

RIEN NE SERT DE COURIR, IL FAUT PARTIR À POINT
RECENT ADVANCES IN MACHINE LEARNING

Théo Lopès-Quintas

BPCE Payment Services,
Université Paris Dauphine

30 janvier 2024

INTRODUCTION

FORMULATION D'UN PROBLÈME DE MACHINE LEARNING

Dans le cadre supervisé, nous avons accès à un dataset \mathcal{D} défini comme :

$$\mathcal{D} = \left\{ (x_i, y_i) \mid \forall i \leq n, x_i \in \mathbb{R}^{d'}, y_i \in \mathcal{Y} \right\}$$

Nombre d'observations Nombre d'informations

Avec $\mathcal{Y} \subseteq \mathbb{R}$ pour un problème de régression et $\mathcal{Y} \subset \mathbb{N}$ dans le cadre d'une classification. Les problèmes de Machine Learning supervisé peuvent souvent s'écrire sous la forme d'une optimisation d'une fonction de perte $\mathcal{L} : \mathbb{R}^d \times \mathcal{M}_{n,d'} \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ comme :

Vecteur des paramètres optimaux

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta, X, y)$$

Dimension du vecteur de paramètres

Dans la suite, pour simplifier les notations, nous omettrons la dépendance de \mathcal{L} en X (matrice des informations) et y (vecteur réponse). Notons qu'en général, nous avons $d \neq d'$ et dans le cas du deep learning, très souvent $d \gg d'$.

PARTIR DU BON PIED

- 1 Partir du bon pied 2**
 - 1.1 Apprentissage dans un réseau de neurone 3
 - 1.2 Une meilleure initialisation 6
- 2 Un mauvais départ n'est pas rattrapé 12
- 3 Bien se lancer pour mieux généraliser 21

PARTIR DU BON PIED

NEURONE

Un neurone renvoie un nombre après l'application d'une fonction f que l'on appelle fonction **d'activation** : nous reviendrons après sur cette notion.

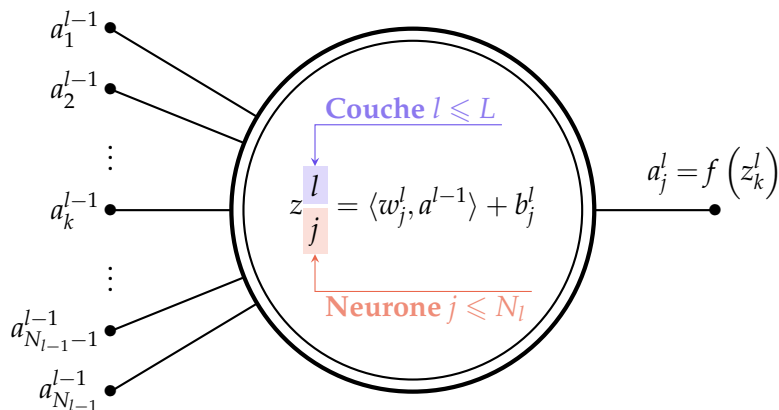


Figure – Neurone j de la couche l paramétré par les poids $w_i^j \in \mathbb{R}^{N_{l-1}}$ et le biais $b_j^l \in \mathbb{R}$

Une question centrale est de comprendre comment on peut *apprendre* ces paramètres.

PARTIR DU BON PIED

COMMENT CALCULER LES DÉRIVÉES PARTIELLES ?

Exercice 1 (Back propagation)

On reprend les notations définies précédemment.

1. Soit $l \leq L$, $j \leq N_l$ et $k \leq N_{l-1}$. Montrer que :

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \quad \text{et} \quad \frac{\partial C}{\partial w_{j,k}^l} = a_k^{l-1} \frac{\partial C}{\partial z_j^l}$$

2. On note $\delta_j^l = \frac{\partial C}{\partial z_j^l}$ qui représente l'erreur du neurone $j \leq N_l$ à la couche $l \leq L$. Montrer que pour la dernière couche L , on a :

$$\forall j \leq N_L, \quad \delta_j^L = f'(z_j^L) \frac{\partial C}{\partial a_j^L}$$

