeV.

Fig. 15 présente les résultats obtenus pour la réponse et écart type en énergie. Fig. 20a et Fig. 15b présentent la fonction obtenue en sortie du réseau de neurones comme étant linéaire à haute énergie, et différant légèrement de l'approximation linéaire à basse énergie. Bien que faible, cette différence a un impact important dans la qualité de la calibration. En effet, en utilisant la fonction obtenue en résolvant l'équation différentielle, R_{calib} reste près de $R = 1$ à 2% près, ce qui n'est pas le cas en utilisant la fonction linéaire (Fig. 15d).
Dans Fig. 15c, la largeur de la distribution des énergies calibrées est plus grande que celle des énergies mesurées. En effet, la fonction de calibration reste supérieure à la droite $y = x$, et cet écart augmente pour les hautes énergies. Ainsi, les distributions sont étirées lors de la calibration.

Le réseau de neurones parvient ainsi à résoudre l'équation différentielle assez précisément pour pouvoir calibrer des énergies mesurées, et ce, pour un cas fidèle à l'expérience. Des résultats avec la même précision, pour une autre forme de réponse et d'écart-type, ont pu être obtenus en utilisant la même méthode, comme présentés dans Ap. F.4.

IV Conclusion

En formalisant le problème de la fonction de calibration en une équation différentielle, nous avons pu tester l'approche des réseaux de neurones pour sa résolution. Nous parvenons à résoudre avec précision cette équation, et ainsi calibrer les énergies pour des cas réalistes d'un point de vue expérimental. Cette résolution permet d'obtenir une fonction non linéaire, et donc augmenter la précision de la calibration.

Il serait intéressant de comparer quantitativement cette méthode à celles utilisées actuellement. Si besoin, il faudra également intégrer cet algorithme à la chaine d'analyse existante utilisant des réseaux de neurones.

Références

[col12] ATLAS COLLABORATION. « Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC ». In : *Physics Letters B* 716.1 (sept. 2012), p. 1-29. DOI : 10.1016/j.physletb.2012.08.020.

[Col20] CMS COLLABORATION. « A Measurement of the Higgs Boson Mass in the Diphoton Decay Channel ». In : *Physics Letters B* 805 (juin 2020), p. 135425. ISSN : 03702693. DOI : 10.1016/j.physletb.2020.135425. arXiv : 2002.06398 [hep-ex].

[Dub+21] Shiv Ram DUBEY et al. *diffGrad : An Optimization Method for Convolutional Neural Networks*. 2021. arXiv : 1909.11015 [cs.LG].

[Nov20] Mykola NOVIK. *torch-optimizer – collection of optimization algorithms for PyTorch*. Version 1.0.1. Jan. 2020.

A Contexte physique

A.1 Le modèle standard et ses limites

Le modèle standard répertorie toutes les particules élémentaires connues, en décrivant leurs interactions via l'interaction forte, faible et électromagnétique.
La symétrie électrofaible, qui implique une masse nulle pour les bosons Z et W, est brisée par le de Higgs. Son existence est confirmée expérimentalement en 2012 au LHC ([col12]). Cette découverte renforce la crédibilité du modèle.
Certains éléments du MS restent incomplets ou inexpliqués, comme le fait qu'il n'y ait que 3 familles, l'absence du graviton, l'asymétrie matière - antimatière, le reste du contenu énergétique de l'univers (Matière et énergie noire). La recherche actuelle tente notamment de trouver les limites du modèle, dans l'espoir de pouvoir constituer les bases d'une théorie plus globale pour la physique des particules.

A.2 Expérience ATLAS

L'expérience ATLAS (A Toroidal LHC ApparatuS) est l'un des 8 détecteurs du LHC. Ce dernier est un collisionneur hadronique, permettant des collisions proton-proton jusqu'à 14 TeV dans le centre de masse, testant donc la théorie du modèle standard à de très hautes énergies.
Lors de la collision p-p l'interaction forte domine, et beaucoup de hardons sont créés à ces énergies. Les gluons formés ne peuvent pas rester isolés à cause du confinement de la couleur, et échangent donc de l'énergie avec le vide (incertitude de Heisenberg) pour former d'autre hardons. On parle de gerbes hadroniques. Ces gerbes sont particulièrement intéressantes, puisque détecter les particules produites renseignent sur l'interaction et particules à l'origine de celles-ci.

À cause de l'incertitude quantique, on ne sait jamais exactement le processus qu'il y a eu lors de l'interaction. On doit donc effectuer une analyse statistique, pour extraire la probabilité qu'un processus ait lieu. C'est pourquoi on effectue un grand nombre de collisions, près de 40 MHz au LHC. Des paquets de 10^{11} protons sont envoyés, dont seul une partie interagie au niveau du détecteur. Le LHC permet d'atteindre une luminosité de $10^{34} cm^{-2} s^{-1}$, correspondant au taux d'évènement par section efficace. Des mesures sont effectuées avec une fréquence de 1 kHz, en appliquant un tri automatique sur la sortie diminuant ainsi l'énorme quantité de datas.

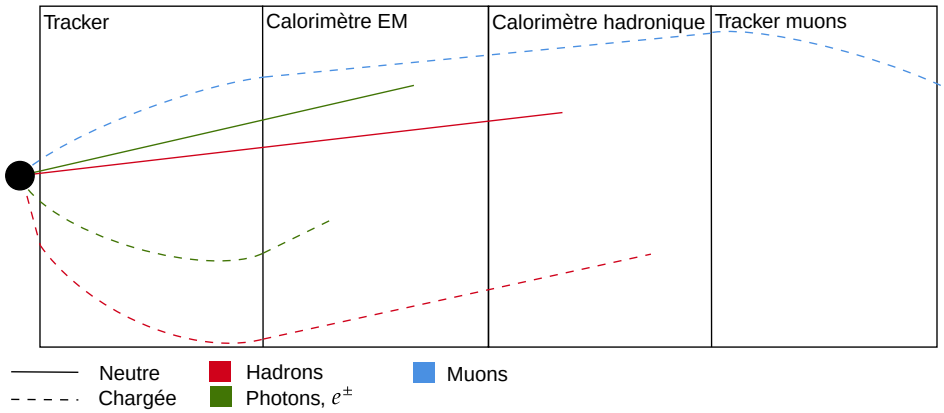


FIGURE 3 – Schéma indicatif du détecteur ATLAS vu de la tranche. Les trajectoires pour différents types de particules y sont illustrées. En réalité, de nouvelles gerbes peuvent se former au sein du détecteur, mais sont omises ici par soucis de clarté.

Le détecteur est placé autour du point de collision, formé de plusieurs couches (illustrée dans 3) :
- $1^{ère}$ couche : tracker. Il détecte les particules chargées en mesurant leur position lors du passage dans un champ magnétique. Ainsi, on obtient le rayon de courbure et impulsion de la particule. En effet, une particule chargée dans un champ magnétique subit une force $\vec{F} = m\vec{a} = q\vec{v} \times \vec{B}$. Si on se simplifie au cas où la vitesse initiale, $\vec{v_0}$, est perpendiculaire au champ magnétique, alors on peut se placer dans la base de Frenet ($\vec{T}$,$\vec{N}$,$\vec{K}$) avec $\vec{T}$ selon $\vec{v}$ et $\vec{K}$ selon $\vec{B}$. Toutes les composantes de $\vec{a}$ sont nulles, exceptées $a_N = \frac{qvB}{m}$. Pour ce choix de vitesse initiale, la vitesse est constante selon $\vec{T}$. Aussi, pour une force à effet centripète, l'accélération est donnée par $a = \frac{v^2}{R}$. On obtient donc $R = \frac{mv}{qB}$. L'impulsion pour une particule relativiste est donnée par $p = \gamma mv = \gamma qBR$. Ainsi, pour une grande impulsion, la trajectoire sera peu courbée et donc la mesure moins précise. La mesure de ce rayon donne donc bien des informations sur l'impulsion de la particule et sa charge.
- $2^{ème}$ couche : calorimètre électromagnétique. Il est utilisé pour détecter les électrons et photons. Les particules (chargées ou neutres) déposent leur énergie dans le matériau, arrachant des électrons qui sont récupérés en appliquant un champ électrique. Un courant électrique est donc créé, qui est donc proportionnel à l'énergie déposée. Il est constitué de plusieurs cellules, permettant donc de localiser le dépôt d'énergie.
- $3^{ème}$ couche : calorimètre hadronique. Il a le même fonctionnement que le précédent, mais pour les hadrons, qui pénètrent plus loin dans la matière. Dans les calorimètres, plus l'énergie déposée est grande, plus la mesure est précise.
- $4^{ème}$ couche : tracker muon. Le muon est un fermion, et n'est donc pas sensible à l'interaction forte. Sa masse ($m_\mu = 105Gev$) étant très grande devant celle de l'électron, le Bremsstrahlung n'est pas un processus atteignable. Ainsi cette particule élémentaire ne peut interagir que par diffusion électromagnétique avec la matière. Le muon est donc très pénétrant, et laisse donc quelques traces dans la 1ère couche, mais peu dans la $2^{ème}$ et $3^{ème}$, on place donc un tracker pour le détecter. Seule cette particule parvient à cette couche, les autre étant arrêtées plus tôt dans le détecteur.
Chaque série de points est localisée dans un tracker / calorimètre. En combinant l'énergie et la position de la particule, on peut donc reconstituer son quadrivecteur $p = (E, p_x, p_y, p_z)$.

A.3 Détection de particules

Pour détecter une nouvelle particule, on étudie les processus pouvant l'impliquer. Si on a une particule Z' de masse $M_{Z'}$ alors, on observera un pic de résonance sur la section efficace au niveau de la masse $M_{Z'}$ si le processus peut bien passer par cette particule virtuelle Z'. On peut prendre comme exemple la désintégration $H \to \gamma\gamma$, étudiée dans [Col20]. La masse invariante de cette réaction est donnée par $M = (\sum p)^2 = m_H = (\sum E)^2 - (\sum p)^2$. Ainsi, en mesurant la présence de photon pour différentes énergies (et donc différentes masses invariantes), on trouve un pic du nombre d'évènements mesurés à $m_H = 125Gev$. En effet, des photons peuvent être créés à cette énergie-ci via un processus supplémentaire : la désintégration du Higgs.
Plus la mesure est précise, plus la distribution sera piquée autour de la valeur de la masse. Cependant, on ne peut pas obtenir une mesure d'une parfaite précision, de par le hasard intrinsèque de chaque phénomène quantique, le fait que pour deux particules identiques le jet n'est pas obligatoirement le même, car les particules créées peuvent être différentes.
Le détecteur ATLAS permet d'obtenir de nombreuses informations à propos des particules le traversant, mais leurs identifications n'est pas une tache aisée. On peut identifier le photon, qui ne dépose de l'énergie que dans le calorimètre électromagnétique. Les phénomènes lui étant accessibles, i.e effet photoélectrique, diffusion compton et création de paire, ne lui permettent pas de laisser de trace. On détecte uniquement le point final de sa trajectoire. Ceci nous permet de le différencier de l'électron, qui dépose également son énergie dans le calorimètre, mais qui laisse une trace. Le muon est identifiable puisqu'il s'agit de l'unique particule laissant une trace dans le dernier calorimètre. Le reste des particules est difficile à identifier, on ne peut donc pas différencier individuellement les hadrons.

A.4 Calibration

Le hasard intrinsèque à la mesure de processus quantiques implique que les mesures d'une énergie suivent une distribution de probabilité. On note le maximum de sa densité de probabilité $E_{true} = E_t$. Cependant, on ne mesure qu'une partie de l'énergie déposée, ainsi on mesure en réalité une gaussienne centrée en $E_* < E_t$. L'énergie E_* dépend de la position angulaire (direction de son impulsion) de la particule, de la façon et l'endroit dont la gerbe hadronique a été créée. Par exemple, si la gerbe est créée entre les deux calorimètres, de l'énergie peut être perdue et donc non détectée.
Le but de la calibration est donc de recentrer la gaussienne, pour parvenir à mesure l'énergie E_t. Pour une énergie donnée, on souhaite déterminer le facteur de correction R à appliquer pour pouvoir calibrer correctement le détecteur. La calibration a une quinzaine de paramètres, il faudrait donc suffisamment de points pour déterminer un facteur d'échelle à appliquer sur les 15 dimensions, mais ce nombre est trop important. On parle de dimension de la dimensionnalité. Pour effectuer une étude statistique sur un intervalle de $\mathbb{R}$ [0,10], quelques centaines de mesures peuvent être suffisantes. Si l'on ajoute plusieurs dimensions au problème, pour ce même nombre de mesures, une grande partie

de l'espace étudié sera trop grand pour effectuer une étude statistique, puisque les mesures couvriront désormais qu'une faible proportion de l'espace total. La collaboration ATLAS utilise l'inversion numérique pour calibrer correctement son détecteur, mais ceci n'est valable que lorsque la fonction étudiée est linéaire, ce qui n'est pas le cas pour des énergies faibles. Pour pallier à ce problème, on souhaite utiliser un réseau de neurones, qui peut traiter un problème en n dimensions.

