Studies of scientific articles
Rethinking Bias-Variance Trade-off for Generalization of Neural Networks

Panongbene Jean Mohamed Sawadogo, Damak Salma, Ait Ettajer Haytham

Abstract

Le compromis biais-variance montre que le biais diminue et que la variance augmente en fonction de la complexité du modèle, conduisant à des courbes de risque en forme de U. Cependant, des travaux récents réaliser sur les réseaux de neurones et sur les modèles sur-paramétrés remettent en question le compromis biais-variance. Ainsi, l'article scientifique **Rethinking Bias-Variance Trade-off for Generalization of Neural Networks** (Yang et al., 2020) fournit une explication de ces résultats sur le compromis biais-variance sur les réseaux de neurones et sur les modèles sur-paramétrés en mesurant le biais et la variance des réseaux de neurones pour observer que le biais est monotone décroissant comme dans la théorie classique, mais que la variance est uni-modale oú en forme de cloche : elle augmente puis diminue avec la largeur du réseau. Pour cela, ils font varier les paramètres des réseaux(fonction de perte, ensemble de données ...) pour confirmer que l'uni-modalité de la variance se produit de manière robuste pour tous les modèles considérés. L'objectif du travail réalisé dans ce rapport est de donner une explication détaillée de l'article scientifique **Rethinking Bias-Variance Trade-off for Generalization of Neural Networks** (Yang et al., 2020) et essayer de reproduire des expériences qui vont confirmer ou infirmer les affirmations faites dans cet article.

1. Biais Variance décomposition

Considérons une distribution de probabilité $\mathcal{P} \in \mathbb{R}^d * \mathbb{R}^c$ définie par une fonction inconnue $f_{\mathcal{P}} : \mathbb{R}^d \to \mathbb{R}^c$ qui à tout $x \in \mathbb{R}^d$ associe $y = f_{\mathcal{P}}(x) \in \mathbb{R}^c$ tel que $(x, y) \in \mathcal{P}$. Considérons l'échantillon indépendamment identiquement distribué $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ suivant la distribution $\mathcal{P}$.

L'objectif est de construire une fonction $f_{\mathcal{T}} : \mathbb{R}^d \to \mathbb{R}^c$ à partir de l'échantillon $\mathcal{T}$ tel que pour tout $(x, y) \in \mathcal{P}$, $\mathbb{E}_{x,y}[\|y - f_{\mathcal{T}}(x)\|_2^2]$ soit le plus minimale possible : c'est à dire que la fonction $f_{\mathcal{T}}$ s'approche le plus possible de la fonction $f_{\mathcal{P}}$ au sens de l'erreur moyenne quadratique.
La fonction $f_{\mathcal{T}}$ est construit à partir de l'échantillon $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ de manière à minimiser l'espérance $\mathbb{E}_{\mathcal{T}}[\|y - f_{\mathcal{T}}(x)\|_2^2]$ pour tout $(x, y) \in \mathcal{T}$.

1.1. Biais Variance

Soit $\overline{f}(x) = \mathbb{E}_{\mathcal{T}}[f_{\mathcal{T}}(x)]$.

Définition: On définit le biais de la fonction $f_{\mathcal{T}}$ par

$$\textbf{Biais}^2 = \mathbb{E}_{x,y}[\|y - \overline{f}(x)\|_2^2] \tag{1}$$

On définit la variance de la fonction $f_{\mathcal{T}}$ par

$$\textbf{Variance} = \mathbb{E}_x[\mathbb{E}_{\mathcal{T}}[\|f_{\mathcal{T}}(x) - \overline{f}(x)\|_2^2]]. \tag{2}$$

Propriété:

$$\mathbb{E}_{x,y}[\mathbb{E}_{\mathcal{T}}[\|y - f_{\mathcal{T}}(x)\|_2^2]] = \textbf{Biais}^2 \textbf{ + Variance} \tag{3}$$