3. On considère à présent la couche $l < L$. Montrer que l'on peut exprimer δ_j^l comme :

$$\delta_j^l = \sum_{k=1}^{n_l} w_{j,k}^{l+1} \delta_k^{l+1} f'(z_j^l)$$

PARTIR DU BON PIED

EXPLOSION ET DISPARITION DES GRADIENTS

Pour le neurone $j \leq N_l$ à la couche $l \leq L$ on a obtenu que $\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l$ et $\frac{\partial \mathcal{L}}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l$ avec

$$\delta_j^l = \sum_{k=1}^{N_l} w_{k,j}^{l+1} \delta_k^{l+1} f'(z_j^l).$$

Si l'on note $\gamma = \max_{x \in \mathbb{R}} f'(x)$ alors :

$$|\delta_j^l| \leq \gamma^{L-l+1} \left| \sum_{k=1}^{n_l} w_{k,j}^{l+1} \times \left(\sum_{k=1}^{n_{l+1}} w_{k,j}^{l+2} \times \left(\dots \left(\sum_{k=1}^{n_{L-1}} w_{k,j}^L \delta_j^L \right) \dots \right) \right) \right|$$

Dans les premiers réseaux de neurones la fonction sigmoid était utilisée et dans ce cas $\gamma = \frac{1}{4}$ ce qui pouvait conduire à une disparition de l'information des gradients dans les réseaux profonds. Ce phénomène est appelé *vanishing gradient*. Le phénomène inverse¹ s'appelle *exploding gradient*.

1. Avoir des gradients qui sont de plus en plus grand quand on remonte le réseau

PARTIR DU BON PIED

UNE MEILLEURE INITIALISATION

On considère un réseau de neurones à $L \in \mathbb{N}^*$ couches. On reprend la majorité des notations précédentes, mais on note z_k^l le résultat du neurone k à la couche l défini par :

$$\forall l \leq L, \forall k \in N_l, z_k^l = \langle w_k^l, x^l \rangle$$

Diagram illustrating the notation for the output of a neuron k at layer l :

- Nombre de couche du réseau** (Number of layers of the network) points to L .
- Vecteur d'entrée de la couche l** (Input vector of layer l) points to x^l .
- Nombre de neurones de la couche l** (Number of neurons of layer l) points to N_l .
- Matrice des poids du neurone k à la couche l** (Weight matrix of neuron k at layer l) points to w_k^l .

Nous faisons les hypothèses supplémentaires :

- **Indépendance et distribution** : $(w^l)_{l \leq L}$, $(x^l)_{l \leq L}$ et $(z^l)_{l \leq L}$ sont indépendants et identiquement distribués (respectivement)
- **Indépendance** : $\forall l \leq L, w^l \perp\!\!\!\perp x^l$
- **Poids centrés** : $\mathbb{E}[w^l] = 0$ et la distribution est symétrique autour de 0
- **Résultats centrés** : $\mathbb{E}[z^l] = 0$ et la distribution est symétrique autour de 0

On souhaite obtenir une bonne propagation des signaux dans les deux sens afin d'éviter une explosion ou une disparition des gradients. Formellement :

$$\forall l \leq L, \mathbb{V}[z^l] = \mathbb{V}[z^1] \quad (1)$$

PARTIR DU BON PIED

UNE MEILLEURE INITIALISATION

Calculons pour le neurone $j \leq N_l$ à la couche $l \leq L$:

$$\begin{aligned}\mathbb{V} \left[z_k^l \right] &= \mathbb{V} \left[\langle w_k^l, x^l \rangle \right] \\ &= \sum_{j=1}^{n_{\text{input}}} \mathbb{V} \left[w_{k,j}^l x_j^l \right] \text{ par indépendance} \\ &= n_{\text{input}} \mathbb{V} \left[w_{k,j}^l x_j^l \right] \text{ car identiquement distribué, pour } j \leq n_{\text{input}} \\ &= n_{\text{input}} \left(\mathbb{V} \left[w_{k,j}^l \right] \mathbb{V} \left[x_j^l \right] + \mathbb{E} \left[w_{k,j}^l \right]^2 \mathbb{V} \left[x_j^l \right] + \mathbb{E} \left[x_j^l \right]^2 \mathbb{V} \left[w_{k,j}^l \right] \right) \\ &= n_{\text{input}} \mathbb{V} \left[w_{k,j}^l \right] \left(\mathbb{V} \left[x_j^l \right] + \mathbb{E} \left[x_j^l \right]^2 \right) \text{ car } \mathbb{E} \left[w_{k,j}^l \right] = 0 \\ &= n_{\text{input}} \mathbb{V} \left[w_{k,j}^l \right] \mathbb{E} \left[\left(x_j^l \right)^2 \right]\end{aligned}$$

Rappelons que $x^l = f \left(z^{l-1} \right)$ avec f la fonction d'activation entre la couche $l - 1$ et la couche l .