B Formulation du problème

On mesure une énergie E_r que l'on souhaite relier à l'énergie calibrée $E_c = u(E_r)$ avec $u(E)$ la fonction de calibration. On peut note que l'on n'a pas nécessairement $u(E_*) = E_t$. En effet, ce n'est pas parce qu'on mesure le plus grand nombre d'évènements à une énergie E_*, que le processus est réellement plus probable à l'énergie $u(E_*)$. En effet, les détecteurs peuvent par exemple être plus sensibles à une certaine énergie, augmentant donc le nombre coups à ce niveau, alors que le processus se déroule en majorité à une énergie différente E_t où le nombre de coups sera plus faible.
Pour chaque énergie E_t, l'énergie mesurée peut différer de cette valeur et suit donc une fonction de densité de probabilité $f(E_r, E_t, \theta)$ où θ sont les paramètres de la mesure. Ainsi, pour chaque E_t, l'énergie calibrée suit également une fonction de densité de probabilité $g(E_c, E_t)$ (puisqu'on l'obtient à partir de E_r).
On peut relier les deux fonctions de densité de probabilités. Soit $\mathcal{E}_r$ et $\mathcal{E}_c$ deux variables aléatoires reliées par $\mathcal{E}_c = u(\mathcal{E}_r)$, et suivant respectivement les densités de probabilités $f(E_r)$ et $g(E_c)$. La probabilité de trouver $\mathcal{E}_c$ dans un intervalle ΔE_c est :

$$P(E_c < \mathcal{E}_c < E_c + \Delta E_c) = g(E_c)\Delta E_c \tag{7}$$

Or en utilisant $E_r = u^{-1}(E_c)$ et $E_r + \Delta E_r = u^{-1}(E_c + \Delta E_c)$, on peut réécrire cette probabilité :

$$P(u(u^{-1}(E_c)) < u(u^{-1}(\mathcal{E}_c)) < u(u^{-1}(E_c{+}\Delta E_c))) = P(u(E_r) < u(\mathcal{E}_r) < u(E_r{+}\Delta E_r)) = P(E_r < \mathcal{E}_r < E_r{+}\Delta E_r) = f(E_r)\Delta E_r \tag{8}$$

qui fonctionne uniquement si $u(E_r)$ est une fonction monotone. En effet, l'égalité $P(u(E_r) < u(\mathcal{E}_r) < u(E_r + \Delta E_r)) = P(E_r < \mathcal{E}_r < E_r + \Delta E_r)$ est correcte uniquement dans ce cas précis. Ainsi, on a :

$$f(E_r)\Delta E_r = g(E_c)\Delta E_c. \tag{9}$$

On note la fonction inverse $u^{-1}(E_c) = v(E_c)$. Ainsi, on a $E_r = v(E_c)$ et $E_r + \Delta E_r = v(E_c + \Delta E_c)$, soit $\Delta E_r = v(E_c + \Delta E_c) - v(E_c)$. Ainsi :

$$g(E_c) = f(E_r)\frac{\Delta E_r}{\Delta E_c} = f(E_r)\frac{v(E_c + \Delta E_c) - v(E_c)}{\Delta E_c} = f(v(E_c))v'(E_c) \tag{10}$$

Les fonctions de densité de probabilité étant positives, si $v(E_c)$ est monotone décroissante, il est nécessaire de prendre la valeur absolue de $v'(E_c)$.
On souhaite que la probabilité d'une énergie de calibration E_c soit maximale en E_t, au centre de la gaussienne. Ceci se traduit en une équation différentielle :

$$\frac{dg(E_c)}{E_c}(E_t, E_t) = 0 \tag{11}$$

Pour une probabilité de mesure de E_r pour une certaine énergie E_t suivant une gaussienne centrée en $E_* = \mu(E_t$:

$$f(E_r, E_t) = C_{norm} e^{-\frac{1}{2\sigma(E_t)^2}(v(E_c) - \mu(E_t))^2}$$

On peut l'injecter Eq. 10 puis Eq. 11. On trouve ainsi :

$$v''(E_t) - \frac{v^{'}(E_t)^2}{\sigma(E_t)^2}(v(E_t) - \mu(E_t)) = 0 \tag{12}$$

Ici E_t est donc une variable muette, que l'on peut remplacer l'énergie normalisée $x = E/E_{norm}$. La résolution de Eq. 12 implique l'inversion de la fonction une fois le réseau entrainé. Ainsi, même lorsque v est déterminée précisément via le réseau de neurone, si son inversion n'est pas précise, u peut être inutilisable.

On cherche donc à exprimer l'équation différentielle à résoudre directement sur u, pour éviter de perdre en précision lors de l'inversion. On utilise ainsi le lien reliant u et v :

$$u \circ v\,(x) = u(v(x)) = x \tag{13}$$

On peut ainsi relier les différentes dérivées avec

$$\frac{d(u \circ v)}{dx}(x) = 1 = u'(v(x)) \times v'(x) \leftrightarrow v'(x) = \frac{1}{u'(v(x))}$$

$$v''(x) = -\frac{\frac{d}{dx}u'(v(x))}{u'(v(x))^2} = -\frac{u''(v(x)) \times v'(x)}{u'(v(x))^2} = -\frac{u''(v(x))}{u'(v(x))^3}$$

On injecte cela dans l'équation différentielle sur v :

$$-\frac{u''(v(x))}{u'(v(x))^3} - \frac{1}{u'(v(x))^2\sigma^2}(v(x) - \mu(x)) = 0$$

On effectue ensuite le changement de variable $x = u(x)$:

$$-\frac{u''(v(u(x)))}{u'(v(u(x)))^3} - \frac{1}{u'(v(u(x)))^2\sigma^2}(v(u(x)) - \mu(u(x))) = 0$$

$$-\frac{u''(x)}{u'(x)^3} - \frac{1}{u'(x)^2\sigma^2}(x - \mu(u(x))) = 0$$

$$u''(x) + \frac{u'(x)}{\sigma^2}(x - \mu(u(x))) = 0 \tag{14}$$

Nous devons donc résoudre cette équation, afin de trouver la fonction de calibration u.

C Introduction aux réseaux de neurones

C.1 Principe d'un neurone / perceptron

Le fonctionnement d'un perceptron est basé sur celui des neurones de notre cerveau, prenant des entrées et renvoyant un signal en sortie. Il est un outil mathématique, prenant des datas en entrée, et ressortant un réel en sortie. Ses paramètres sont les poids, biais et fonction d'activation. Les étapes clés de son fonctionnement sont :
1. **Sommation des entrées**, pondérée par les poids ω_j et le biais $b_i \rightarrow z_i = b_i + \sum_{j=1}^{n} \omega_j x_j$, avec n le nombre d'entrées du neurone.
2. Passage du résultat dans la **fonction d'activation**. $a_i = a(z_i) = \frac{1}{1-e^{-z_i}}$ (fonction d'activation sigmoïde)
3. **Output**. Ce dernier peut être le résultat de la fonction d'activation, une valeur binaire choisie à partir de a_i, ou même simplement z_i dans certains cas.

C.2 Fonctions d'activation

Elle permet d'introduire des non-linéarités entre les couches du réseau de neurones.
Ex : Si on connecte un neurone A avec une sortie x, à un neurone B qui a une entrée y. On a donc $y = \omega x + b$. On applique ensuite la fonction d'activation, par exemple ReLU → a=ReLU(y). Si y est négatif alors $a = 0$ sinon $a = y$. Ainsi la sortie du neurone B n'est pas linéaire en fonction de la sortie du neurone A.
Ces fonctions sont primordiales, car on ne pourrait pas introduire de non linéarités sans elles, simplement en combinant des neurones entre elles.
Ex : en sortie de B, on a désormais $a_b = \omega_a x + b_b$. Donc en sortie d'un neurone C, on aurait $a_c = \omega_b a_b + b_c = \omega_b\omega_a x + \omega_b b_b + b_c$. La sortie du neurone C est toujours linéaire en fonction de celle du neurone A ou B.
On parle d'activation, puisque lorsque la sortie d'un neurone est nulle, il n'a donc aucun impact sur les couches suivantes. On parle de neurone inhibé, ou actif dans le cas contraire.
On peut citer quelques exemples de fonctions d'activation :

- **ReLU(x) = max(x,0)**

$$a(x) = max(x, 0)$$

Permet un filtre des données, ne laissant passer que les positives. Elle est utilisée dans les **couches intermédiaires**, mais **jamais la finale**.

- **Sigmoid**

$$a(x) = \frac{1}{1 - e^{-x}}$$

Renvoie une valeur entre 0 et 1, et s'interprète comme une probabilité. Utilisée pour les **classification binaire** (lorsqu'un modèle doit déterminer deux labels).

Cependant, sa dérivée devient très rapidement très faible. Ainsi, lors de la descente de gradient, effectuée par chaine de dérivée, elle peut faire tendre le gradient vers 0 et donc empêcher le réseau de s'ajuster. En effet, si on a deux neurones reliés, donc le dernier renvoie $y = a(x)$. Alors la descente de gradient sera :

$$\frac{\partial loss}{\partial \omega_j} = \frac{\partial loss}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial \omega_j} = \frac{\partial loss}{\partial y}\frac{\partial a(x)}{\partial x}\frac{\partial x}{\partial \omega_j}$$

Ainsi le terme $\frac{\partial a(x)}{\partial x}$ peut tendre vers 0, empêchant le réseau d'ajuster ω_j. Ainsi, il fait éviter d'utiliser des Sigmoid en série, et plutôt l'utiliser uniquement dans la dernière couche par exemple.

- **Softplus**

$$a(x) = \frac{1}{\beta}\ln(1 + e^{\beta x})$$

Cette fonction est strictement positive. Ainsi, elle peut être utile lorsqu'on souhaite travailler avec des outputs uniquement positifs. Cela peut être le cas lorsque la fonction loss prend passe l'output dans une fonction log, afin d'éviter les erreurs.

C.3 Organisation en réseau

Les perceptrons peuvent être reliés les uns aux autres, formant un réseau. Ce dernier peut contenir un nombre arbitraire de couches. Le choix difficile de cet hyperparamètre est sujet de débat dans la communauté scientifique, et varie d'un problème à l'autre.
La première couche prend les data du problème en entrée. L'entrée de la couche suivante est la sortie de chaque neurone de la couche précédente, i.e. les fonctions d'activation. La sortie de la dernière couche peut varier en fonction du problème. On peut souhaiter obtenir une unique valeur binaire (à partir de la fonction d'activation), une quantité non normalisée que l'on compare à une donnée connue (on ressort simplement z_i sans passer par la fonction d'activation)... Pour un set de data contenant des vecteurs de dimension n à donner en entrée du réseau, la première couche sera constituée de n neurones, de sorte que chaque élément du vecteur ait un poids attribué. Les couches cachées ont une taille arbitraire. Le nombre de neurones de la dernière couche dépend de la sortie souhaitée. Si l'on souhaite un réel en sortie, elle sera composée d'un unique neurone, si l'on souhaite un vecteur de taille d elle comportera d neurones.
La mise en réseau des neurones peut également être vue d'un point de vue matriciel. Cette équivalence entre les deux approches est illustrée dans Fig. 4.

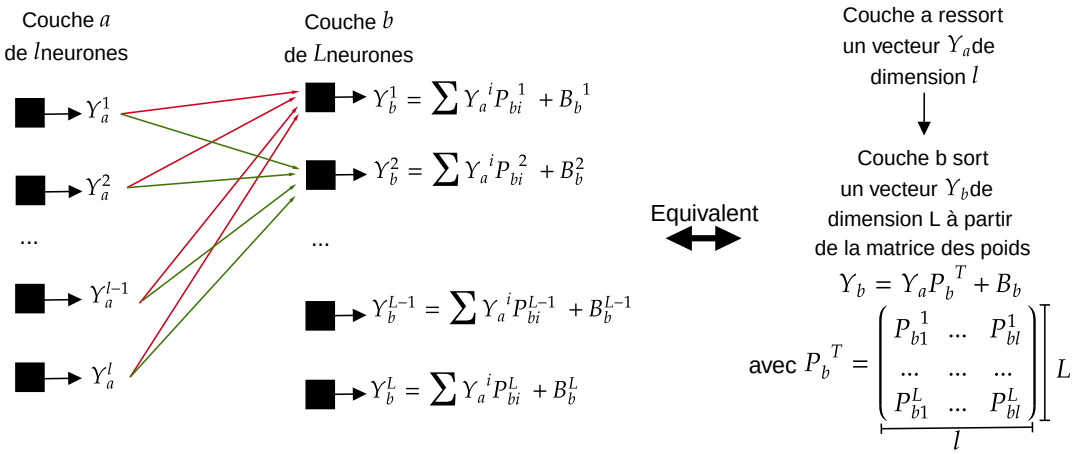


FIGURE 4 – Représentation de la connexion d'une couche a formés de l neurones avec une couche b formé de L neurones. La partie droite présente l'équivalence de cette situation avec un simple produit matriciel.