1.2. Estimation du biais et de la variance

Pour estimer le biais et la variance, on utilisera une méthode qui consiste à estimer l'espérance $\mathbb{E}_{\mathcal{T}}$ en divisant l'échantillon d'apprentissage $\mathcal{T}$ en N sous ensemble $\mathcal{T}_1, ..., \mathcal{T}_N$. Avec cette méthode, on sera soumis à un compromis sur le sous-échantillonnage : en effet,on devra faire le choix entre avoir un grand nombre de sous-échantillons de petite taille(N très grand) ou bien avoir un petit nombre d'échantillons de grande taille.

1.2.1. Erreur quadratique moyenne(MSE)

L'idée consiste à construire un estimateur sans biais de la variance et obtenir un estimateur du biais en soustrayant la variance du risque. Ainsi, la variance non biaisée est construite de la manière suivante :

$$\hat{VAR}_{\mathcal{T}}(x) = \frac{1}{N-1} \sum_{j=1}^{N} \|f_{\mathcal{T}_|}(x) - \sum_{i=1}^{N} \frac{1}{N} f_{\mathcal{T}_i}()\|_2^2 \tag{4}$$

la variance de l'estimateur précédant peut être réduit en utilisant plusieurs sous-échantillons aléatoires et en calcu-

*Equal contribution . **AUTHORERR: Missing \icmlcorrespondingauthor.**

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

lant pour chaque sous-échantillon un estimateur de la variance. On obtient ainsi les estimateurs $V\hat{A}R_1, ..., V\hat{A}R_k$ auxquels on prendra la moyenne pour avoir la valeur de $V\hat{A}R_{\mathcal{T}}$. Cela réduit la variance de l'estimateur de variance puisqu'on a :
$VAR_{\mathcal{T}}(\frac{1}{k}\sum_{i=1}^{k} V\hat{A}R_i) = \frac{\sum_{i,j} COV_{\mathcal{T}}(V\hat{A}R_i, V\hat{A}R_j)}{k^2}$
$\leq VAR_{\mathcal{T}}(V\hat{A}R_1)$

1.2.2. Cross-Entropy Loss(CE)

On peut utiliser la décomposition biais variance de la fonction de perte de l'entropie croisée.

Algorithm1 : Estimating Generalized Variance

Input : Test point x, Training set $\mathcal{T}$
for i = 1 **to** k **do**
  Split the $\mathcal{T}$ into $\mathcal{T}_1^{(i)}, ..., \mathcal{T}_N^{(i)}$.
  for j = 1 **to** N do
    Train the model using $\mathcal{T}_j^{(i)}$;
    Evaluate the model at x; call the result $\pi_j^{(i)}$;
  end for
end for
Compute $\hat{\pi} = exp\left\{\frac{1}{N.k}\sum_{ij} log(\pi_j^{(i)})\right\}$
(using element-wise log and exp; $\hat{\pi}$ estimates $\overline{\pi}$)
Normalize $\hat{\pi}$ to get a probability distribution
Compute the variance $\frac{1}{N.k}\sum_{ij} D_{KL}(\hat{\pi}\|\pi_j^{(i)})$

Considérons la variable aléatoire $\pi_{\mathcal{T}}(x) \in \mathbb{R}^c$ représentant la distribution de probabilité des classes du modèle. Soit $\pi_0(x) \in \mathbb{R}^c$ le codage one-hot représentant le bon label de l'élément x. Alors en omettant les dépendances de π et de π_0 à x et $\mathcal{T}$, on définit la fonction de perte du cross entropy par :

$$H(\pi_0, \pi) = \sum_{l=1}^{c} \pi_0[l] log(\pi[l])$$

Soit $\overline{\pi}$ la moyenne des log-probabilités après normalisation : $\overline{\pi}[l] \propto exp(\mathbb{E}_{\mathcal{T}} log(\pi[l]))$ pour $l = 1, .., c$