PARTIR DU BON PIED

UNE MEILLEURE INITIALISATION

Nous venons d'obtenir que :

$$\mathbb{V} \left[z_k^l \right] = n_{\text{input}} \mathbb{V} \left[w_{k,j}^l \right] \mathbb{E} \left[\left(x_j^l \right)^2 \right]$$

Exercice 2 (Cas d'une fonction impaire)

On suppose ici que f est impaire.

1. Montrer que $\mathbb{E} \left[\left(x_j^l \right)^2 \right] = \mathbb{V} \left[z_k^{l-1} \right]$
2. En déduire une condition suffisante pour vérifier l'équation souhaitée :

$$\forall l \leq L, \mathbb{V} \left[z^l \right] = \mathbb{V} \left[z^1 \right]$$

La condition précédente a été obtenue en considérant une passe forward avec une fonction d'activation impaire. On peut se convaincre que la passe backward abouti à la même forme d'équation sauf que cette fois n_{input} représente le nombre de neurones de la couche inférieure.

PARTIR DU BON PIED

INITIALISATION GLOROT

Nous venons de reproduire le raisonnement de l'article *Understanding the difficulty of training deep feedforward neural networks* [Glorot and Bengio, 2010] écrit par Xavier Glorot et Yoshua Bengio en 2010. Pour résoudre le problème final, choisir la condition qui conviendra pour les deux passes, les auteurs proposent de prendre la moyenne entre les deux conditions.

Notons que nous n'avons supposé aucune loi pour l'initialisation des poids à part des conditions d'indépendance, distribution et que la loi doit être symétrique autour de 0 et d'espérance 0.

Exercice 3 (Forme uniforme et normale)

On considère un réseau de neurones avec une fonction d'activation impaire. Donner l'expression de la variance en respectant la condition précédente, en supposant que les poids suivent une loi normale puis une loi uniforme.

PARTIR DU BON PIED

UNE MEILLEURE INITIALISATION

Nous avons obtenu :

$$\mathbb{V} \left[z_k^l \right] = n_{\text{input}} \mathbb{V} \left[w_{k,j}^l \right] \mathbb{E} \left[\left(x_j^l \right)^2 \right]$$

Exercice 4 (Cas de la fonction ReLU)

On suppose ici que f correspond à la fonction $\text{ReLU}(x) = \max\{0, x\}$.

1. Montrer que $\mathbb{E} \left[\left(x_j^l \right)^2 \right] = \frac{1}{2} \mathbb{V} \left[z_k^{l-1} \right]$
2. En déduire une condition suffisante pour vérifier l'équation souhaitée :

$$\forall l \leqslant L, \quad \mathbb{V} \left[z^l \right] = \mathbb{V} \left[z^1 \right]$$

PARTIR DU BON PIED

INITIALISATION HE

Nous venons de reproduire le raisonnement de l'article *Delving deep into rectifiers : Surpassing human-level performance on ImageNet classification* [He et al., 2015] publié en 2015 par Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. Contrairement à l'initialisation Glorot, les auteurs proposent de choisir entre le *fan in* et le *fan out*.

Notons que nous n'avons supposé aucune loi pour l'initialisation des poids à part des conditions d'indépendance, distribution et que la loi doit être symétrique autour de 0 et d'espérance 0.

Exercice 5 (Forme uniforme et normale)

On considère un réseau de neurones avec une fonction d'activation ReLU. Donner l'expression de la variance en respectant la condition précédente, en supposant que les poids suivent une loi normale puis une loi uniforme.

UN MAUVAIS DÉPART N'EST PAS RATTRAPÉ

- 1 Partir du bon pied 2
- 2 Un mauvais départ n'est pas rattrapé 12
 - 2.1 Expériences 14
 - 2.2 Mesurer la force de connexion 16
 - 2.3 Périodes critiques 18
- 3 Bien se lancer pour mieux généraliser 21

UN MAUVAIS DÉPART N'EST PAS RATTRAPÉ

CRITICAL LEARNING PERIODS IN DEEP NEURAL NETWORKS

Critical periods are time windows of early post-natal development during which sensory deficits can lead to permanent skill impairment

— Alessandro Achille, Matteo Rovere et Stefano Soatto (2017)

Intuitivement, puisque un réseau peut être amené à s'entraîner longtemps, si l'on lui permet de propager correctement les gradients, il semblerait possible que l'algorithme puisse performer *quoi qu'il arrive*. C'est ce qu'explore l'article *Critical learning periods in deep neural networks* [Achille et al., 2017] publié en 2017 par Alessandro Achille, Matteo Rovere et Stefano Soatto. Dans l'ensemble des expériences, c'est une architecture ResNet² qui est exploité.

