
# SUSY

Gaétan Grima,Kejian Wang

2020/12

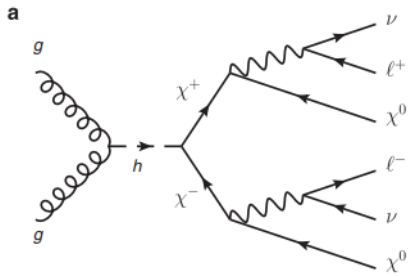
# Table des matières

# 1 Introduction

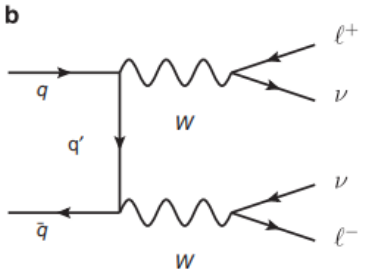
SuSy est une abréviation fréquente pour la supersymétrie, une théorie en physique des particules.

Pour le Machine-Learning,la tâche est une classification [1] de référence consiste à faire la distinction entre un processus où de nouvelles particules supersymétriques sont produites "signal", conduisant à un état final, dans lequel certaines particules sont détectables et d'autres sont invisibles pour l'appareil de mesure, et un processus de "background" avec les mêmes particules détectables mais avec moins de particules invisibles et des features cinématiques différentes.

Le processus de "signal" est la production de particules supersymétriques chargées électriquement qui se désintègrent en bosons W et particules supersymétriques neutres qui sont invisibles pour le détecteur. Le processus de "background" est la production de paires de bosons W, qui se désintègrent en leptons chargés et en neutrinos invisibles.




(a) signal




(b) background

La tâche de classification nécéssite de distinguer ces deux processus en utilisant les mesures de l' impulsion de lepton chargés et l'impulsion transverse perdue. Elle est utilisée comme benchmark pour mesurer l'efficacité d'un algorithme de classification de particules.

# 2 Contexte

## 2.1 Supersymétrie

La supersymétrie[2]est une symétrie supposée de la physique des particules qui postule une relation profonde entre les particules de spin demi-entier (les fermions) qui constituent la matière et les particules de spin entier (les bosons) véhiculant les interactions. Dans le cadre de la SuSy, chaque fermion est associé à un « superpartenaire » de spin entier, alors que chaque boson est associé à un « superpartenaire » de spin demi-entier.

L'un des grands intérêts de la supersymétrie, au niveau phénoménologique, vient de la stabilisation du boson de Higgs, et donc la hiérarchie des masses des particules élémentaires. Les corrections radiatives dans le modèle standard sont considérées comme bien maîtrisées pour les fermions et pour les bosons de jauge. Pour les fermions, les corrections sont logarithmiques et surtout proportionnelles à la masse de la particule, ce qui assure l'ordre de grandeur. Pour une particule de masse nulle, elles sont nulles, et on retombe sur un modèle avec une symétrie supplémentaire qui protège les fermions : la symétrie chirale. Pour les bosons de jauge, l'invariance de jauge assure le même genre de « protection » aux masses. On constate que pour les deux cas, une « symétrie » empêche les corrections d'atteindre des échelles trop importantes, puisqu'elles doivent être proportionnelles au « degré de brisure » de la symétrie.

Or, il existe dans le modèle standard un autre type de particule, le boson de Higgs, qui est introduit afin d'expliquer pourquoi certains bosons acquièrent des masses et brisent ainsi

la symétrie de l'interaction. Il s'avère que la masse du Higgs est dangereusement divergente, puisqu'elle dépend quadratiquement de l'échelle d'énergie.

Mais le vrai souci vient du fait que le modèle standard n'est considéré par la majorité des physiciens que comme un modèle à basse énergie qui doit, à partir d'une certaine échelle, donner la main à un autre modèle qui inclurait plus de phénomènes. Ceci implique que les divergences ne peuvent pas être simplement absorbées dans la procédure de renormalisation, puisqu'on se retrouve très rapidement dans des régimes énergétiques qui ne sont pas censés être décrits par le modèle standard , alors qu'on se place justement à des énergies correspondant au modèle standard.

La supersymétrie est une des solutions proposées pour atténuer l'effet de ces corrections, en introduisant une symétrie entre bosons et fermions. Cette symétrie assure en effet que pour une particule de spin $j$, on peut avoir une particule de spin $j \pm \frac{1}{2}$ (suivant le cas), mais surtout que ces particules qui sont liées ainsi auront des contributions opposées, qui finissent donc par s'annuler.