Définition : Nous définissons le biais de π par

$$\textbf{Biais}^2 = D_{KL}(\pi_0\|\overline{\pi})$$

Nous définissons la variance de π par

$$\textbf{variance} = \mathbb{E}_{\mathcal{T}}[D_{KL}(\overline{\pi}\|\pi)]$$

Propriété : Nous obtenons la décomposition ci-dessous qui est un cas particulier de la décomposition générale de la divergence de Bregman.

$$\mathbb{E}_{\mathcal{T}}[H(\pi_0, \pi)] = \textbf{biais}^2 \textbf{ + variance} \qquad (5)$$

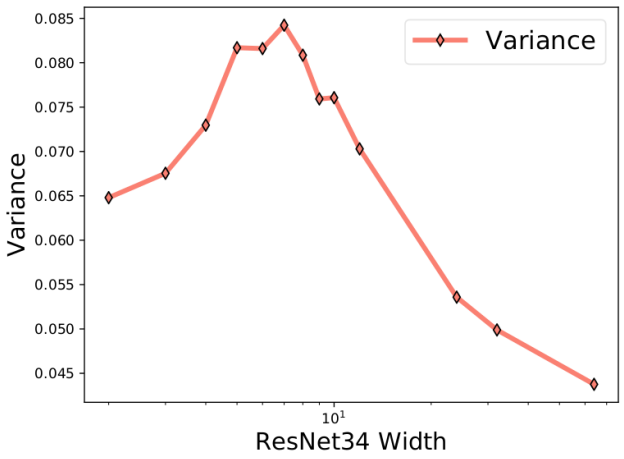
On utilise l'algorithme défini précédemment pour estimer la variance généralisée (5). On n'a pas pu obtenir un estimateur de la variance, mais cependant, l'estimateur devient meilleur si nous faisons plus de sous-échantillons aléatoires : c'est-à-dire (k très grand). En pratique, on choisit le k le plus grand à partir duquel on obtient une stabilisation de la variance si on augmente encore le k...

2. Mesures du Biais et de la Variance des réseaux de neurones

Les expériences effectuées dans cette partie ont pour but de vérifier les propriétés énoncées au début concernant le compromis biais variance sur les réseaux de neurones. En effet comme énoncé au début, dans les réseaux de neurones, le biais décroît de manière monotone en fonction de la complexité du modèle comme dans tous les algorithmes d'apprentissage alors que la variance est uni-modale (elle augmente d'abord jusqu'à atteindre un certain seuil avant de commencer à diminuer). Le biais et la variance considérés dans cette partie sont ceux des équations (1) et (2).