C.3.1 Utilisation des données

Le réseau de neurone est un outil pour déterminer la valeur d'un paramètre $\hat{y}_i$ attribué à chaque data $\vec{X}_i$. Si l'on dispose d'un grand nombre de vecteurs $\vec{X}_i$, on peut séparer ce set de datas en un set de data pour l'entrainement du réseau (80%), et un pour le tester (20%). L'objectif du premier est de donner en entrée un vecteur $\vec{X}_i$ et chercher à faire correspondre la sortie $y_{(i)}$ du réseau à la valeur connue du paramètre $\hat{y}_{(i)}$. À chaque passage, on modifie la valeur des poids et paramètres pour chaque neurone pour se rapprocher de $\hat{y}_{(i)}$. En répétant cela pour l'ensemble du train data, on entraine notre réseau à déterminer le paramètre souhaité pour un vecteur en entrée. On peut ensuite tester que notre modèle fonctionne en donnant les données test en entrée pour les poids et biais des neurones fixés par les

données d'entrainement. En comparant le paramètre $\hat{y}_{(i)}$ connu de chaque donnée test à celui en sortie du réseau, on peut jauger la précision de notre modèle.

C.3.2 Entrainement du réseau (descente de gradient)

Pour fixer les poids et biais du réseau, on utilise la **descente de gradient**.

On peut comparer le résultat de sortie du réseau y_i (valeur prédite) à la valeur réelle connue $\hat{y}_i$. Pour cela, on passe par une fonction *loss*, dont il existe plusieurs types. On cherche à minimiser cette fonction, afin que la valeur de sortie soit la plus proche possible de la réelle. Pour cela, on calcule le gradient de la fonction *loss* par rapport à chaque paramètre du réseau (poids et biais de chaque neurone). Et on fait varier chaque paramètre dans la direction opposée de la variation, se rapprochant donc du minimum de *loss* :

$$\theta \leftarrow \theta - \eta \nabla_\theta loss(\theta)$$

L'hyperparamètre η permet de fixer la tailler des pas lors de la descente de gradient. Un η trop grand rendra le minimum difficile à atteindre, un trop petit prendra trop de temps pour y arriver. Il faut donc le choisir judicieusement.

C.3.3 Fonctions loss

Différentes fonctions loss peuvent être utilisées, et sont à choisir avec minutie en fonction du problème étudié.
Soit N datas $\vec{X}_{(i)}$ en dimension n dont on connait la caractéristique $\hat{y}_i$ de dimension variable.
Dans la suite, les sommes sont effectuées sur toutes les data N. On compare pour chaque data le résultat prédit avec celui attendu. La fonction loss est donc la somme de cette comparaison pour chacune des data. Pour être plus efficace, on peut créer des sous systèmes "batches" formé de $1 < m < N$ data. La taille du batch est le nombre data données en entrée du réseau avant que les paramètres soient modifiés. L'algorithme va calculer la fonction *loss* en sommant le résultat des m datas, puis le gradient à partir de cette fonction loss, et modifier les paramètres. Puisqu'il s'agit d'une somme, une autre manière d'opérer est de calculer l'élément sommé dans la fonction *loss* et le gradient pour une data du batch, sans effectuer la descente de gradient. On effectue ensuite la moyenne des gradients en sommant chaque résultat et normalisant par m, avant d'effectuer la descente de gradient avec ce gradient moyen. Une époque correspond donc à un minima de m/N batch pour lesquels on modifie les paramètres.
Dans la suite, les formules de fonctions loss sont présentées avec une somme sur tous les éléments du set de data (N). En pratique celle-ci est faite sur les éléments de chaque batch de taille m.
On présente les calculs avec une unique somme sur chaque data. Dans le cas où la sortie est un vecteur $\vec{y}_{(i)}$, il faut ajouter une seconde somme sur chaque élément du vecteur. On peut citer les fonctions les plus couramment utilisées :

- **La vraisemblance (Log-vraisemblance négative - Negative Log-Likelihood) :**
Pour des données dont la sortie attendue est **binaire** (0 ou 1), on peut chercher à maximiser la vraisemblance :

$$L = \prod_{i=1}^{N} a(z_i)^{\hat{y}_i} (1 - a(z_i))^{\hat{y}_i}$$

qui sera donc maximale lorsque les valeurs des poids et biais seront en adéquation avec le $\hat{y}_i$ prédit connu. En pratique, on cherche le minimum de $-log(L)/N$:

$$-LL/N = -1/N \sum \hat{y}_{(i)} \log a_i + (1 - \hat{y}_{(i)}) \log(1 - a_i)$$

-**Erreur quadratique moyenne (Mean Squared Error - MSE) :**
Très utilisé pour les problèmes de **régression**. Elle est sensible aux valeurs aberrantes et peut conduire à des gradients plus importants pour les erreurs importantes. On calcule la moyenne de l'écart au carré de chaque data.

$$MSE(X, \hat{y}) = 1/N \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$

avec n la dimension du vecteur de sortie.

Ces fonctions ne sont cependant pas utilisables dans notre problème.

C.4 Risques du réseau de neurones

La création du réseau de neurone repose sur le choix de ses hyperparamètres, qui ne sont pas modifiés lors de la descente de gradient. On peut citer :

- Le nombre de neurones par couches
- Le nombre de couches
- Le choix de la fonction loss
- Le choix de la fonction d'activation pour chaque neurone
- Le learning rate η

Un mauvais choix de ces paramètres peut influencer grandement les résultats obtenus. Ces derniers peuvent également être impactés par la préparation des data, ou aussi l'entrainement. Le nombre d'époques, i.e. de cycle complet dans l'ensemble des data, peut grandement impacter l'utilisation du réseau sur les données test. Plus le nombre d'époques est grand, plus la fonction loss diminue (jusqu'à atteindre son minimum). Cependant, un entrainement trop long sur les train-data n'améliore pas la précision sur les données de test, et peut même la dégrader. On parle alors d'overfitting. Le réseau a calibré ses paramètres trop spécifiquement sur les data d'entrainement, et n'est plus capable de s'adapter à des data nouvelles.

C.5 Exemples d'applications

Un exemple très simple d'utilisation de réseau de neurones est la régression linéaire. Si on a un nuage de points distribué dans un espace 2D (x_1,$\hat{x_2}$), on peut essayer de faire passer une droite $x_2 = ax_1 + b$ au milieu de ces points. Dans ce cas, le rôle du RN est d'ajuster les paramètres a et b pour minimiser la distance entre la droite et les points. Donc pour une data donnée dans notre set, on donne x_1 en entrée, le RN prédit un x_2, qu'il compare à $\hat{x_2}$.

D Libraire Pytorch

D.1 Autograd

La fonction loss peut être une fonction de tenseur, et donc elle-même un tenseur. Cependant, pour modifier les poids, il faut que la fonction "torch.autograd" puisse ressortie un vecteur, afin de modifier correctement chaque poids. Pour cela, on peut se placer dans le cas général où la fonction loss est $\vec{y} = f(\vec{x})$, avec $\vec{x}$ les poids dans un vecteur de dimension n. La Jacobienne est donnée par J :

$$J = \left(\frac{\partial \mathbf{y}}{\partial x_1}, \ ..., \ \frac{\partial \mathbf{y}}{\partial x_n} \right) = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{15}$$

Pour pouvoir modifier les poids, la fonction "torch.autograd" doit pouvoir ressortir un vecteur de taille n et non pas une matrice $n \times n$. Ainsi, on lui fournit un vecteur $\vec{v} = (1, ..., 1)$ de dimension n, pour faire le produit scalaire avec la transposée de la Jacobienne $J^T \cdot \vec{v}$:

$$J^T \cdot \vec{v} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} \frac{\partial y_i}{\partial x_1} \\ \vdots \\ \sum_{i=1}^{m} \frac{\partial y_i}{\partial x_n} \end{pmatrix} \tag{16}$$

Chaque composante du vecteur gradient encode les variations de la fonction loss selon chaque direction par rapport à un élément du poids. On peut ensuite modifier le poids en utilisant ce vecteur $\vec{x} - J^T \cdot \vec{v}$.
Une méthode alternative est de sommer toutes les composantes de $\vec{y}$ pour le transformer en scalaire, et ainsi de calculer le gradient sur ce dernier :

$$\vec{\nabla} \sum_{i=1}^{m} \partial y_i$$

Prenons un exemple. On peut définir un poids $\vec{a} = (a_1, a_2)$ et une fonction loss $\vec{Y} = 3a^3$. Ainsi Y est un vecteur de taille 2 $\vec{Y} = (3a_1^3, 9a_2^3)$.
- **Méthode 1 :** La Jacobienne est donnée par

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial a_1} & \frac{\partial y_1}{\partial a_2} \\ \frac{\partial y_2}{\partial a_1} & \frac{\partial y_2}{\partial a_2} \end{pmatrix} = \begin{pmatrix} 9a_1^2 & 0 \\ 0 & 9a_2^2 \end{pmatrix} \tag{17}$$

On donne à "torch.autograd" un vecteur de taille 2 :

```
external_grad = torch.tensor([1., 1.])
Y.backward(gradient=external_grad)
```

lui permettant de calculer le vecteur avec lequel il effectuera la descente en gradient :

$$Grad = J^T \cdot \vec{v} = \begin{pmatrix} \frac{\partial y_1}{\partial a_1} & \frac{\partial y_1}{\partial a_2} \\ \frac{\partial y_2}{\partial a_1} & \frac{\partial y_2}{\partial a_2} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\partial y_1}{\partial a_1} + \frac{\partial y_1}{\partial a_2} \\ \frac{\partial y_2}{\partial a_1} + \frac{\partial y_2}{\partial a_2} \end{pmatrix} = \begin{pmatrix} \frac{\partial y_1}{\partial a_1} \\ \frac{\partial y_2}{\partial a_2} \end{pmatrix} = \begin{pmatrix} 9a_1^2 \\ 9a_2^2 \end{pmatrix} \tag{18}$$

- **Méthode 2 :** on peut aussi sommer la fonction loss sur toutes ses composantes.

$$Y = Y_1 + Y_2 = 3a_1^3 + 9a_2^3$$

Ainsi le gradient est donné par

$$Grad = \vec{\nabla} Y = \begin{pmatrix} \frac{\partial Y}{\partial a_1} \\ \frac{\partial Y}{\partial a_2} \end{pmatrix} = \begin{pmatrix} 9a_1^2 \\ 9a_2^2 \end{pmatrix}$$

Les deux méthodes permettent bien d'aboutir au même résultat. Ainsi, on peut modifier les poids

$$a \to a - Grad = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} - \begin{pmatrix} 9a_1^2 \\ 9a_2^2 \end{pmatrix}$$

La gestion de la descente de gradient, est géré par l'*optimizer*. Il en existe de différentes sortes. Nous utilisons ici ceux fournit par la librairie Pytorch et [Nov20]. Empiriquement, le plus efficace pour notre problème est DiffGrad, introduit récemment dans [Dub+21].

E Résolution d'une équation différentielle avec un réseau de neurones

E.1 Équation différentielle du premier ordre

Nous pouvons commencer par le cas le plus simple : l'équation différentielle d'ordre 1 :

$$y' + f(x, y) = 0,$$

avec $f(x, y)$ une application dans $\mathcal{R}$. Une équation différentielle du premier ordre a une solution analytique simple. Nous pouvons ainsi vérifier facilement la solution obtenue par le réseau de neurones. Pour la résoudre via un réseau de neurones, nous utilisons la configuration présentée dans Fig. 5.