En 2015, cependant, aucun « super-partenaire » des particules connues n'a encore été observé. Si elle existe, la SuSy doit donc être une symétrie brisée : ceci implique en particulier que les « super-partenaires » doivent avoir des masses différentes de celles de leurs partenaires et qu'il est nécessaire de considérer des phénomènes à des échelles d'énergie élevées afin de restaurer et voir réapparaitre cette symétrie. Le Grand collisionneur de hadrons (LHC), mis en route pendant l'été 2009, devrait permettre de vérifier ou d'invalider l'hypothèse de l'existence de la supersymétrie (en juillet 2012 le LHC a mis en évidence ce qui ressemble à 99,9999 % à un boson de Higgs).

## 2.2 Boson de Higgs

Le boson de Higg[3], connu aussi sous d'autres noms comme boson BEH ou particule de Dieu, est une particule élémentaire dont l'existence, postulée indépendamment en 1964 par François Englert et Robert Brout, par Peter Higgs et par Gerald Guralnik, Carl Richard Hagen et Thomas Kibble, permet d'expliquer la brisure de l'interaction unifiée électrofaible en deux interactions par l'intermédiaire du mécanisme de Brout-Englert-Higgs-Hagen-Guralnik-Kibble et d'expliquer ainsi pourquoi certaines particules ont une masse et d'autres n'en ont pas. Son existence a été confirmée de manière expérimentale en 2012 grâce à l'utilisation du LHC et a conduit à l'attribution du prix Nobel de physique à François Englert et Peter Higgs en 2013.

Le boson de Higgs, quantum du champ de Higgs, confère une masse non nulle aux bosons de jauge de l'interaction faible (bosons W et boson Z), leur conférant des propriétés différentes de celles du boson de l'interaction électromagnétique, le photon.

Cette particule élémentaire constitue l'une des clefs de voûte du modèle standard de la physique des particules. La connaissance de ses propriétés peut par ailleurs orienter la recherche au-delà du modèle standard et ouvrir la voie à la découverte d'une nouvelle physique, telle que la supersymétrie ou la matière noire.

Le 4 juillet 2012, le CERN annonce, lors d'une conférence, avoir identifié, avec un degré de confiance de 99,99997 % , un nouveau boson dans un domaine de masse de l'ordre de 125–126 GeV$c^{-2}$, qui paraît compatible avec celui du boson de Higgs. L'annonce est suivie, le 17 septembre 2012, par la publication de deux articles dans la revue Physics Letters B. Le 15 mars 2013, le CERN confirme que, selon toute vraisemblance, il s'agit bien du boson de Higgs.

## 2.3 Problème du LHC

La grande majorité des collisions de particules ne produisent pas de particules exotiques. Par exemple, si le grand collisionneur de hadrons (LHC) produit environ $10^{11}$ collisions par

heure,environ 300 de ces collisions aboutissent à un boson de Higgs,en moyenne. Par conséquent, une bonne analyse des données dépend de la distinction collisions qui produisent des particules d'intérêt (signal) à partir de celles produisant d'autres particules (background). [1]

Même si des particules intéressantes sont produites, les détecter pose des défis considérables. Elles sont trop petites pour être directement observées et se désintègrent presque immédiatement en d'autres particules. Bien que les nouvelles particules ne puissent pas être observées directement, les plus légeres des particules stables vers lesquelles elles se désintègrent, appelées "decay products", peuvent être observées. Plusieurs couches de détecteurs entourent le point de collision à cet effet.

Quand chaque produit de désintégration traverse ces détecteurs, il interagit avec eux d'une manière qui permet de mesurer sa direction et son moment cinétique.

Le coût de construction et d'opération des accélérateurs de particule est très élevé, donc toute puissance de classification supplémentaire sur les données extraites des collisions est très appréciée.les techniques actuelles utilisées dans la physique des hautes énergies ne peuvent pas capturer toutes les informations disponibles, même lorsqu'elle est renforcée par des fonctionnalités 'physics-inspired' construites manuellement. Cela réduit enormément la puissance du collisionneur pour découvrir de nouvelles particules

# 3 Outils

## 3.1 Pytorch

Pour aborder ce problème de classification, nous allons utiliser l'outil Pytorch pour générer un réseau de neurones profond , pytorch inclut les modules nécéssaires pour créer entraîner et tester le réseau de neurones simplement. Il nous permet aussi notamment d'utiliser la puissance de notre GPU pour les calculs tensoriels , ce qui permet d'accélérer énormément la vitesse des calculs.