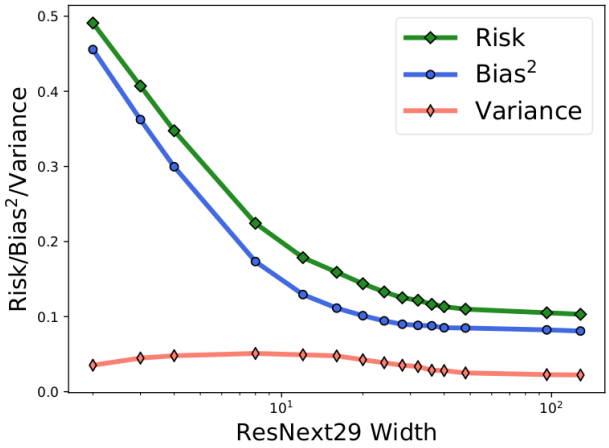
2.1. Paramètres d'apprentissage

Pour les expérimentations, nous avons entraîné un réseau ResNet34(He et al., 2016) sur les données CIFAR10(Krizhevsky et al., 2009). Nous avons utilisé une descente de gradient stochastique (SGD) pour l'apprentissage avec un momentum de 0.9. Le taux d'apprentissage est initialisé à 0.1 et décroît avec un facteur de 10 lors de l'entraînement tous les 200 epochs, on utilise aussi une décroissance de 5.10^{-4}. Pour modifier la complexité du modèle de réseau de neurones, nous modifions le nombre de filtres (c'est-à-dire la largeur) des couches convolutives. C'est-à-dire avec une largeur de ω, le nombre de filtres est de $[\omega, 2\omega, 4\omega, 8\omega]$ On fait varier ω de 2 à 64. Remarquons que dans le réseau initial de ResNet34(He et al., 2016) on a $\omega = 16$.
Pour mesurer la variance et le biais, nous avons besoin de deux modèles formés sur deux ensembles de données indépendantes. Pour ce faire, on divise notre ensemble de données en deux parties : deux ensembles de 25 000 = 50 000/2 éléments. Nous estimons la variance en faisant la moyenne sur N = 3 de ces divisions aléatoires(nous formons aux totale 6 = 3*2 copies de chaque modèle.). Sur la figure ci-dessous on peut observer l'évolution de la variance en fonction de la largeur des filtres qui est uni-modale comme énoncé précédemment.



2.2. 3.2. Architectures, fonctions de perte, jeux de données variables

En faisant varier l'architecture du modèle en utilisant notamment l'architecture ResNext29(Xie et al., 2017) Nous observons le même phénomène de biais et de variance unimodale décroissante.

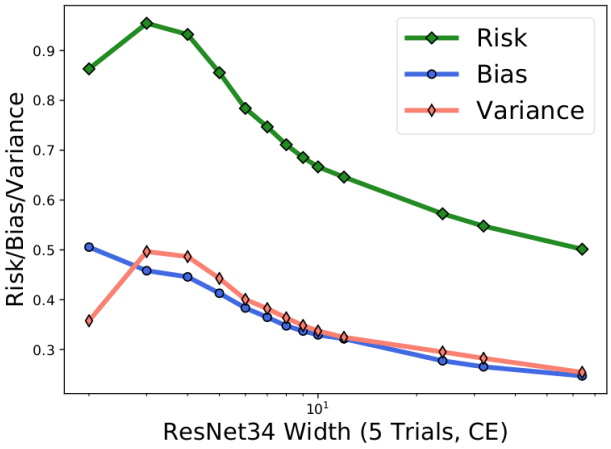


(a) ResNext29, MSE loss, CIFAR10

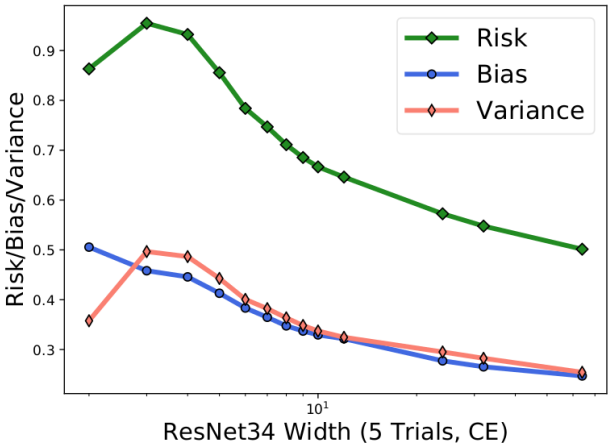
Dans cette figure, on peut observer les résultats obtenus avec le réseau ResNext29 en faisant varier la largeur du réseau.

En utilisant la fonction de perte de cross-entropy avec la décomposition donnée par l'équation (5), nous observons un comportement du biais et de la variance similaire à nos précédentes observations : le biais est décroissant et la variance est uni-modale. En utilisant n = 10 000 échantillons d'apprentissage et en répétant pour N = 4fois avec des divisions aléatoires indépendantes, la figure ci-dessus nous donne la variation du biais et de la variance en fonction de la largeur du réseau.