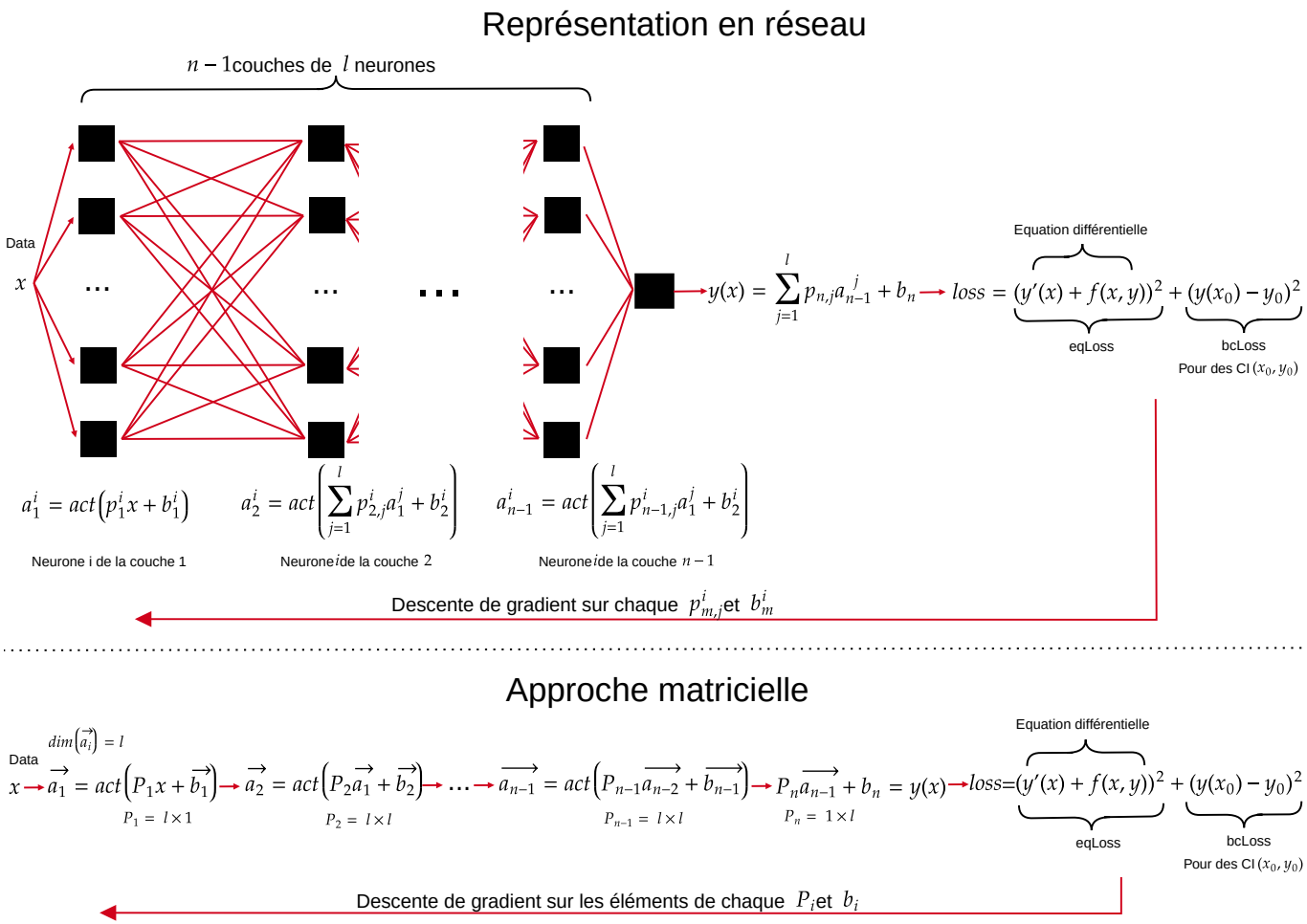


FIGURE 5 – Représentation du réseau de neurone utilisé pour résoudre une équation différentielle d'ordre 1. La partie basse correspond à une description matricielle des calculs appliqués sur les data. On note $act(X)$ la fonction d'activation de chaque neurone. Le réseau prend un scalaire x en entrée, et ressort un scalaire $y(x)$.

On donne en entrée du réseau une valeur x prise aléatoirement dans une liste couvrant un intervalle arbitraire. L'aléatoire permet d'éviter l'overfitting sur une partie de l'intervalle. Après le passage dans les n couches du réseau, ce dernier renvoie un scalaire y. En minimisant la fonction loss, nous cherchons à faire correspondre ce y prédit avec la valeur de la fonction y solution de l'équation différentielle, pris en x. En utilisant la classe `torch.autograd.grad(x,y`, on obtient également la dérivée y' de y prise en x.

Les $n-1$ premières couches sont constituées de l neurones. Un neurone effectue une somme pondérée des entrées par un poids, et ajoute un biais. On passe ensuite le résultat dans une fonction d'activation, pour introduire des non-linéarités. Celle-ci permet d'obtenir des résultats plus complexes. Nous avons ici utilisé la fonction d'activation suivante :

$$Mish(X) = X \tanh\left(\ln(1 + e^X)\right) \tag{19}$$

Cette fonction a l'avantage d'être $\mathcal{C}^\infty$, ce qui est important pour les gradients, et le sera d'autant plus pour les équations d'ordre supérieur.
La dernière couche du réseau est formée d'un seul neurone, renvoyant donc un scalaire. Il est important de ne pas passer l'output dans une fonction d'activation, ou sinon une fonction d'activation non bornée. En effet, on souhaite que l'output $y(x)$ puisse prendre n'importe quelle valeur dans $\mathcal{R}$, il ne faut donc pas que celle-ci soit restreinte par la fonction d'activation. Si l'on travaille avec solution connue comme étant uniquement positive/négative, il peut être judicieux d'utiliser une fonction d'activation bornée, dans ce cas-là.

Pour que y soit solution, il faut $y' + f(x,y)$ soit nul. Ainsi, on utilise la fonction loss $eqLoss = (y' + f(x,y))^2$, que nous souhaitons donc minimiser. Lorsque celle-ci est minimale, alors y est solution de l'équation différentielle.
Pour atteindre une solution unique, il faut prendre en compte les conditions initiales. Pour un problème en une dimension, on donne (x_0, y_0) comme CI. On peut calculer après chaque descente de gradient $bcLoss = (y(x_0) - y_0)^2$. Cette fonction sera minimale lors l'estimation de y en x_0, notée $y(x_0)$, sera égale à la valeur réelle y_0.

Ainsi, pour obtenir une solution **unique** y à l'équation différentielle, on cherche à minimiser la fonction totale :

$$loss = eqLoss + bcLoss = (y' + f(x,y))^2 + (y(x_0) - y_0)^2$$

À partir de cette fonction, on applique une descente de gradient sur l'ensemble des paramètres du réseau.

E.2 Résolution d'une équation différentielle du second ordre

Une équation différentielle de second ordre est de la forme

$$y'' = f(x, y, y')$$

La structure et principe du réseau de neurone utilisé est exactement le même que celui pour les équations différentielle d'ordre 1. La seule différence se situe dans les conditions initiales. Pour obtenir une solution unique, il faut cette fois-ci fournir deux conditions initiales. Soit deux sur la fonction : $y(x_0 = y_0)$ et $y(x_1) = y_1$. On peut aussi en donner une sur la fonction $y(x_0) = y_0$ et une sur la dérivée $y'(x_1) = y'_1$.
On implémente donc une fonction loss supplémentaire

$$bcloss = (y(x_0) - y_0)^2 + (y(x_1) - y_1)^2 \quad \text{ou} \quad bcloss = (y(x_0) - y_0)^2 + (y'(x_1) - y'_1)^2$$

E.3 Résolution de l'équation différentielle sur la fonction inverse de calibration

On souhaite résoudre Eq. 12, avec un réseau de neurones, i.e :

$$v''(E_t) - \frac{v^{'}(E_t)^2}{\sigma(E_t)^2}(v(E_t) - \mu(E_t)) = 0$$

où v, σ, μ sont des fonctions de l'énergie.
On définit également le rapport $r(E_r, E_t) = \frac{E_r}{E_t}$, qui compare donc les énergies mesurées pour une gaussienne, au centre calibré de la gaussienne. On peut également l'écrire $r(E_r, E_t) = \frac{v(E_c)}{E_t}$. Le rapport maximal est donné par $R = \frac{E_*}{E_t}$. Cette quantité compare le maximum de probabilité pour les énergies calibrée, et non calibrée. On pourrait notamment calculer $\frac{v(E_t)}{E_t} = r(E_{rt}, E_t)$. Ici $v(E_t)$ renvoie la valeur mesurée E_{rt} telle qu'une fois calibrée, on retombe sur E_t, et on la compare à E_t, ce qui donne la valeur du rapport r. . On n'a pas nécessairement $R = r(E_{rt}, E_t)$ car on n'a pas forcément $v(E_t) = E_*$. Les deux maximums ne sont pas obligatoirement reliés directement par la calibration, comme l'illustre Fig. 6. Ainsi, dans l'équation différentielle, on calcule $v(E_t) - \mu(E_t) = E_t \times r(E_{rt}, E_t) - R(E_t) \times E_t$. Le fait que ce terme ne soit pas forcément nul réside donc justement dans la nuance existant entre r et R.

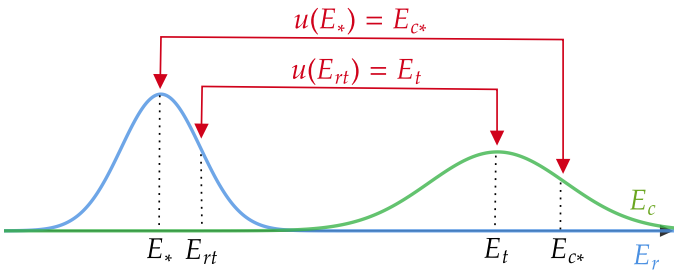


FIGURE 6 – Illustration des différences possibles pouvant exister entre les énergies calibrées et non calibrées. La courbe bleue correspond à l'énergie non calibrée de maximum E_*, la verte à l'énergie calibrée de maximum E_t. On fait apparaitre les liens entre ces énergies et leur énergie calibrée/mesurée.

La fonction $\mu(E_t) = E_*$ est reliée à ces quantités par $\mu(E_t) = RE_t$. Ainsi, la connaissance de R nous renseigne sur l'équation différentielle à résoudre.

E.4 Normalisation des données

Pour trouver une solution avec un réseau de neurones, il est préférable d'utiliser des données normalisées. En effet, cela permet de pallier à différents problèmes. Lorsqu'en entrée, on donne des sets de data de différents types, normaliser chaque type permet d'éviter de donner naturellement trop de poids aux données avec des valeurs importantes. Par exemple, si on souhaite étudier l'état d'une voiture via un RN, avec en entrée la valeur de kilométrage total, et

celle de l'âge du conducteur, les normaliser permet de remettre à plat les ordre de grandeur pouvant les séparer. Il est également important de normaliser ses données d'entrée pour faciliter la descente de gradient. Des valeurs importantes peuvent nécessiter des poids très petits, rendant moins précis les calculs. Aussi, certaines fonctions d'activation tuent le neurone très rapidement pour des valeurs n'appartenant pas à $[-1, 1]$. Ainsi, la normalisation permet de créer un modèle avec les fonctions d'activations souhaitées, en limitant les erreurs de calculs.

Dans notre cas, on souhaite obtenir une fonction de calibration inverse $v(x)$ où $x \in [-1, 1]$. Expérimentalement, certaines énergies ne sont pour le moment pas atteignables $E_t < E_{max}$. On peut donc utiliser une énergie de normalisation $E_t < E_{norm} < E_{max}$, pour normaliser nos data. Ainsi, lorsque nous travaillerons avec de vraies énergies, il suffira de calculer $E_r/E_{norm} = x \to u(x) = X_c \to E_c = E_{norm}X_c$ pour obtenir l'énergie calibrée à partir de celle mesurée, en utilisant la fonction de calibration que l'on aura déterminée via les data normalisées. L'équation à résoudre devient :

$$v''(x) - \frac{v'(x)^2}{\sigma(x)^2}(v(x) - \mu(x)) = 0$$

avec $\mu(x) = xR(x)$. En normalisant nos données, la fonction de réponse devient donc :

$$R(x) = \alpha\left(1 - \frac{1}{(\beta + x)p}\right)$$

Dans le cas général, on peut utiliser la réponse en énergie R suivante :

$$R(E_t) = \frac{E_*}{E_t} = \alpha\left(1 - \frac{1}{(\beta + E_t)^p}\right) \tag{20}$$

On retrouve $R \to \alpha$ pour $E_t \to \infty$. On peut fixer la valeur de β à partir d'une valeur du rapport connue expérimentalement, à basse énergie, tel que $R(20GeV) = 0.8$. Dans ce cas-là, le maximum de la distribution non calibrée devient donc :

$$E_* = \mu(E_t) = E_t R(E_t) = E_t \times \alpha\left(1 - \frac{1}{(\beta + E_t)^p}\right)$$

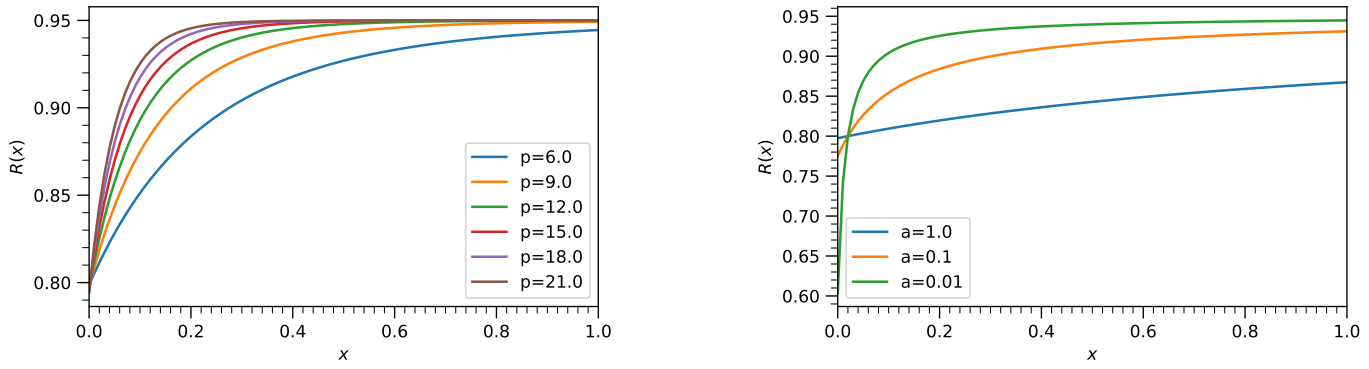
On souhaite que $R(x_0 = 20Gev/E_{norm}) = 0.8$, et $R(x_\infty = 1) = \alpha$. D'après l'expérience, on fixe $\alpha = 0.95$. Les deux conditions nous servent donc à fixer β et p.

$$\begin{cases} \alpha\left(1 - \frac{1}{(\beta+x_0)^p}\right) = 0.8 \\ \alpha\left(1 - \frac{1}{(\beta+1)^p}\right) = \alpha \end{cases} \leftrightarrow \begin{cases} \beta = \left(\frac{1}{1-0.8/\alpha}\right)^{1/p} \\ 1 - \frac{1}{(\beta+1)^p} = 1 \end{cases} \tag{21}$$

Une autre forme que l'on pourrait envisager est :

$$R(E_t) = \alpha + \frac{b}{E_t + c}$$

où R tend bien vers α à haute énergie. On détermine b avec $b = (E_0 + c)(R(E_0) - \alpha)$ avec $E_0 = 20GeV$ et $R(E_0) = 0.8$. Ainsi, on a $\mu = E_t R(E_t) = \alpha E_t + b$. Pour des données normalisées, il faut que $\left|\frac{b}{1+c}\right| << 1 \leftrightarrow |b| << |1 + c| < 1 + |c|$. Ainsi, $||b| - |c|| << 1$. On peut prendre $||b| - |c|| = a$, avec $a = 0.1$, ou$a = 0.01$ pour commencer. Si $b > 0$ et $c < 0$ ou si $b < 0$ et $c > 0$ cela revient à $b + c = a$.