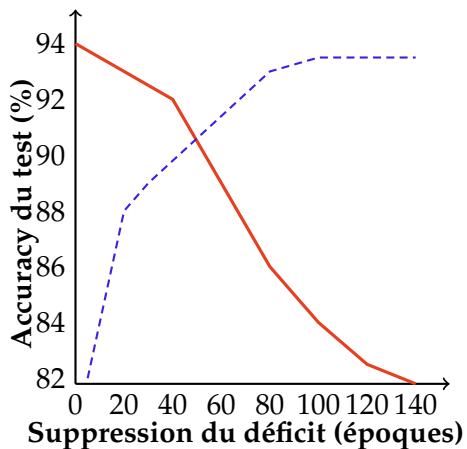
2. Voir annexe

UN MAUVAIS DÉPART N'EST PAS RATTRAPE

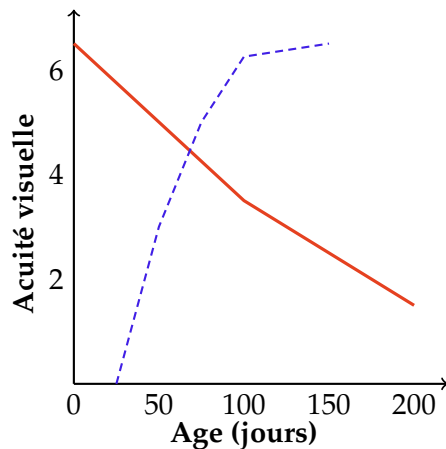
PREMIÈRE EXPÉRIENCE



(a) Description de l'expérience



(b) Réseau de neurones

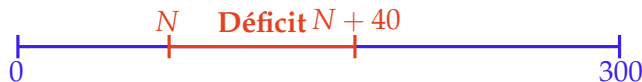


(c) Chaton [Giffin and Mitchell, 1978, Mitchell, 1988]

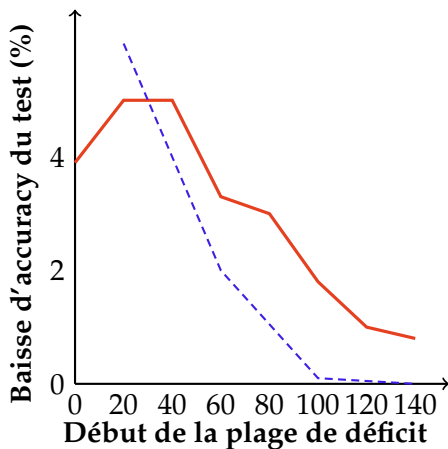
Figure – Performance/acuité selon la durée du déficit par rapport à un développement normal

UN MAUVAIS DÉPART N'EST PAS RATTRAPÉ

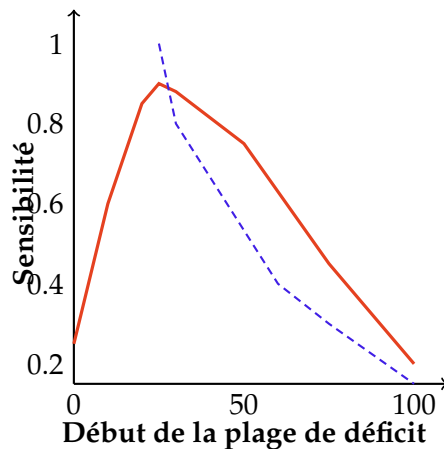
DEUXIÈME EXPÉRIENCE



(a) Description de l'expérience



(b) Réseau de neurones



(c) Chaton [Olson and Freeman, 1980]

Figure – Sensibilité selon le début du déficit par rapport à un développement normal

UN MAUVAIS DÉPART N'EST PAS RATRAPÉ

DIVERGENCE DE KULLBACK-LEIBLER

On représente le résultat d'un réseau de neurones comme $\mathbb{P}_w(y | x)$ où w représente l'ensemble des poids qui paramètre le réseau de neurones. Si l'on note $w' = w + \delta w$ où δ représente une petite perturbation, alors on peut mesurer l'impact des poids modifiés par la perturbation dans le réseau de neurones comme l'écart entre $\mathbb{P}_w(y | x)$ et $\mathbb{P}_{w'}(y | x)$.

Mais comment mesurer l'écart entre deux distributions ? Cette question rejoint la théorie de l'information et plus particulièrement la divergence de Kullback-Leibler :

$$D_{\text{KL}} \left(\underbrace{P}_{\text{Distribution de densité } p} \parallel \underbrace{Q}_{\text{Distribution de densité } q} \right) = \int_{-\infty}^{+\infty} p(x) \ln \left(\frac{p(x)}{q(x)} \right) dx \quad (\text{Kullack-Leibler})$$

Cette divergence mesure ce que l'on peut espérer perdre en remplaçant la **vrai distribution** par une **distribution estimée**.