En changeant notre ensemble de données, par exemple les données MNIST (LeCun, 1998) et les données Fashion-MNIST(Xiao et al., 2017), nos observations restent inchangé : la variance est à nouveau uni-modale et le biais décroît de manière monotone.



(b) ResNet34, CE loss, CIFAR10



(b) ResNet34, CE loss, CIFAR10

On peut observer les résultats du biais et de la variance sur les autres ensembles de données à travers les figures ci-dessus.

3. Origine de la variation du biais et de la variance dans les réseaux de neurones

3.1. Modélisation

Considérons une distribution de probabilité $X \sim \mathcal{N}(0, \mathbf{I}_d/d)$ et $Y = f_0(X) = X^T\theta$ avec $\theta \in \mathbb{R}^d$ inconnu. Notre objectif est de construire une fonction f qui

a tout $(x, y) \sim (X, Y)$ nous donne $y = f(x)$ à partir d'un échantillon d'apprentissage aléatoire $\mathcal{T} = (x_i, y_i)_{i=1}^n$ extrait de la distribution (X, Y).
Pour se faire on construit un réseau de neurones à deux couches de la manière suivante : $\mathbf{W} \in \mathbb{R}^{p*d}$ et $\beta \in \mathbb{R}^p$, on a :

$$f(x) = (\mathbf{W}x)^T\beta$$

Ici, on fait l'hypothèse que le paramètre $\mathbf{W}$ est indépendant de l'échantillon d'apprentissage. et le paramètre β est estimé en utilisant la régression rigide ci-dessous :

$$\beta_\lambda(\mathcal{T}, \mathbf{W}) = \underset{\beta\in\mathbb{R}^p}{argmin}\|(\mathbf{W}X)^T\beta - y\|_2^2 + \lambda\|\beta\|_2^2 \quad (6)$$

oú $\mathbf{X} = [x_1, ..., x_n] \in \mathbb{R}^{d*n}$ est la matrice contenant les données d'entraînement, le vecteur $y = [y_1, ..., y_n] \in \mathbb{R}^n$ est le vecteur des labels et $\lambda \in \mathbb{R}^+$ est le paramètre de régularisation.

En résolvant (6) on obtient :

$$\beta_\lambda(\mathcal{T}, \mathbf{W}) = (WXX^T + \lambda I)^{-1}WXy$$

Ainsi, on obtient un estimateur de f donnée par :

$$f_\lambda(x; \mathcal{T}, \mathbf{W}) = x^T\mathbf{W}^T\beta_\lambda(\mathcal{T}, \mathbf{W}) \quad (7)$$

3.2. Analyse du biais et de la variance

On obtient le biais et la variance du modèle :

$$\textbf{biais}_\lambda(\theta)^2 = \mathbb{E}_x[\mathbb{E}_{\mathcal{T},\mathbf{W}}f_{\lambda(x;\mathcal{T},\mathbf{W})} - f_0(x)]^2$$

$$\textbf{Variance}_\lambda(\theta) = \mathbb{E}_x VAR_{\mathcal{T},\mathbf{W}}[f_{\lambda(x;\mathcal{T},\mathbf{W})}]^2$$

Pour simplifier le modèle nous introduisons le à priori $\theta \sim \mathcal{N}(0, \mathbf{I}_d)$ et calculons le biais attendu et la variance attendue comme :

$$\mathbf{Biais}_\lambda^2 = \mathbb{E}_\theta\mathbf{Biais}_\lambda(\theta)^2 \quad (8)$$

$$\mathbf{Variance}\lambda = \mathbb{E}_\theta\mathbf{Variance}_\lambda(\theta) \quad (9)$$

La littérature scientifique suggère que le risque et la variance atteignent un pic au seuil d'interpolation (n = p). Dans le régime où n est très grand, le risque ne présente plus de pic, mais le modèle uni-modal de variance tient toujours. Dans le reste de la section, nous considérons le régime où le n est grand (risque décroissant de façon monotone), et dérivons l'expression précise du biais et de la variance du modèle. À partir de notre expression, nous obtenons l'emplacement où la variance atteint le pic. Pour cela, nous considérons le régime asymptotique suivant de n, p et d.