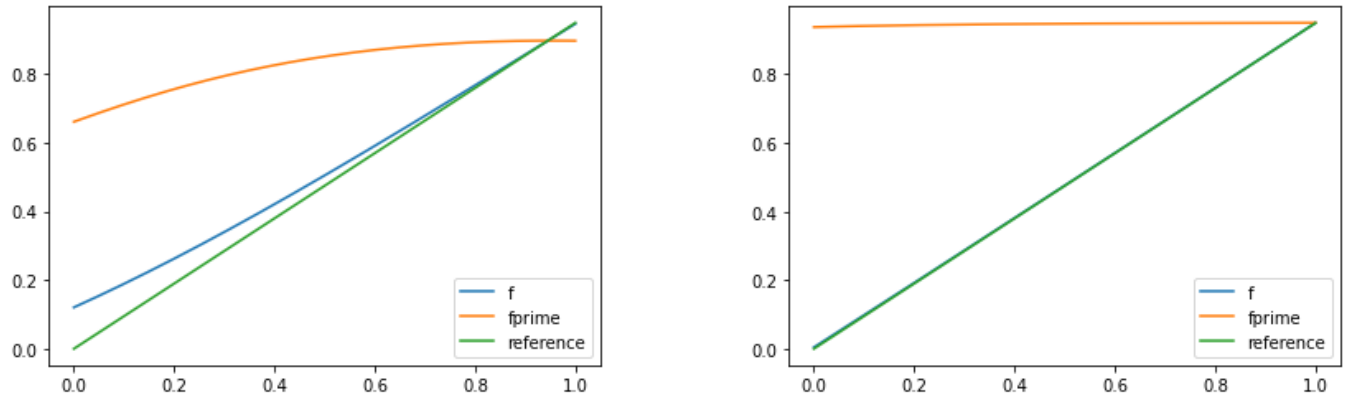
(a) R avec p (b) R avec a

FIGURE 7 – Fig. 7a : $R(x)$ pour différente valeur de puissance p allant de 6 à 20. Fig. 7b : Réponse $R(x)$ pour différente valeur a allant de 0.01 à 1.

Fig. 7a illustre la forme de la réponse R pour différent p. Ne sont affichés que les résultats pour lesquels le système Eq. 21 est satisfait. À partir de $p = 6$ on obtient bien $1 - \frac{1}{(\beta+1)^p} \simeq 1$, ce qui n'est pas le cas pour des valeurs de p plus faibles. On souhaite que pour les hautes énergies, R soit constant à une valeur α, donc $p = 9$ semble faire l'affaire.
Fig.7b illustre la forme de la fonction de réponse pour différente valeur de a. Ainsi, $0.01 < a < 0.1$ semble être une condition satisfaisant $R \to \alpha$ pour $x \to 1$.

E.5 Choix des conditions initiales

Le set de CI que l'on peut choisir est $(x_0, v_0), (x_1, v_1)$ ou $(x_0, v_0), (x_1, v_1')$. Nous souhaitons qu'à haute énergie v soit une fonction affine de pente α, correspondant ainsi au cas $R = \alpha$. Pour ce faire, on prend donc x_0 et x_1 proches de 1 (donc hautes énergies). Pour que la fonction soit affine, on choisit : $v_0 = \alpha x_0$et $v_1 = \alpha x_1$, ou $v_1' = \alpha$.
Cependant, le réseau a des difficultés à effectuer le fit pour des x_0 et x_1 trop rapprochés, lorsqu'on utilise des CI uniquement sur la fonction. C'est ce qu'illustre Fig. 8. On teste pour le cas $R = \alpha$, si v est bien une fonction affine. On choisit cette fois-ci des CI prises en $x_0 = 0.9$ et $x_1 = 0.99$, i.e. proches. Le cas ayant une CI satisfaite par la dérivée de v parvient à fit la droite souhaitée. Cependant, lorsqu'on choisit les CI uniquement sur la fonction, le réseau ne parvient pas à sortir la solution souhaitée. On utilisera ainsi uniquement les CI sur la fonction et la dérivée dans la suite.



(a) CI uniquement sur la fonction

(b) CI sur la fonction et la dérivée

FIGURE 8 – Fonction inverse de calibration renvoyée par le réseau dans le cas de $R = \alpha \ \forall E_t$, après 40 000 époques. Fig. 8a correspond aux CI : $(0.9, 0.9\times0.95), (0.09, 0.99\times0.95)$ sur la fonction. Fig. 8b correspond aux CI : $(0.9, 0.9\times0.95)$ sur la fonction, et , $(0.99, 0.95)$ sur la dérivée.

Il est également intéressant de vérifier si l'on aboutit bien sur les mêmes solutions pour des CI $(x_0, v(x_0))$, $(x_1, v'(x_1)$ ou $(x_0, v(x_0))$, $(x_0, v'(x_0)$.

E.6 Cas $R = \alpha$

Pour les hautes énergies, on dénote qu'expérimentalement R est constant $R = \alpha \ \forall E_t$. Ceci signifie que chaque maximum de distribution est reliée linéairement lors de la calibration par $\mu(E_t) = \alpha E_t$. Aussi, on remarque que pour $v(E_t) = \alpha E_t$, soit $v'(E_t) = \alpha$ et $v''(E_t) = 0$, l'équation est bien respectée :

$$0 - \frac{r_0^2}{\sigma^2}(r_0 E_t - r_0 E_t) = 0$$

En effet, ceci revient à choisir $r = R \ \forall E_t$. Il s'agit ici d'une remarque, et non pas d'une démonstration mathématique. Cette calibration revient à décaler toutes les gaussiennes d'un facteur alpha.

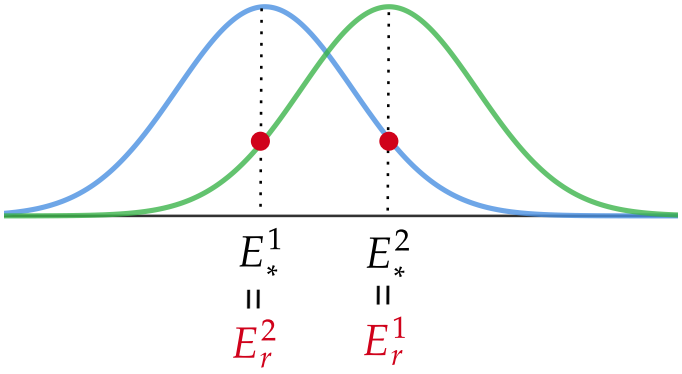


FIGURE 9 – Calibration en énergie pour deux mesures gaussiennes.

Fig. 9 illustre une façon d'appréhender le cas R constant. En effet, on considère une gaussienne centrée en E_*^1, et une gaussienne la chevauchant centrée en E_*^2. En se concentrant sur la première gaussienne, la calibration de l'énergie est telle que $u(E_*^1) = E_{c*}^1$. On peut aussi voir l'énergie E_*^1 comme une énergie mesurée dans la gaussienne 2, soit $E_r^2 = E_*^1$. Ainsi $u(E_*^1) = u(E_r^2) = E_{c*}^1 = E_{cr}^2$ car la fonction de calibration ne dépend pas de quelle gaussienne on étudie. Ainsi, lorsqu'on connait $R = \frac{E_*}{E_t} = \alpha$, i.e la fonction de calibration au centre de la gaussienne $u(E_*) = E_t = E_*/\alpha$, pour TOUTES les énergies centrales, c'est équivalent à connaitre la calibration pour un continuum de gaussienne. Ainsi, pour une gaussienne d'énergie centrale E_*, on peut voir ses énergies E_r comme étant le centre d'une gaussienne dont on connait la calibration i.e $E_c = E_r/\alpha$.

Pour résoudre ce cas de l'équation, via le réseau de neurone, il suffit d'injecter l'expression de $\mu(E_t) = \alpha E_t$, et de donner les CI suivante $v(0) = 0$ et $v(E_{inf}) = \alpha E_{inf}$. On peut ensuite vérifier que le réseau converge bien vers une droite de pente α.

Résultat :
On finit par bien et bien aboutir à une fonction linéaire, passant par les CI données.

E.7 Cas d'un R non constant

Dans le cas où la réponse correspond à ce qui est observé dans l'expérience, on choisit :

$$R(x) = \alpha \left(1 - \frac{1}{(\beta + x)^9}\right) \tag{22}$$

On peut donc injecter cette formule dans Eq. 12, pour laquelle on cherche donc des solutions. Pour des hautes énergies, R retrouve un comportement constant, ainsi, on s'attend à ce que v retombe sur sons comportement linéaire précédemment trouvé. Pour ce faire, on donne les CI suivante, $v(x_\infty^1) = \alpha x_\infty^1$ et $v(x_\infty^2) = \alpha x_\infty^2$, où x_∞^i sont deux très grandes énergies devant $x_0 = 20GeV/E_{norm}$.

On remarque que le cas $v(x) = \alpha x + b$, avec $\mu(x) = \alpha x + b$ est également solution. On peut prendre b tel que $b = \mu_0 - x_0\alpha$ pour un x_0 et μ_0 arbitraires. En prenant soin de changer les CI sur la fonction : $(x_i, v(x_i) = \alpha x_i + b)$. La condition sur la dérivée, elle, ne change pas. On a pu vérifier avec le réseau de neurones que ceci fonctionnait correctement.

E.8 Inversion de la fonction

À la sortie du premier réseau de neurones, on obtient la fonction $v(E)$, pour un ensemble de points $E \in [0, 1]$. On souhaite obtenir la fonction de calibration, $u(v(E)) = E$. Ainsi, une méthode serait de fournir à un second réseau les valeurs obtenues de $v(E)$ en entrée, reliée à leur valeur réelle de E. Le réseau ressortira une valeur $y_{pred} = u_{pred}(v(E))$ que l'on souhaite donc faire correspondre à E. On peut donc utiliser une fonction loss comparant les deux valeurs $(y_{pred} - E)^2$ que l'on cherche donc à minimiser. Une fois le minimum atteint, l'ensemble des valeurs de y_{pred} forme la fonction de calibration $u(E_r) = E_c$.

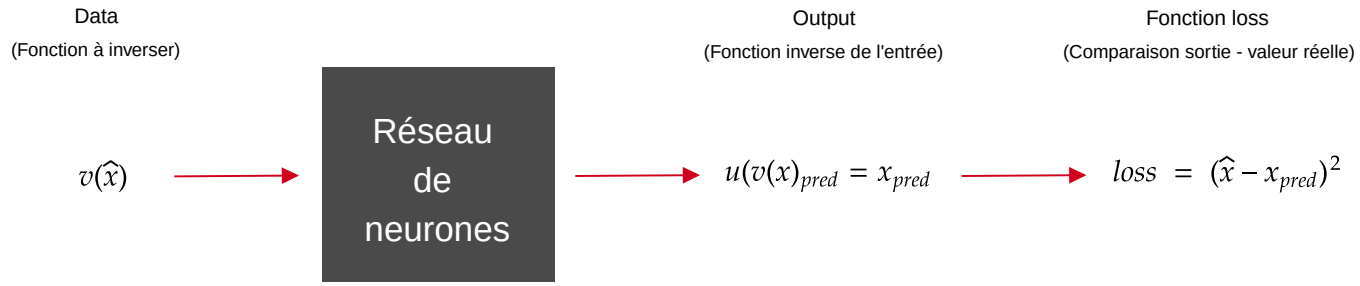
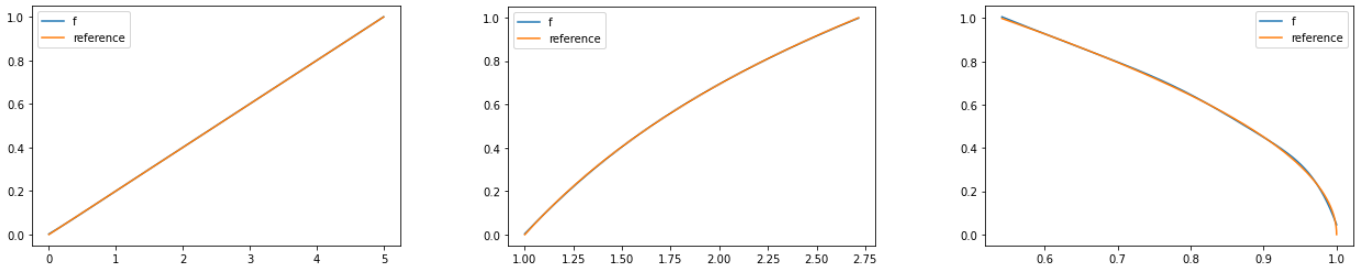


FIGURE 10 – Représentation schématique des principales étapes d'inversion de fonction par un réseau de neurones. On suppose connues un set de data $(v(x), x)$.