## 3.2 SUSY dataset

Le dataset utilisé contient 5 000 000 mesures produites par méthode Monte Carlo , 19 features pour chaque mesure ,la première est un booléen signal/bruit les 8 suivantes sont les mesures brutes , enfin les 10 dernières sont des fonctions calculées des 8 premières connues pour être utile pour repérer la supersymétrie. Respectivement les data sont lepton 1 pT, lepton 1 eta, lepton 1 phi, lepton 2 pT, lepton 2 eta, lepton 2 phi, missing energy magnitude, missing energy phi, MET-rel, axial MET, M-R, M-TR-2, R, MT2, S-R, M-Delta-R, dPhi-r-b, cos(theta-r1)

## 3.3 Réseau de neurones

Le réseau de neurones initialement généré utilise 18 , 10 ou 8 features pour la première couche selon si l'on souhaite utiliser les données low-level ou high-level , et deux couches "cachées" de n=200 et n=100 neurones , puis une couche output n=2 .D'autres layout du réseau seront utilisés plus tard pour déterminer le plus optimal pour ce problème.

## 3.4 Normalisation , dropout

Afin d'éviter l'overfitting et autres problèmes liés à une interpretation inégale des données , nous utilisons le dropout et la normalisation de batch intégrés dans Pytorch , le dropout permet à l'algorithme de "sauter" aléatoirement des neurones pendant l'entraînement , la normalisation

de batch permet de rajouter une couche qui régularise chaque mini batch en fonction de sa variance et moyenne.

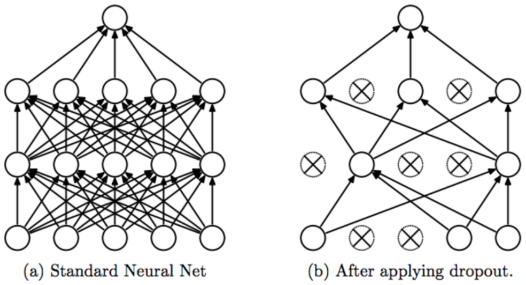



FIGURE 2 – Algorithme dropout

## 3.5 activation , loss , optimizer

### 3.5.1 activation

Les fonctions d'activation sont des fonctions de calcul pour le calcul et l'interaction des neurones. Ce sont des fonctions qui engagent chaque neuron dans l'apprentissage actif des modèles entre les datas input et ses datas target correspondantes. Nous avons sigmoïde, unités linéaires rectifiées (relu), unité linéaire exponentielle (elu), tanh etc.

Différents choix de non-linéarités conduisent à différents calculs et formations propriétés des neurones. nous avons besoin d'un peu de logique / heuristique pour savoir quelle fonction d'activation doit être utilisée dans quelle situation.

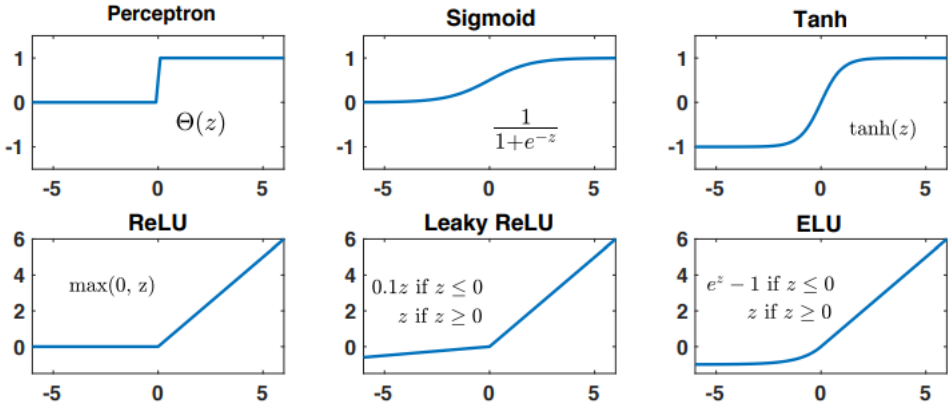



FIGURE 3 – fonctions d'activiation

### 3.5.2 loss

En machine learning les fonctions de loss sont des algorithmes mathématiques permettent de quantifier la différence entre l'output de l'algorithme et les résultats réels.