Hypothèse : Soit $(d, n(d), p(d))$ une suite de trois nombres entiers. On suppose qu'il existe un $\gamma > 0$ tel que :

$\lim_{d\to+\infty} \frac{p(d)}{d} = \gamma$ et $\lim_{d\to+\infty} \frac{n(d)}{d} = +\infty$
Pour simplifier, on posera $n(d) = n$ et $p(d) = p$.

Théorème : Soit une suite $(d, n(d), p(d))_{d=1}^\infty$ qui l'hypothèse et soit $\lambda = \frac{n}{d}\lambda_0$ pour un certain $\lambda_0 > 0$. Ainsi, on obtient les expressions asymptotique du biais et de la variance par :

$$\lim_{d\to\infty} \mathbf{Biais}_\lambda^2 = \frac{1}{4}\phi_3(\lambda_0, \gamma)^2 \quad (10)$$

$$\lim_{d\to\infty} \textbf{\textit{Variance}}_\lambda =$$
$$\begin{cases} \frac{\Phi_1(\lambda_0,\frac{1}{\gamma})}{2\Phi_2(\lambda_0,\frac{1}{\gamma})} - \frac{(1-\gamma)(1-2\gamma)}{2\gamma} - \frac{1}{4}\Phi_3(\lambda_0,\gamma)^2, & \gamma \le 1, \\ \frac{\Phi_1(\lambda_0,\gamma)}{2\Phi_2(\lambda_0,\gamma)} - \frac{\gamma-1}{2} - \frac{1}{4}\Phi_3(\lambda_0,\gamma)^2, & \gamma > 1, \end{cases}$$

where

$$\Phi_1(\lambda_0, \gamma) = \lambda_0(\gamma + 1) + (\gamma - 1)^2,$$
$$\Phi_2(\lambda_0, \gamma) = \sqrt{(\lambda_0 + 1)^2 + 2(\lambda_0 - 1)\gamma + \gamma^2},$$
$$\Phi_3(\lambda_0, \gamma) = \Phi_2(\lambda_0, \gamma) - \lambda_0 - \gamma + 1.$$

Corollaire : (Monotonicité du biais). La dérivée du biais limite attendu dans (9) peut être calculée comme suite :

$$-\frac{(\sqrt{2(\gamma+1)\lambda_0 + (\gamma-1)^2 + \lambda_0^2} - \gamma - \lambda_0 + 1)^2}{2\sqrt{\gamma^2 + 2\gamma(\lambda_0 - 1) + (\lambda_0 + 1)^2}}$$

Corollaire : (Uni-modalité de la variance - petite λ_0 limite): sous les hypothèses du théorème précédent, l'effet du premier ordre de λ_0 sur la variance est donné par :
Le corollaire 2 suggère que lorsque λ_0 est suffisamment petit, la variance du modèle est maximale lorsque $p = \frac{d}{2}$, et l'effet de λ_0 est de décaler légèrement le pic vers $\frac{d}{2} - \lambda_0 d$
D'un point de vue technique, pour calculer la variance dans le cadre de conception aléatoire, nous devons calculer l'espérance élémentaire de certaines matrices aléatoires. Pour cela, nous appliquons la combinatoire du comptage des partitions non croisées pour caractériser l'espérance asymptotique des produits des matrices de Wishart.

$$\lim_{d\to\infty} \mathbb{E}\textbf{\textit{Variance}}_\lambda = \begin{cases} O\left(\lambda_0^2\right), & \gamma > 1, \\ -(\gamma-1)\gamma - 2\gamma\lambda_0 + O\left(\lambda_0^2\right), & o.w. \end{cases}$$

and the risk is given by

$$\lim_{d\to\infty} \mathbb{E}\textbf{\textit{Risk}}_\lambda = \begin{cases} 1-\gamma+O\left(\lambda_0^2\right), & \gamma \le 1, \\ O\left(\lambda_0^2\right), & \gamma > 1. \end{cases}$$