Fig. 10 illustre la méthode suivie pour inverser une fonction. On dispose d'une série de valeurs de la fonction $v(x)$, ainsi que de ses antécédents x. Nous cherchons à déterminer sa fonction inverse $u(X)$. En composant les deux fonctions, on a $u(v(x)) = x$. On utilise ainsi les valeurs connues de $v(x)$ en entrée du réseau. Nous souhaitons obtenir $u(X)$ en sortie du réseau. Ainsi, on cherche à minimiser la fonction $loss = (u(v(x))_{pred} - x)^2$, où x est connu. Le réseau va donc adapter ses paramètres pour pouvoir fournir, en sortie $x_{pred} = u(v(x))$, le plus fidèle au x réel. En somme, le réseau renvoie l'antécédent d'une fonction donnée en entrée, i.e. sa fonction inverse.



(a) $v(x) = 5 \times x$, $u(x) = x/5$ (b) $v(x) = e^x$, $u(x) = \ln x$ (c) $v(x) = \cos x$, $u(x) = \arccos x$

FIGURE 11 – Exemples d'inversion de fonctions via le réseau de neurones présenté dans Fig.10. La fonction $v(x)$ est donnée en entrée. La fonction inverse $u(x)$ est plot en bleue, avec sa forme analytique de référence en orange.

En utilisant cette méthode, on a pu vérifier le fonctionnement de notre modèle, pour différentes fonctions dont on connait la fonction inverse analytiquement.

Fig. 11 donne quelques exemples de fonctions dont on connait l'inverse. Le résultat fournit en sortie du réseau au bout d'un grand nombre d'époques, colle parfaitement avec la fonction inverse théorique.

Dans notre cas, la fonction de calibration inverse $v(x)$ est strictement croissante, et est donc bijective, et donc inversible. On ne parvient cependant pas à inverser la fonction suffisamment précisément pour l'utiliser pour calibrer les énergies.

F Résolution de l'équation différentielle sur la fonction de calibration

Pour obtenir la fonction de calibration u, on peut directement chercher à résoudre Eq. 3, au lieu de l'équation différentielle sur v. Cela évite de perdre en précision lors de l'inversion.

F.1 Vérification de convergence

Il est nécessaire de vérifier si la solution obtenue par le réseau de neurone correspond bien à la fonction de calibration. Pour cela, on peut comparer la solution directement à une formule analytique, dans un cas où une solution exacte existe, comme fait dans F.2. Un autre moyen est de générer un set de mesure d'énergie, puis vérifier sa calibration, comme illustré dans Fig. 12.

Soit une fonction de réponse non calibrée R connue. Celle-ci relie l'énergie centrale de la gaussienne mesurée $E_* = \mu(E_t)$ à l'énergie réelle E_{true} pour laquelle la distribution est maximale. On fera l'approximation de mesures suivant une gaussienne, de même pour l'énergie réelle. Ainsi, E_* et E_t peuvent être approximées par les moyennes des gaussiennes réelles et mesurées.

On génère par monte-carlo des mesures suivant une gaussienne centrée en R, pour un certain E_{true}. On répète cela pour un ensemble d'énergies E_{true}, obtenant donc un ensemble de gaussiennes dont les valeurs sont distribuées suivant une gaussienne selon un axe $r_{reco} = E/E_{true}$, centrées en $R = E_*/E_{true}$. En multipliant chaque r_{reco} par le E_t attribué à la gaussienne, on obtient E_{reco}. Une fois normalisées, on donne ces énergies en entrée au réseau de neurones, qui donne l'énergie calibrée en sortie. On calcule la moyenne de la distribution obtenue en sortie, correspondant donc à E_{true}/E_{norm}. On peut calculer la réponse calibrée pour cette distribution avec $R_{calib} = \frac{E_{max}}{E_t}$, où E_{max} est l'énergie calibrée pour laquelle la distribution est maximale après passage dans le RN. Si $R_{calib} = 1$, alors l'énergie a été correctement calibrée. On répète cela pour chaque gaussienne.

On peut ensuite tracer la réponse calibrée en fonction de l'énergie E_{true}, et vérifier que celle-ci soit constante à 1, aux fluctuations statistiques près.

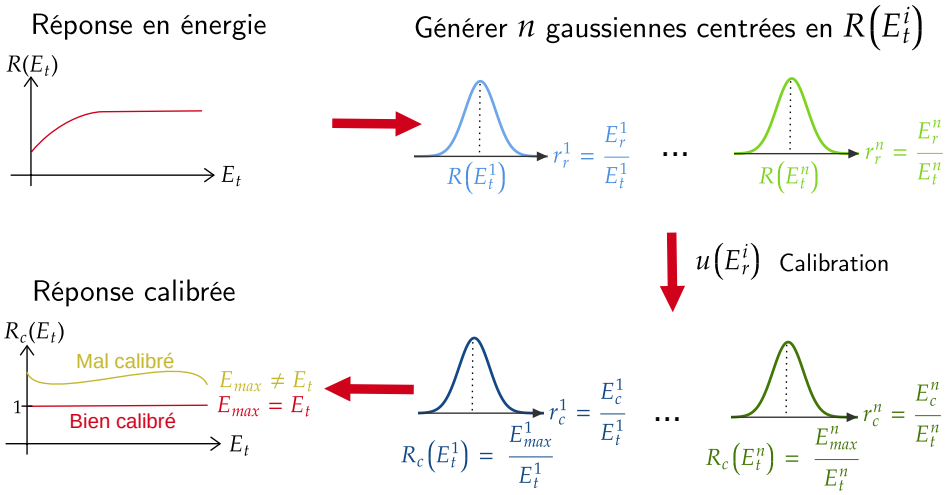


FIGURE 12 – Schéma explicatif de la vérification de la convergence vers la solution de la fonction de calibration.

F.2 Solution analytique connue : $R = \alpha + \frac{b}{x}$

On peut vérifier que l'on parvient à converger vers une solution pour un cas où l'on connait la solution analytique. On choisit une fonction de réponse non calibrée suivante :

$$R(x) = \alpha + \frac{b}{x} \tag{23}$$

où b est une constante que l'on détermine via $R_0 = R(x_0 = 20/E_{norm}) = 0.8$, soit $b = (R_0 - alpha) \times x_0$. On note que cette forme est juste utile à des fins de vérifications. En effet, cette fonction ne correspond pas à la fonction de réponse mesurée à haute énergie, puisque $R(1) \neq \alpha$.

Dans ce cas on a $v = \alpha x + b$ qui est solution de l'équation différentielle Eq. 12. On peut inverser la fonction et vérifier que dans ce cas la fonction de calibration suit :

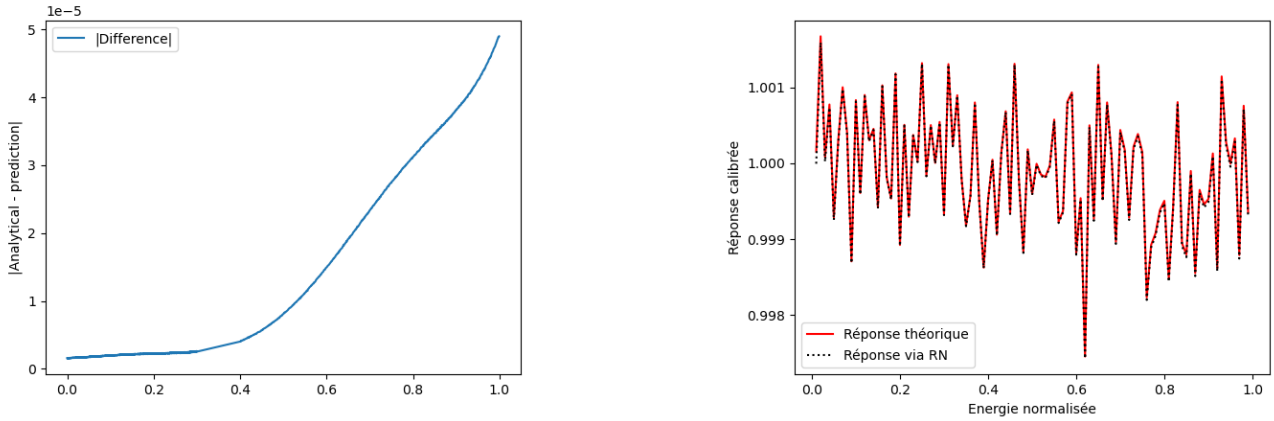
$$u = \frac{x - b}{\alpha} \tag{24}$$

Il est important d'étudier l'influence du choix des CI sur la résolution de l'équation différentielle nous intéressant.

F.2.1 CI écartées

Prendre des CI pour des x écartés permet d'étaler les contraintes sur la fonction sur l'axe des x, et donc facilite la convergence vers la solution. Il s'agit ainsi du cas le plus simple testable, i.e. une fonction linéaire avec des CI prises de part et d'autres de l'intervalle $[0, 1]$.

On choisit les CI sur la fonction $u(x_\infty^2 = 990/E_{norm}) = \frac{x_\infty^2 - b}{\alpha}$ et sur la dérivée $u'(x_0) = 1/\alpha$.



(a) $|u_{théorique} - u_{RN}|$ (b) Réponse calibrée

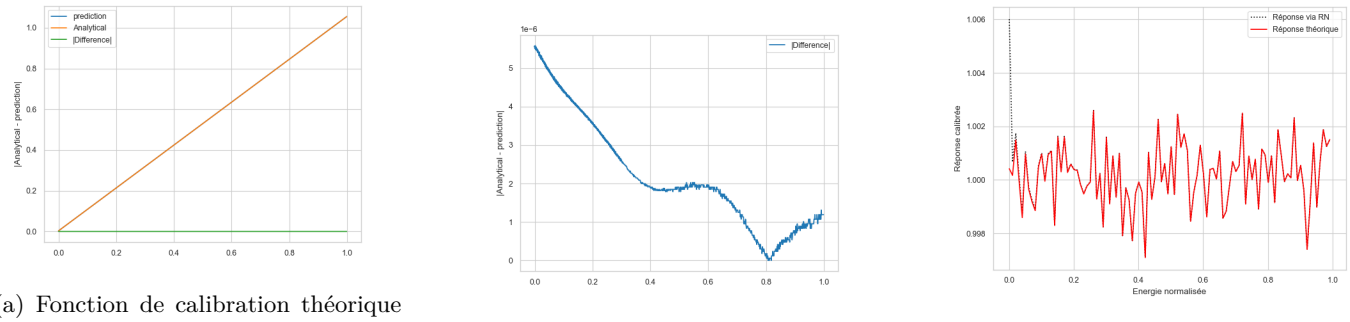
FIGURE 13 – Test de la convergence du réseau de neurones vers la solution souhaitée.La figure 14b présente la valeur absolue de la différence entre la fonction de calibration théorique et celle obtenue en sortie du réseau de neurones. Le panel de droite présente la réponse en énergie une fois calibrée, via la fonction théorique et celle obtenue par le réseau de neurones entrainé. Pour la calibration, on utilise des gaussiennes ayant une énergie E_{true} entre 1 et 1000GeV.

Fig. 13 présente les résultats obtenus pour cette fonction de réponse et ces CI. Fig. 14b illustre la très faible différence entre la fonction de calibration théorique, et celle en sortie du RN, de l'ordre de 10^{-6}. Cette précision permet de calibrer au mieux l'énergie. En effet, la fonction de réponse calibrée via la fonction théorique colle parfaitement avec celle calibrée via le RN. De plus, la fonction de réponse est bien autour de la valeur 1, et ses écarts ne sont uniquement statistiques.

On parvient ainsi à obtenir la fonction de calibration, pour un cas linéaire avec des CI écartées.

F.2.2 CI rapprochées

On souhaite tester la convergence de notre réseau vers une solution connue, avec des CI plus proches de ce qu'on utilisera dans le cas Physique. On utilise désormais les CI sur la fonction $u(x^2_\infty = 990/E_{norm}) = \frac{x^2_\infty - b}{\alpha}$ et sur la dérivée $u'(x^1_\infty = 950/E_{norm}) = 1/\alpha$.



(a) Fonction de calibration théorique et en sortie du NN (b) $|u_{théorique} - u_{RN}|$ (c) Réponse calibrée

FIGURE 14 – Test de la convergence du réseau de neurones vers la solution souhaitée.Fig. 20a présente la fonction de calibration par rapport à l'énergie normalisée. La figure 14b présente la valeur absolue de la différence entre la fonction de calibration théorique et celle obtenue en sortie du réseau de neurones. Le panel de droite présente la réponse en énergie une fois calibrée, via la fonction théorique et celle obtenue par le réseau de neurones entrainé. Pour la calibration, on présente les résultats pour des énergies E_{true} comprises entre 1 et 1000 GeV.