Il existe différentes fonctions loss pour différents problèmes. Les problèmes ML peuvent être soit des problèmes de classification, soit des problèmes de régression. Dans notre projet, nous concentrons sur un problème de classification. Nous utilisons les fonctions loss basées sur la classification : binary cross entropy, categorical cross entropy, cosine similarity etc.

La variable de sortie dans le problème de classification est généralement une valeur de probabilité f(x), appelée score pour l'entrée x. En général, l'ampleur du score représente la confiance de notre prédiction. La variable cible y est une variable binaire, 1 pour vrai et -1 pour faux.

### 3.5.3 optimizer

Un optimizer est essentiellement un algorithme qui aide un autre algorithme à atteindre ses performances maximales de la manière la plus rapide. En ce qui concerne le ML (réseau de neurones), nous pouvons dire qu'un optimizer est un algorithme mathématique qui aide notre fonction loss à atteindre son point de convergence avec un délai minimum (et surtout, à réduire la possibilité d'explosion de gradient). Les exemples incluent, adam, descente de gradient stochastique (SGD), adadelta, rmsprop, adamax, adagrad, nadam etc.

## 3.6 Matériel Utilisé

Le code à été fait avec, python 3.8 et est executé sur un ordinateur sous windows 10, équipé d'un CPU AMD ryzen 5600X 6c/12t et d'un GPU Nvidia RTX 3080 , cuda version 11.0.

# 4 Résultats

## 4.1 Performance initiale

Le point de départ de nos résultats est la performance obtenue avec un réseau de neurones avec deux couches cachées de dimension n=200, en faisant une grid search sur les learning rates et la dataset size nous arrivons à obtenir une précision plutôt mauvaise, jusqu'à 62-65% avec une learning rate à 0.1 et une dataset size assez élevée (200 000).

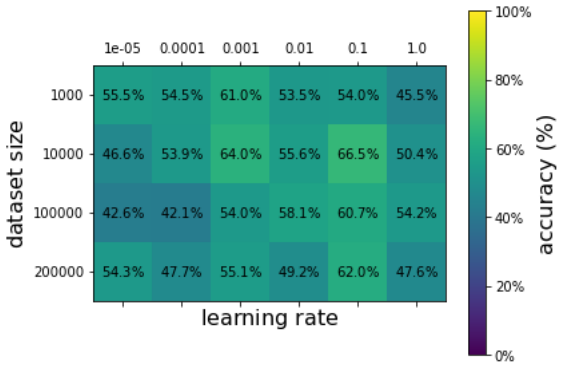
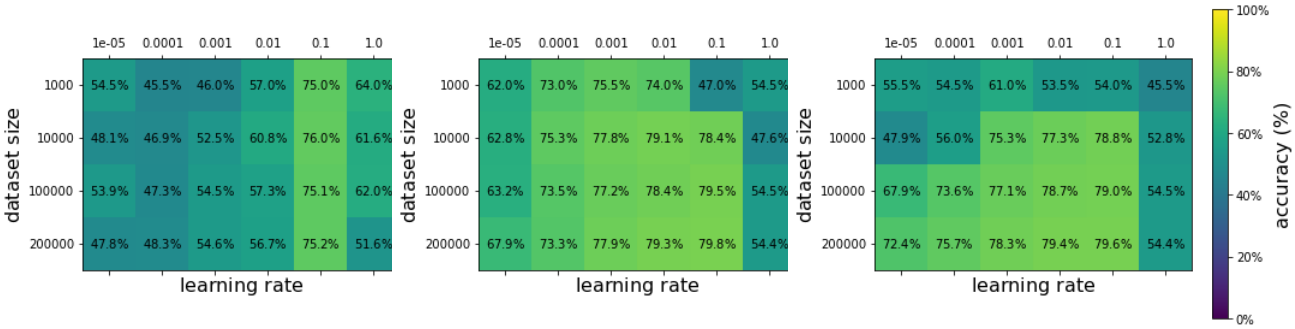



FIGURE 4 – Performance initiale

## 4.2 Mini batch et epoch

Nous utilisons le dataloader de pytorch (que nous avons du modifier à cause de problèmes de performances multiprocessing windows) pour pouvoir générer des mini-batch à partir de la base de données et pouvoir boucler sur ces-derniers à chaque epoch afin de mieux contrôler la variation des poids de l'optimizer. Nous avons comparé la performance sans batch et avec , ici de taille fixée à 1000 , puis , a 10% du datasize .Nous avons aussi augmenté le nombre d'epoch sur lequel le réseau est entraîné des 10 initales à 20. Ces changements ont apporté une très grande amélioration du modèle , sa précision atteignant maintenant plus de 79% de précision dans certains cas.