Moreover, up to first order, the peak in the variance is

$$\textbf{\textit{Peak}} = \frac{1}{2} - \lambda_0 + O\left(\lambda_0^2\right).$$

4. Simulation

Pour les simulations, nous avons utilisé le code implémenté dans l'article. Nous avons fait varier plusieurs paramètres (dataset, algorithmes et les fonctions d'activation) en fin de tracer les fonctions de biais, de variances et de perte en fonction de la largeur des réseaux.

4.1. Modèle basé sur les données

Dans cette partie, nous avons fait varier les données utilisées pour ensuite tracer la fonction de perte, le biais et la variance en fonction de la largeur des couches de convolutions. La largeur des couches de convolution définies dans cette partie la complexité du modèle. Ainsi, en faisant varier la largeur des couches de convolutions, nous faisons varier la complexité de notre modèle.

L'architecture utilisée est le réseau de convolutions resnet34; nous avons configuré un taux d'apprentissages $lr = 0.1$ et un nombre d'epoch de 20. Nous avons appliqué un entraînement par étape (taux d'apprentissage de la décroissance d'un facteur 10 toutes les 2 epoch), et utilisé la décroissance du poids $5 * 10^{-4}$.
Nous avons fait varier la complexité du modèle en variant la largeur du réseau. En effet, la largeur du réseau définit le nombre de filtres de convolution : si ω est la largeur du réseau, alors le nombre de filtres de convolution est donné par $[\omega, 2\omega, 4\omega, 8\omega]$. L'apprentissage se fait avec une descente de gradient stochastique avec un momentum de 0.9.

4.1.1. CIFAR10

Ici, nous avons utilisé les données de cifar10 qui sont composées de 60 000 images. Dans ces 60 000 images, nous avons utilisé 10 000 images pour les tests et 50 000 images pour l'apprentissage. Nous avons fait varier la largeur du réseau de 5 à 50 avec des pas de 10, et nous avons obtenu les résultats ci-dessous. Comme on peut le constater, le biais diminue avec la complexité du modèle alors que l'évolution de la variance en fonction de la complexité

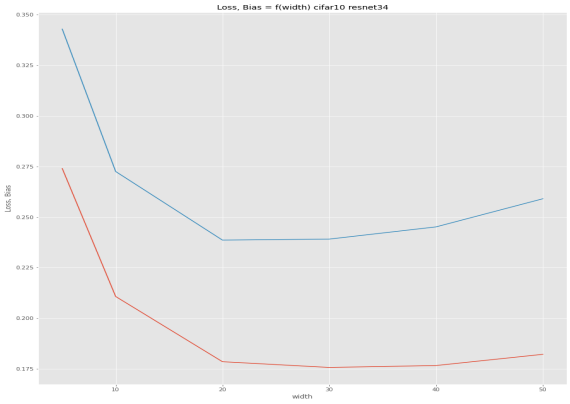


Figure 1. Variation du biais et de la fonction de perte en fonction de la largeur