UN MAUVAIS DÉPART N'EST PAS RATRAPÉ

INFORMATION DE FISHER

Puisque nous nous intéressons ici à des réseaux de neurones avec énormément de paramètre, l'estimation de la divergence de Kullback-Leibler va être difficile en l'état. On peut montrer, mais ce n'est pas l'objet de ce cours, que le développement de Taylor à l'ordre deux de la divergence de Kullback-Leibler s'écrit sous la forme suivante :

$$D_{\text{KL}}(\mathbb{P}_w(y | x) \parallel \mathbb{P}_{w'}(y | x)) = \delta w F(\delta w) + o(\delta w^2)$$

Avec F la métrique d'information de Fisher. On peut la comprendre dans ce contexte comme une métrique **locale** de la perturbation de la valeur d'un poids. Concrètement, si un poids a une information de Fisher faible c'est que le poids n'est pas très important pour la sortie du réseau de neurone. À nouveau la matrice d'information de Fisher est trop grande pour être calculée systématiquement. Ce que l'article propose finalement, est de ne considérer que la force de connexion entre chaque couche : ne considérer que la trace de la matrice d'information de Fisher.

UN MAUVAIS DÉPART N'EST PAS RATRAPÉ

PÉRIODES CRITIQUES

Puisque l'on peut interpréter la trace de la matrice de Fisher comme la quantité d'information *apprise* par le modèle, on devrait voir une augmentation continue au fur et à mesure de l'entraînement. Mais peut-être moins quand il y a un déficit.

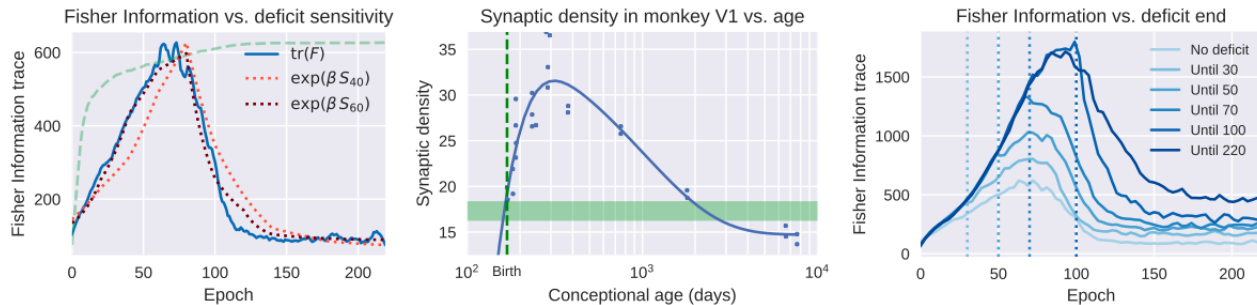


Figure – Evolutions de la trace de la matrice d'information de Fisher (source : [Achille et al., 2017])

UN MAUVAIS DÉPART N'EST PAS RATTRAPÉ

PÉRIODES CRITIQUES

Voyons comment les différentes couches participent à ce phénomène.

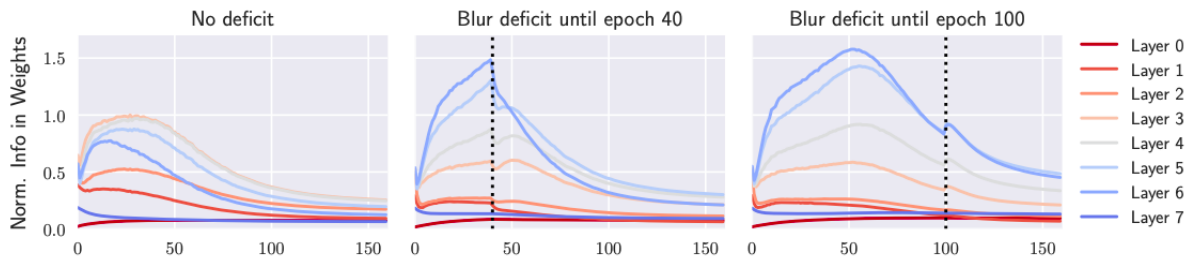


Figure – Evolutions de la trace de la matrice d'information de Fisher par couche (source : [Achille et al., 2017])

UN MAUVAIS DÉPART N'EST PAS RATTRAPÉ

PÉRIODES CRITIQUES

Une manière de s'assurer qu'un réseau ne *mémorise* pas les input est de le régulariser. Observons l'impact du *weight decay*³.