Fig. 14 présente les résultats du réseau de neurones, auquel on donne des CI rapprochées. On obtient bien une fonction de calibration linéaire, comme illustré dans Fig.20a. La Fig. 14b présente de nouveau la différence entre la théorie et la fonction en sortie du réseau, de nouveau très faible, de l'ordre de 10^{-6}. Dans Fig. 15d, on remarque que la fonction de calibration permet d'obtenir exactement les mêmes résultats que la théorie à partir d'environ $10GeV$. Pour des énergies plus faibles, la réponse calibrée par le réseau de neurone s'écarte légèrement de la théorie, mais reste très proche de 1.

Ainsi, on parvient également à obtenir la fonction de calibration en utilisant des CI proches sur l'axe des x.

F.3 Réponse réaliste

Pour des jets de taille moyenne, la réponse en énergie est donnée par

$$R(x) = \alpha \left(1 - \frac{1}{(\beta + x)p}\right). \tag{25}$$

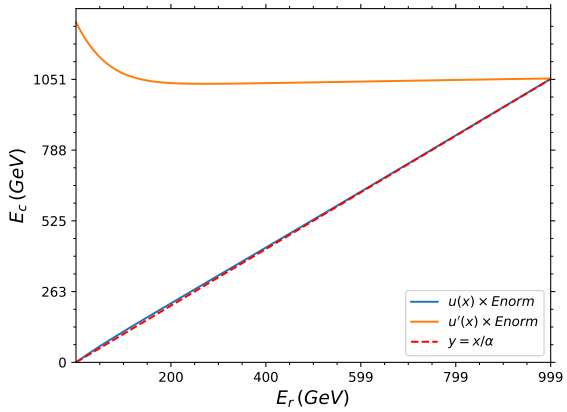
Comme illustré dans Fig. 7a, le cas $p = 9$ satisfait à ce qui est observé dans l'expérience. Aussi, on ajoute une dépendance en énergie à $\sigma(E)$, afin de coller aux observations.

$$\sigma/E = A + B/E \leftrightarrow \sigma = Ax + B$$

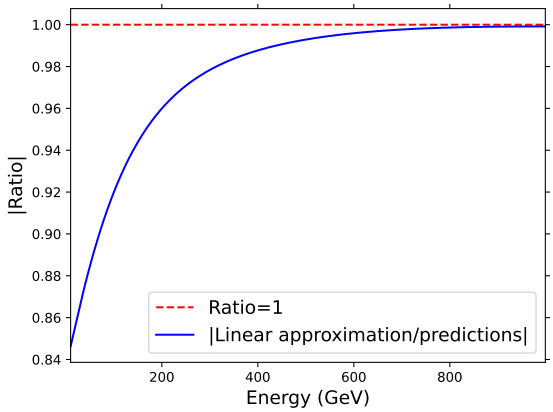
avec $A = 0.1$. Expérimentalement on obtient $(\sigma/E)\,(20GeV) = 0.3$. Ainsi $B = (0.3 - 1) \times 20GeV = 4GeV$. Pour obtenir l'écart type en terme d'énergies normalisées, on applique E_{norm} :

$$\sigma_{norm} = \frac{\sigma}{E_{norm}} = Ax + \frac{B}{E_{norm}} = Ax + b \tag{26}$$

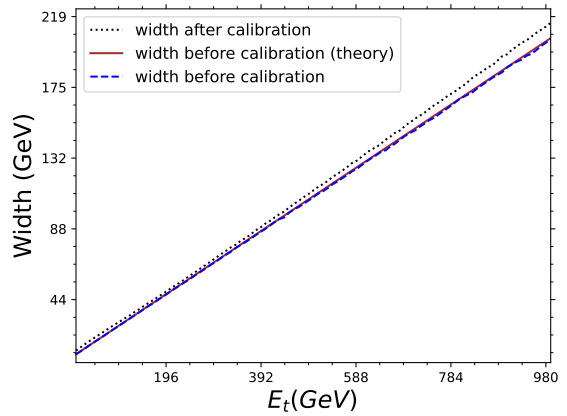
avec $b = B/E_{norm}$. Cette dépendance permet d'obtenir un écart type faible à basse énergie, et plus élevé à haute.



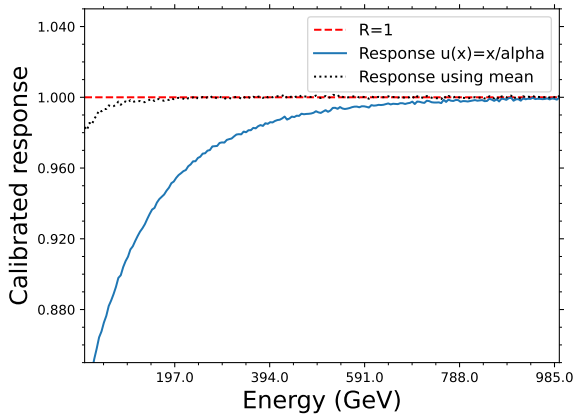
(a) $u(x)$ en sortie du NN



(b) $|u_{linéaire}/u_{RN}|$



(c) Largeur de la distribution



(d) Réponse calibrée

FIGURE 15 – Test de la convergence du réseau de neurones vers la solution souhaitée.Fig. 20a présente la fonction de calibration par rapport à l'énergie normalisée. La figure 15b présente la valeur absolue du ratio entre la fonction de calibration approximée linéaire $u(x) = x/\alpha$ et celle obtenue en sortie du réseau de neurones. Le panel de droite présente la réponse en énergie une fois calibrée, via la fonction théorique et celle obtenue par le réseau de neurones entrainé. Fig. 15c illustre l'évolution de la largeur de la distribution, i.e. $\simeq 2\sigma$, avant et après la calibration. Pour la calibration, on présente les résultats pour des énergies E_{true} comprises entre 10 et 1000GeV.

Fig. 15 présente les résultats obtenus pour une réponse réaliste. La fonction de calibration obtenue tend bien vers une réponse linéaire en $y = x/\alpha$ à hautes énergies. Comme le montre Fig. 15b, la fonction obtenue en sortie du RN est diffère légèrement de l'approximation linéaire à basse énergie. Bien que faible, cette différence a un impact important

dans la qualité de la calibration. En effet, en utilisant la fonction obtenue en résolvant l'équation différentielle, on parvient à obtenir R_{calib} qui reste près de $R = 1$ à 2% près, ce qui n'est pas le cas en utilisant la fonction linéaire. La largeur de la distribution des énergies calibrées est plus grande que celle des énergies mesurées. En effet, la fonction de calibration reste supérieure à la droite $y = x$, et cet écart augmente pour les hautes énergies. Ainsi, les distributions sont étirées lors de la calibration.

On parvient ainsi à résoudre l'équation différentielle assez précisément pour pouvoir calibrer des énergies mesurées, et ce, pour un cas correspondant à un cas expérimental.

F.4 Réponse en masse

Expérimentalement, la réponse en masse est linéaire, soit :

$$R(E) = \alpha_{SI} E + b = \alpha_{SI} \times E_{norm} \times x + b = \alpha_m \times x + b \tag{27}$$

avec $\alpha_m = \alpha_{SI} E_{norm}$. Via l'expérience, on a $\alpha_{SI} = \frac{1.5-0.8}{4500-500} = 1.75e{-}4$. Aussi, on peut utiliser $R(E = 4500GeV) = 1.5$ pour obtenir $b = 1.5 - \alpha_{SI} \times 4500 = 0.7125$.

Cette réponse complique le choix des CI. On peut toujours fixer $u(0) = 0$, puisque la réponse en énergie ne change pas le fait qu'on peut approximer l'absence d'énergie mesurée, à l'absence d'énergie déposée. Ici, on ne connait pas de solution analytique à haute énergie satisfaisant l'équation différentielle. Ainsi, on ne peut plus imposer un u linéaire à haute énergie. On fait donc l'approximation que lorsque $R(x_1) = 1$, alors, il n'est pas nécessaire de calibrer cette énergie, soit $u(x_1) = x_1$.
On peut ainsi déterminer x_1 via $R(x_1) = 1 = \alpha_m x_1 + b \rightarrow x_1 = \frac{1-b}{\alpha_m}$. En énergie, on a $E_1 = 1642.86GeV$.

Pour la réponse en masse, l'expérience montre une dépendance linéaire pour les hautes énergies :

$$\sigma/E = A + B/E + C \times E$$

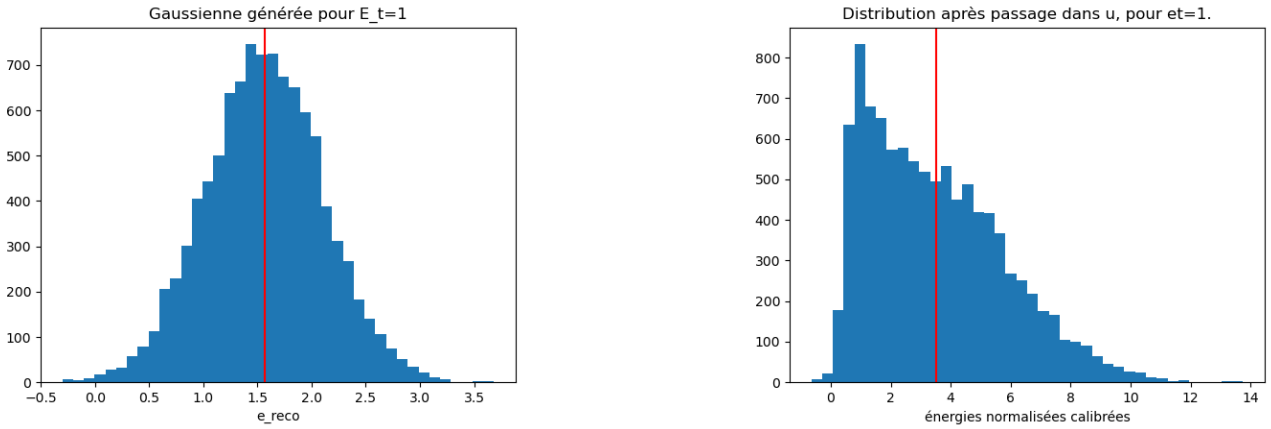
De la même manière, on a :

$$\sigma_{norm} = \frac{\sigma}{E_{norm}} = \sigma_a x + \frac{B}{E_{norm}} + CxE = \sigma_a x + \sigma_b + C \times E_{norm} x^2 = \sigma_a x + \sigma_b + \sigma_c x^2 \tag{28}$$

Avec $\sigma_a = 0.1$, $\sigma_b = B/E_{norm}$, $\sigma_c = C \times E_{norm}$. Expérimentalement, on obtient C, qui est le coefficient directeur de la droite à haute énergie de l'écart relatif, avec les valeurs suivantes $C = \frac{0.6-0.3}{4500-1200}$.
On remarque σ_{norm} dépend des E_{norm}. Plus ce dernier est faible, plus σ sera faible à haute énergie, et plus élevé à basses énergies.

F.4.1 Résolution avec un NN

On souhaite résoudre l'équation différentielle pour la fonction de masse, et calibrer l'énergie, via le réseau de neurones.



(a) Gaussienne avant calibration

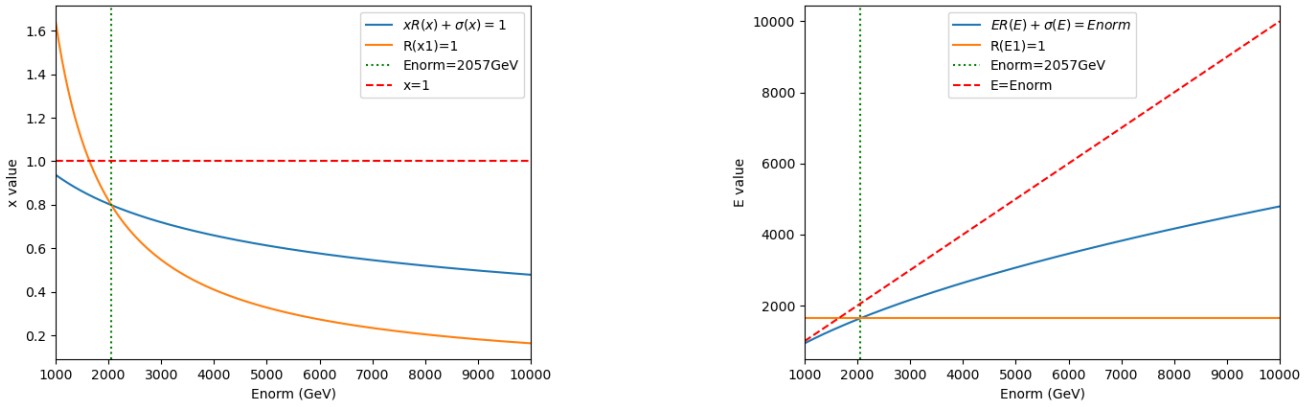
(b) Après calibration

FIGURE 16 – Fig. 16a correspond à la distribution avant la calibration, pour un $E_{true} = 1$. Fig. 16b correspond à la distribution après passage par la fonction de calibration obtenue par le NN.