(a) No batch , 20 epoch (b) Batch scale 10% , 20 epoch (c) Batch size 1000 , 20 epoch

## 4.3 Accélération GPU et vitesse de calcul

Le code utilisé à l'origine utilise uniquement le calcul sur CPU , cependant il est possible avec pytorch d'utiliser la puissance de calcul de notre GPU pour accélérer énormément l'entraînement du réseau de neurones ,l'utilisation du multiprocessing ne fonctionne pas de manière optimale sous windows il à fallu trouver un fix pour le dataloader natif de pytorch pour pouvoir executer le code avec plusieurs workers sans qu'il soit enormément ralenti. Il est souvent arrivé que le code se "soft-lock" lorsque l'on utilise un nombre de workers supérieur à 0 avec windows , il résulte malgré tout de cette implémentation une amélioration considérable de la vitesse d'entraînement du réseau. Nous avons utilisé un grid search avec les paramètres initiaux pour benchmark cette différence. Il ne s'agit pas du best case scenario car le CPU etait toujours la limite dans ce cas avec le GPU à 10-30% d'utilisation des coeurs CUDA mais cela nous à permis de nous donner une base pour vérifier que tout fonctionnait bien pour la suite. Le temps d'execution du gridsearch précédent est passé de 931.9 secondes avec uniquement le CPU , à 546.0 secondes avec GPU et 0 workers , puis enfin 211.5 secondes avec GPU et 4 workers (maximum stable pour ce cas). On attend une amélioration encore plus significative dans les scénarios futurs , avec des batch size plus petits ou réseaux plus profonds.

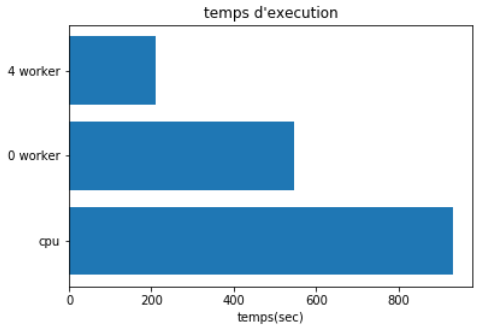
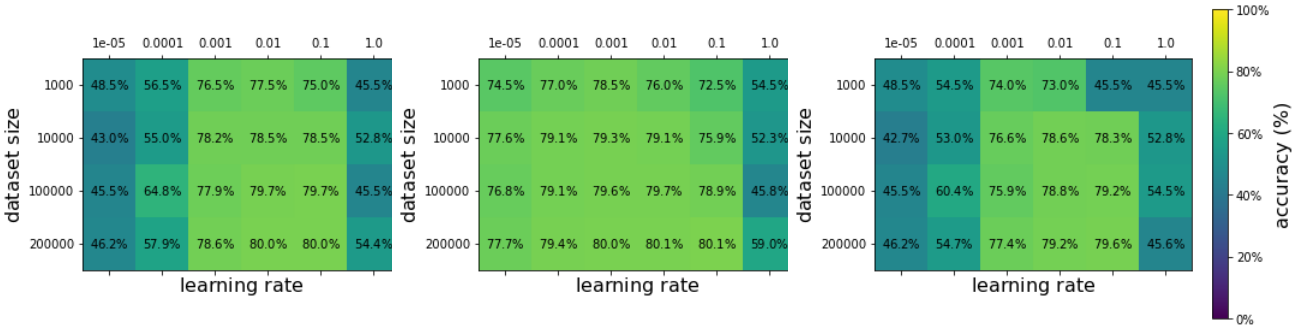



FIGURE 6 – Acceleration GPU

## 4.4 Normalisation des batch et dropout

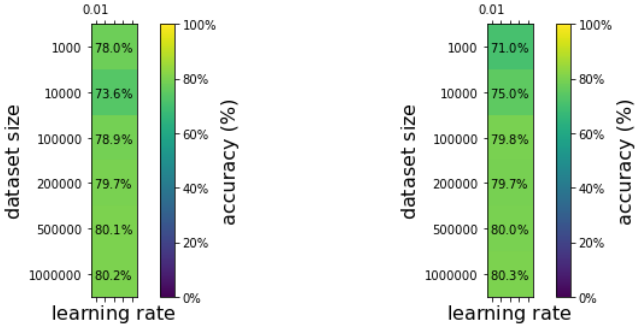
On ajoute ensuite la normalisation des batch pour en mesurer l'impact sur l'efficacité d'apprentissage , on remarque une performance similaire ou supérieure dans la plupart des cas , impact positif mais très léger avec une learning rate optimale. Pour l'algorithme de dropout , les résultats sont mitigés et il semble avoir un impact légèrement négatif sur la performance dans notre cas. Dans la suite on se concentrera sur des learning rate entre $10^{-3}$ et 1 car ce sont celles pour lesquelles l'on observe les meilleures performances ,ce qui nous permettra notamment d'entraîner sur plus d'epoch et plus de data sans perdre trop de temps de calcul inutilement.