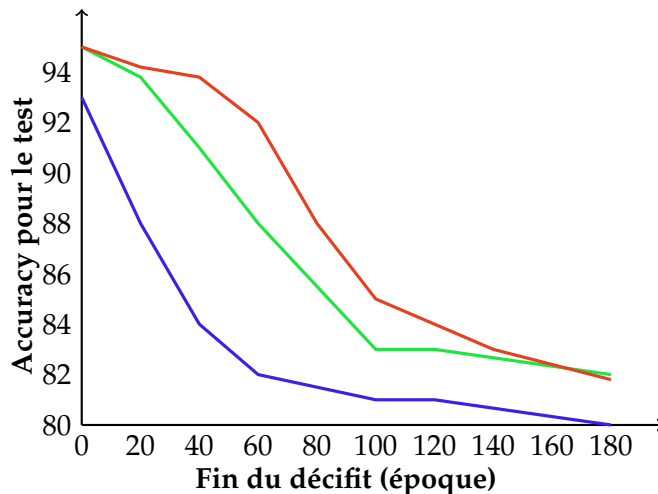


Figure – Performance sur le dataset de test en fonction du nombre d'époque de maintien du déficit pour un weight decay de $\lambda = 0$, $\lambda = 5 \cdot 10^{-4}$ et $\lambda = 10 \cdot 10^{-4}$

3. La notion est présentée plus en détail dans un séance prochaine

BIEN SE LANCER POUR MIEUX GÉNÉRALISER

- 1 Partir du bon pied 2
- 2 Un mauvais départ n'est pas rattrapé 12
- 3 Bien se lancer pour mieux généraliser 21

BIEN SE LANCER POUR MIEUX GÉNÉRALISER

DROPOUT

Le dropout [Hinton et al., 2012] et [Srivastava et al., 2014] est introduit en 2012 par Geoffrey Hinton Nitish Srivastava, Alex krizhevsky, Ilya Sutskever et Ruslan Salakhutnov. Cette méthode de régularisation est rapidement devenu une brique essentielle des réseaux de neurones. Un des facteurs du succès immédiat de la méthode est son utilisation dans le modèle AlexNet [Krizhevsky et al., 2012], vainqueur de la compétition ImageNet en 2012.

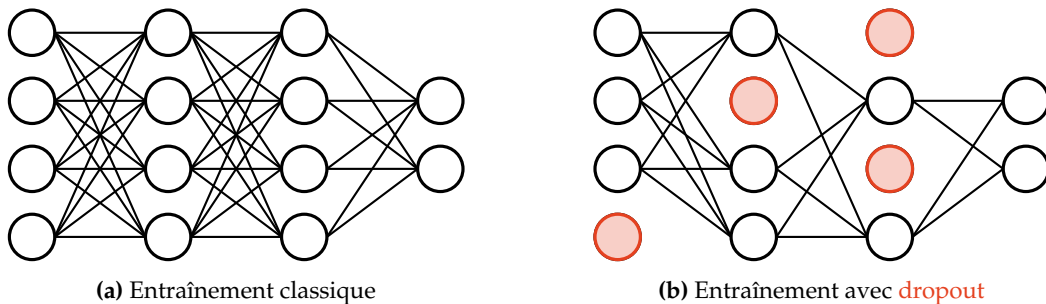


Figure – Réseau de neurones avec et sans **dropout**

BIEN SE LANCER POUR MIEUX GÉNÉRALISER

EARLY DROPOUT

[Liu et al., 2023] propose une étude approfondie de cette mécanique en comparant un modèle entraîné avec dropout de son équivalent sans dropout. Nous retenons trois informations :

- ▶ **Magnitude** : Un modèle avec du dropout produira des gradients de plus faible magnitude que son équivalent sans dropout
- ▶ **Déplacement** : Cependant, un modèle avec du dropout tend à avoir un déplacement plus grand dans l'espace des paramètres
- ▶ **Variance de direction** : Un modèle avec un dropout présente une variance de direction des gradients plus faible batch par batch

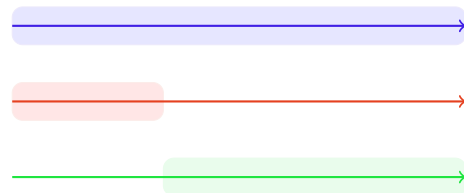


Figure – Dropout, early dropout et late dropout

L'article propose cependant de ne pas utiliser le dropout tout au long de l'entraînement et propose l'**early dropout** et le **late dropout** qui sont censé résoudre respectivement l'underfitting et l'overfitting spécifiquement.