Plusieurs soucis se révèlent dans ce cas :

— La réponse peut être plus grande que 1 dans l'intervalle étudié. Ainsi, lors de la calibration, on peut obtenir une gaussienne centrée au-delà de 1. Le réseau ne pourra pas la calibrer efficacement, puisqu'il n'est pas entrainé pour ces valeurs-là de x. C'est ce qui est illustré dans Fig. 16.

— La fonction sigma implique des valeurs plus élevées d'écart type. Ainsi, les gaussiennes peuvent être particulièrement large à hautes énergies. On peut donc avoir des gaussiennes centrées dans l'intervalle d'étude, mais donc les queues dépassent au-delà de 1, voire dans les négatifs. De nouveau, le réseau ne pourra pas calibrer correctement ces distributions, et faussera donc la réponse.



(a) Énergies normalisées

(b) Énergies en GeV

FIGURE 17 – Délimitation des zones intéressantes pour l'entrainement du réseau et la calibration. L'aire sous la courbe bleue correspond aux x/E pour lesquels on peut générer des gaussiennes et les calibrer avec le réseau. La courbe orange présente le x_1 ou E_1 de la CI initiale.

Pour qu'on puisse calibrer correctement une gaussienne générée, il faut que son centre appartienne à l'intervalle d'entrainement du réseau. Si on utilise des données normalisées $x < 1$, alors cela revient à $xR(x) < 1$. Pour obtenir une bonne calibration, on peut étendre cette condition à ce que la queue de la gaussienne également soit à l'intérieur de l'intervalle, soit :

$$xR(x) + \sigma(x) < 1. \tag{29}$$

Avec eq. 28 et eq. 27, l'égalité est atteinte lorsque :

$$x^2(\alpha_m + \sigma_c) + x(b + \sigma_a) + (\sigma_b - 1) = 0 \tag{30}$$

Les racines positives de ce polynôme suivent :

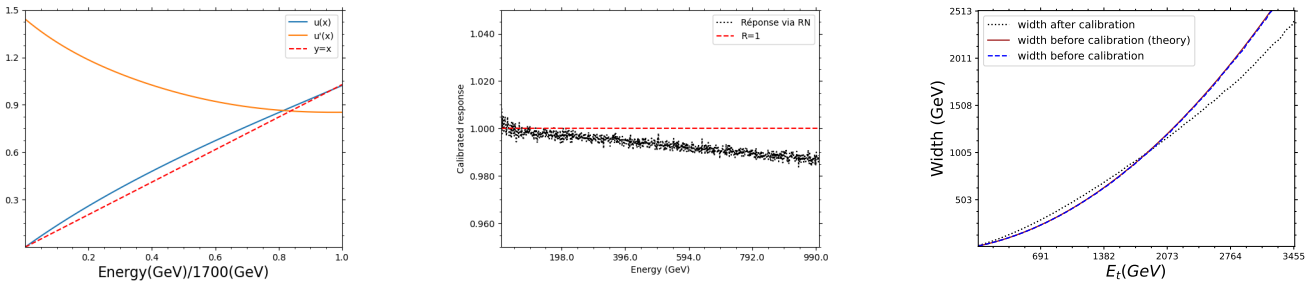
$$x = \frac{-(\sigma_a + b) + \sqrt{(\sigma_a + b)^2 - 4(\sigma_b - 1)(\alpha_m + \sigma_c)}}{2(\alpha_m + \sigma_c)} \tag{31}$$

Fig. 17a résume les contraintes qui s'appliquent sur les x et la normalisation de notre problème. Il est nécessaire que la CI x_1 appartiennent à l'intervalle d'étude. Ainsi, il faut que la courbe orange soit en dessous de $x = 1$, soit $E_{norm} > E_1 = 1643GeV$. Les x situés sous la courbe bleu sont dont la calibration sera théoriquement possible par le réseau de neurones, puisque satisfaisant $xR(x) + \sigma(x) < 1$. Ainsi, si on souhaite uniquement étudier les énergies les plus basses, on peut choisir un E_{norm} quelconque, tant qu'il satisfait $E_{norm} > E_1$.
Si on souhaite pouvoir calibrer des énergies plus hautes que E_1, alors il est nécessaire que la courbe orange soit en dessous de la bleue, i.e. que x_1 appartiennent à l'intervalle pour lesquels les x sont calibrables. Cette condition est satisfaite pour $E_{norm} > 2057GeV$. Plus ces deux courbes sont écartées, plus on pourra étudier des énergies éloignées de x_1. Cependant, l'écart augmente avec E_{norm}, mais ce dernier diminue la proportion de l'intervalle utile pour la calibration (délimité par la courbe bleue). Ainsi, sauf si l'on souhaite étudier des énergies encore plus hautes, il ne semble pas nécessaire d'utiliser des énergies de normalisation plus hautes que $E_{norm} = 5000GeV - 6000GeV$.
Fig. 17b présente les mêmes contraintes, mais en utilisant des énergies non normalisées. La pente, $x = 1$ délimitant la zone où les données sont normalisées, est remplacé par son équivalent $E = E_{norm}$. On remarque que plus E_{norm} est élevé, plus on peut calibrer à des hautes énergies.
On note que cette discussion concerne uniquement des données normalisées, comprises entre 0 et 1. On pourrait également choisir d'entrainer le réseau sur des données plus grandes que 1, agrandissant l'intervalle d'étude possible

pour une même normalisation. Ainsi, pour un réseau prenant en entrée des $0 < x < c$, il faut que $c \times E_{norm} > E_1$. Pour un même nombre de data d'entrainement, une borne supérieure proche de x_1 permettra une meilleure convergence vers la solution aux basses énergies, mais rendra impossible l'étude de la calibration pour les énergies où $R > 1$. Pour pouvoir étudier la calibration pour des énergies supérieures à E_1 via cette méthode, on pourrait entrainer le réseau entre 0 et 2, avec $Enorm = 1700$. Puis calibrer pour des E_t entre 0 et 1.2.
En résumé, ces graphiques sont utiles pour déterminer le E_{norm} à choisir lorsqu'on souhaite utiliser des données normalisées, pour une certaine plage d'énergie que l'on veut pouvoir calibrer.

F.4.2 Etude pour $E_{norm} = 1700GeV$

Dans ce cas particulier, x_1 est supérieur aux x satisfaisant eq. 29. Ainsi, on ne peut pas étudier la calibration des énergies plus hautes que E_1. Plus précisément, seul les $x < 0.84$ peuvent être calibrés ici, soit $E = 1424$. Ainsi, notre étude de $10GeV$ à $1000GeV$ est totalement compatible.



(a) Plot (b) R_{calib} (c) Largeur de la distribution

FIGURE 18 – Résultats obtenus pour des données normalisées et $Enorm = 1700GeV$). Fig. 20a correspond à la fonction de calibration obtenue en sortie du NN. Fig. 20b présente R_{calib}, après passage dans le réseau de neurones. Fig. 20c illustre la largeur pour un certain E_t avant et après calibration.

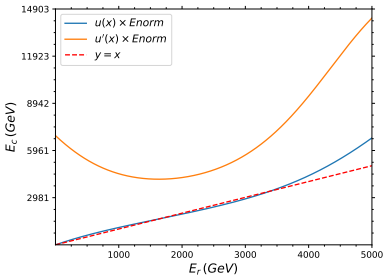
Fig. 18 présente les résultats pour cette normalisation. On parvient ici à atteindre $loss \simeq 4e-10$. Les résultats sont donc très près de ce que l'on obtiendrait avec une solution exacte. Bien que R_{calib} ne soit pas totalement constant, on parvient à calibrer les énergies avec une précision comprise entre 0% et 2%. On parvient donc bien à calibrer ces énergies pour cette normalisation, sur la plage attendue.
La convergence vers la solution étant excellente, le fait que R_{calib} diffère de 1 peut être dû à l'approximation de la CI en x_1. Celle-ci implique surement que u passe en ce niveau à trop basse énergie. Cependant, ce choix de Ci permet tout de même d'atteindre une bonne convergence à ce niveau.
Fig. 20c présente la largeur de la distribution avant et après la calibration. Pour des énergies faibles devant E_1, la fonction de calibration est au-dessus de $y = x$. Ainsi les énergies mesurées sont toutes augmentées lors de la calibration. Tant que l'écart avec la droite $y = x$ augmente, les gaussiennes mesurées sont étirées lors de la calibration. Une fois l'écart maximum passé, le côté droit de la gaussienne est moins décalé vers les hautes énergies que celui de gauche, ainsi la distribution est resserrée. Ainsi, à partir d'une certaine énergie, la distribution calibrée est plus fine que celle mesurée.

F.4.3 Étude pour $Enorm = 5000GeV$

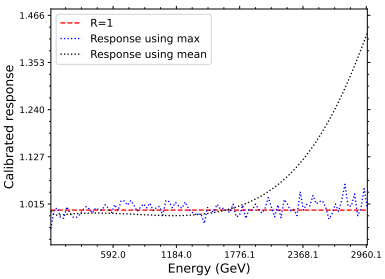
Dans ce cas $x_1 \simeq 0.32$ est au milieu de l'intervalle d'étude où la borne max est environ à 0.6. On peut ainsi étudier la calibration des énergies bien plus haute, pour lesquelles $R > 1$, impliquant ainsi une calibration différente des cas rencontrés jusqu'à maintenant.

Le réseau de neurones semble fournir une solution s'éloignant de ce qu'on attendait à hautes énergies. Ainsi, des résultats provocants, ou non, de l'overfit à haute énergie, sont présentés dans la suite.

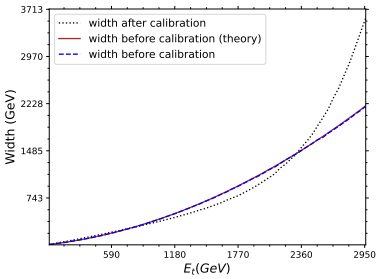
Sans overfit : data homogènes.



(a) Plot



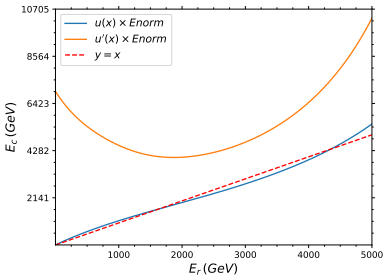
(b) R_{calib}



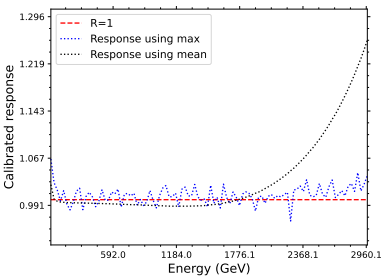
(c) Largeur de la distribution

FIGURE 19 – Résultats obtenus pour des données normalisées et $Enorm = 5000GeV$). Fig. 20a correspond à la fonction de calibration obtenue en sortie du NN. Fig. 20b présente R_{calib}, après passage dans le réseau de neurones. Fig. 20c illustre la largeur pour un certain E_t avant et après calibration.

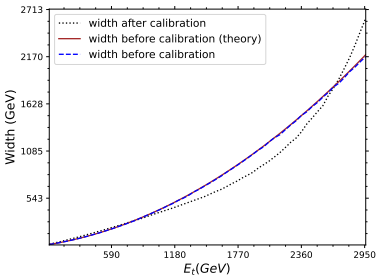
Avec overfit : on ajoute des valeurs pour les hautes énergies, et pour les très basses, afin de limiter la divergence à haute énergie.



(a) Plot



(b) R_{calib}



(c) Largeur de la distribution

FIGURE 20 – Résultats obtenus pour des données normalisées et $Enorm = 5000GeV$). Fig. 20a correspond à la fonction de calibration obtenue en sortie du NN. Fig. 20b présente R_{calib}, après passage dans le réseau de neurones. Fig. 20c illustre la largeur pour un certain E_t avant et après calibration.

On observe que la fonction de calibration diverge à haute énergie, ce qui est inattendu. En effet, on a $R > 1$ pour les hautes énergies, donc on s'attend à ce que le réseau compense, en renvoyant des énergies plus faibles que celles en entrée, ce qui ne semble pas être le cas ici. Même en essayant de provoquer de l'overfitting, le réseau diverge toujours (bien que légèrement moins). Ce problème se voit également avec la largeur de la distribution calibrée qui redevient plus grande que celle des énergies mesurées. Ceci est totalement logique au vu du u obtenu en sortie du réseau, mais ne correspond pas à ce qu'on s'attendait à voir pour le u espéré.

Cependant, cela n'empêche pas d'avoir de bons résultats pour la calibration, comme on peut le voir sur *Rcalib* qui reste autour de 1 à 5% près, avec un résultat légèrement meilleur pour le cas sans overfitting. En comparant les résultats obtenus avec la moyenne et le max (combiné avec la moyenne pondérée sur 30% de la distribution), on voit que les résultats sont meilleurs avec la moyenne à basse énergie, mais moins bon à haute énergie. Ceci est dû au fait que la moyenne utilise toutes les valeurs disponibles, limitant donc les erreurs statistiques, et à l'apparition d'asymétrie dans la distribution à haute énergie.

Pour la réponse en masse, nous sommes donc capables de calibrer avec précision des énergies plus faible que $1700GeV$, où on obtient les résultats attendus. Pour les énergies plus hautes, la fonction finit par dévier du résultat prédit, mais la calibration reste précise.