(a) No normalisation No dropout (b) Normalized (c) With dropout

## 4.5 Optimizer et fonction loss

L'optimizer de base utilisé est le SGD , standard gradian descent , et la fonction loss est la norme L1 , on utilise aussi la cross entropy , pas de différence mesurable entre les deux sur notre modèle. Les optimizer utilisés ensuite sont Adam et Adadelta , basés sur une autre méthode qui calcule des learning rates adaptatives pour chaque paramètre , en prenant compte le momentum. Ils fonctionnent comme une balle lourde avec de la friction et de l'inertie, il cherche donc un minimum plat. En général il semble équivalent ou meilleur que le SGD. Nos calculs montrent une différence marginale de précision.



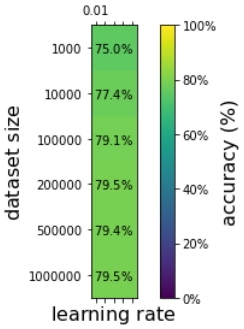

(a) SGD (b) Adadelta

## 4.6 Dimensions du réseau

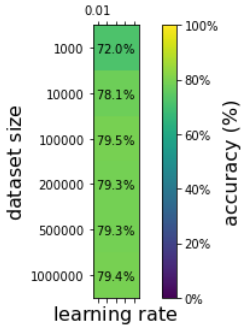
Le réseau utilise jusque là deux couches cachées de dimensions n=200, on essaie différents layout de neurones , après plusieurs essais , il semble très difficile de trouver un layout qui améliore la performance d'apprentissage , la plupart des tentatives semblent donner entre 79.9% et 80.1% de précision. Cependant ,avec certains layout , l'on arrive à avoir jusqu'à 80.5% de précision , ce qui constitue une amélioration faible.

## 4.7 Nombre de features

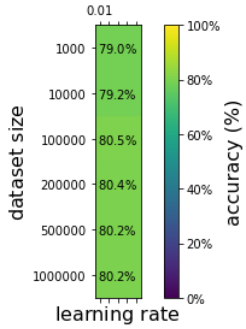
Le code permet d'utiliser au choix 8 ,10 ou 18 features , on fait donc des essais avec seulement les low-level , seulement les high-level pour les comparer aux résultats en utilisant toutes les mesures. On remarque dans les deux cas une baisse de performance , il vaut donc mieux utiliser toutes les features.




(a) Low level only




(b) High level only




(c) All features

# 5 Conclusion

Dans ce projet , nous avons choisi d'aborder le problème de classification SUSY benchmark à l'aide d'un réseau de neurones que nous avons peu à peu amélioré et optimisé pour discerner le signal du bruit , le réseau de neurones tel que nous l'avons construit a semblé atteindre assez vite un plateau de précision aux alentours de 80% , malgré les très nombreux tweaking de paramètres et les longues simulations , la vaste majorité des modifications semblait n'impacter que très marginalement ou ne pas impacter la performance finale qui reste néanmoins correcte. Cependant, dans le papier[1] de Baldi, Sadowski et Whiteson , les chercheurs ont réussi à obtenir jusqu'à 87-89% de précision avec une architecture de réseau et de training plus raffinée, nous espérions nous approcher un peu plus de ce résultat mais nous n'avons pas réussi à améliorer plus loin notre modèle. Une telle différence de performance est importante pour la recherche de nouvelles particules car cela signifie que le taux d'erreur est diminué de moitié sur un phénomène déjà difficile à observer.

# Références

[1] P. Baldi, P. Sadowski & D. Whiteson *Searching for exotic particles in high-energy physics with deep learning* Nature COMMUNICATION.2014
`https://www.nature.com/articles/ncomms5308`

[2] Supersymmetry
`https://en.wikipedia.org/wiki/Supersymmetry`

[3] Boson de Higgs
`https://fr.wikipedia.org/wiki/Boson_de_Higgs`