BIEN SE LANCER POUR MIEUX GÉNÉRALISER

STOCHASTIC DEPTH

[Liu et al., 2023] conclut en l'utilité de l'early dropout pour réduire l'underfitting, et que pour réduire l'overfitting dans un ResNet il vaut mieux utiliser la variante *stochastic depth* [Huang et al., 2016] :

We propose deep networks with stochastic depth, a novel training algorithm that is based on the seemingly contradictory insight that ideally we would like to have a deep network during testing but a short network during training. We resolve this conflict by creating deep Residual Network (with hundreds or even thousands of layers) with sufficient modeling capacity; however, during training we shorten the network significantly by randomly removing a substantial fraction of layers independently for each sample or mini-batch.

— Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Weingerger (2016)






L'idée est de supprimer aléatoirement des ResBlocks entiers pendant l'entraînement !

Exercice 6 (Expected network depth)






Il est proposé que chaque ResBlock ait une probabilité p_l d'être présent. On fixe $p_0 = 1$ et p_L pour la dernière couche. Pour le reste, on décide que $p_l = 1 - \frac{l}{L}(1 - p_L)$.

Quelle est l'espérance de la profondeur d'un réseau de neurone à l'entraînement ?






BIBLIOGRAPHIE I

-  [Achille, A., Rovere, M., and Soatto, S. \(2017\).](#)
Critical learning periods in deep neural networks.
arXiv preprint arXiv :1711.08856.
-  [Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., and McWilliams, B. \(2017\).](#)
The shattered gradients problem : If resnets are the answer, then what is the question?
In International Conference on Machine Learning. PMLR.
-  [Giffin, F. and Mitchell, D. E. \(1978\).](#)
The rate of recovery of vision after early monocular deprivation in kittens.
The Journal of Physiology.
-  [Glorot, X. and Bengio, Y. \(2010\).](#)
Understanding the difficulty of training deep feedforward neural networks.
In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256.
JMLR Workshop and Conference Proceedings.
-  [He, K., Zhang, X., Ren, S., and Sun, J. \(2015\).](#)
Delving deep into rectifiers : Surpassing human-level performance on imagenet classification.
In Proceedings of the IEEE international conference on computer vision, pages 1026–1034.

BIBLIOGRAPHIE II

-  [He, K., Zhang, X., Ren, S., and Sun, J. \(2016\).](#)
Deep residual learning for image recognition.
In Proceedings of the IEEE conference on computer vision and pattern recognition.
-  [Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. \(2012\).](#)
Improving neural networks by preventing co-adaptation of feature detectors.
arXiv preprint arXiv :1207.0580.
-  [Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. \(2016\).](#)
Deep networks with stochastic depth.
In Computer Vision—ECCV 2016 : 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14.
-  [Krizhevsky, A., Sutskever, I., and Hinton, G. E. \(2012\).](#)
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems.
-  [Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. \(2018\).](#)
Visualizing the loss landscape of neural nets.
Advances in neural information processing systems, 31.

BIBLIOGRAPHIE III

-  [Liu, Z., Xu, Z., Jin, J., Shen, Z., and Darrell, T. \(2023\).](#)
Dropout reduces underfitting.
arXiv preprint arXiv :2303.01500.
-  [Mitchell, D. E. \(1988\).](#)
The extent of visual recovery from early monocular or binocular visual deprivation in kittens.
The Journal of physiology.
-  [Olson, C. R. and Freeman, R. \(1980\).](#)
Profile of the sensitive period for monocular deprivation in kittens.
Experimental Brain Research.
-  [Simonyan, K. and Zisserman, A. \(2014\).](#)
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv :1409.1556.
-  [Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. \(2014\).](#)
Dropout : a simple way to prevent neural networks from overfitting.
The journal of machine learning research.

BIBLIOGRAPHIE IV

 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015).

Going deeper with convolutions.

In Proceedings of the IEEE conference on computer vision and pattern recognition.

 Zhang, H., Dauphin, Y. N., and Ma, T. (2019).

Fixup initialization : Residual learning without normalization.

ICLR.

ANNEXE : LEAKY RELU

INITIALISATION DES POIDS

ReLU souffre du phénomène de mort des neurones. Puisque *la moitié du temps* ses gradients sont nuls, les neurones n'apprennent plus. Leaky ReLU a été proposé pour contrer cet effet en ajoutant une pente quand x est négatif. On considère un réseau de neurones avec Leaky ReLU de paramètre $\alpha \in [0, 1]$. En reprenant les notations, on a :

$$\begin{aligned}\mathbb{E}[x^2] &= \mathbb{E}[f(z)^2] \\ &= \int_{-\infty}^{+\infty} \max\{\alpha t, t\}^2 p(t) dt \\ &= \int_{-\infty}^0 \alpha^2 t^2 p(t) dt + \int_0^{+\infty} t^2 p(t) dt \\ &= \frac{\alpha^2 + 1}{2} \mathbb{V}[z]\end{aligned}$$

- Forme normale : $\mathcal{N}(0, \sigma^2) : \sigma^2 = \frac{2}{(1 + \alpha^2) \times \text{fan mode}}$
- Forme uniforme : $\mathcal{U}([-a, a]) : a^2 = \frac{6}{(1 + \alpha^2) \times \text{fan mode}}$

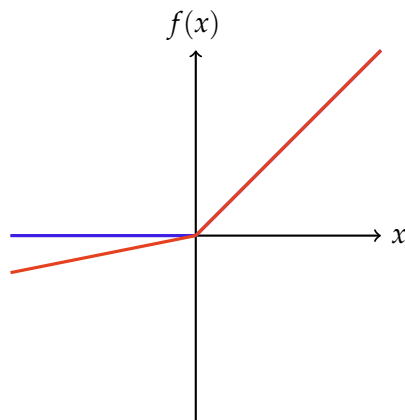


Figure – Fonctions **ReLU** et **Leaky ReLU**

ANNEXE : RESNET

UNE ARCHITECTURE PLUS LISSE

L'architecture ResNet remporte la prestigieuse compétition ImageNet en 2015 [He et al., 2016], un an après l'architecture Inceptionv1 [Szegedy et al., 2015] et VGG [Simonyan and Zisserman, 2014]. Les ResNets sont des réseaux très profonds qui utilisent des *ResBlock* :

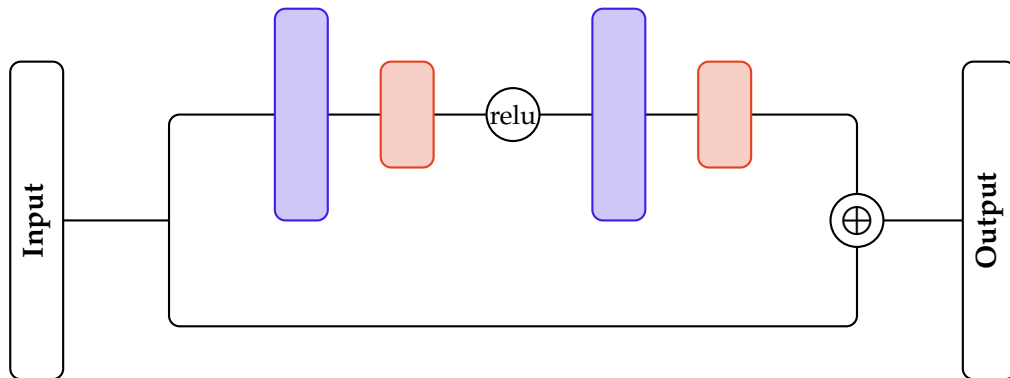


Figure – ResBlock dans un ResNet avec des couches de **convolutions** et de **Batch Normalisation**

L'input passe deux couches de convolution et le résultat s'ajoute à l'input pour produire l'output. De cette manière on peut apprendre des features plus intéressantes plus profondément sans pour autant tomber dans les difficultés d'effondrement des réseaux de neurones profonds. Ainsi, le ResNet50 utilise 50 couches !

ANNEXE : RESNET

UNE ARCHITECTURE PLUS LISSE

De nombreux travaux théoriques ont été mené pour expliquer les performances de ce type d'architecture. La première réponse est lié à la profondeur du réseau qui permet une puissance d'expression forte, d'où la proposition de la mécanique de stochastic depth [Huang et al., 2016]. Nous pouvons citer trois résultats fort sur les ResNet :

- ▶ **Gradients** : les ResNet ont des gradients plus lisses que les autres architectures [Balduzzi et al., 2017]
- ▶ **Surface** : la surface de perte est plus lisses que pour les autres architectures [Li et al., 2018]
- ▶ **Initialisation** : la particularité des ResNet, les *skip connections*, peuvent être remplacé par une bonne initialisation et tout de même obtenir des résultats [Zhang et al., 2019]

Le dernier résultat montre encore combien les premiers pas d'un réseau de neurone sont cruciaux.