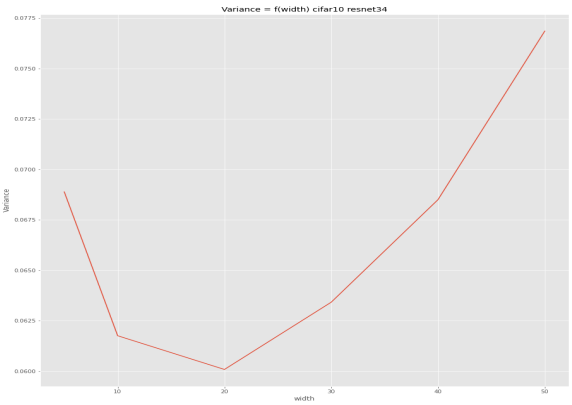


Figure 2. Variation de la variance en fonction de la largeur

du modèle(largeur des réseaux de convolutions) décrit une courbe en forme de cloche. Donc les résultats de l'article sont vérifiés dans ces données. La courbe de la variance ci-dessus ne décrit pas exactement une courbe en forme de cloche, en effet comme on peut le constater, la courbe décrit plutôt une cloche inversée. On a eu beaucoup de difficulté à reproduire l'expérience et avoir les mêmes courbes comme ceux qui sont dans l'article du coup, on a dû l'adapté pour avoir quelques choses de fonctionnelles.

4.1.2. CIFAR100

Les données cifar100 ressemblent beaucoup aux données cifar10 à la différence que ces données contiennent 60 000 images avec 100 classes différentes. Chaque classe est représentée par 500 images dans les données

d'apprentissages et 100 dans les données de test. Les résultats ont été obtenu en faisant varier la largeur du réseau de 10 à 50 comme on peut le constater dans la figure ci-dessous. Ici aussi, nous constatons les résultats de l'article : le biais diminue avec la complexité du modèle alors que la variance est uni-modale.

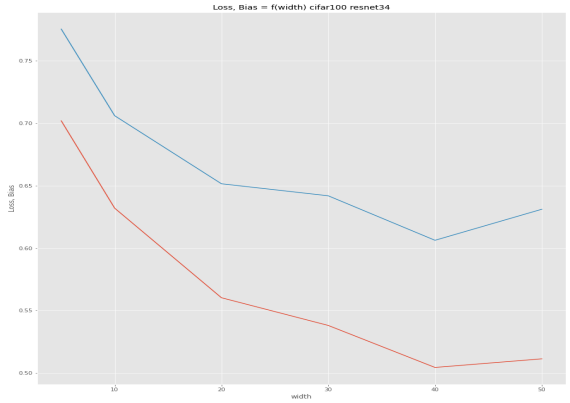


Figure 3. Variation du biais et de la fonction de perte en fonction de la largeur

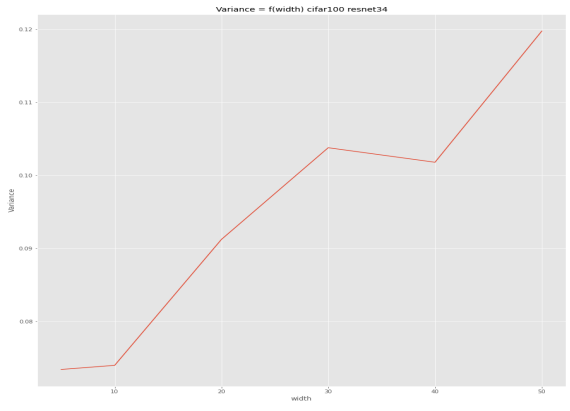


Figure 4. Variation de la variance en fonction de la largeur

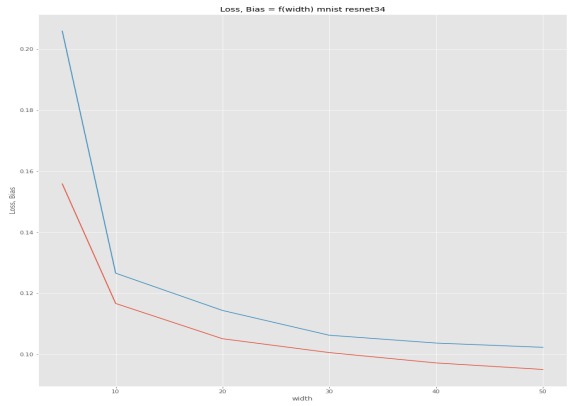


Figure 5. Variation du biais et de la fonction de perte en fonction de la largeur

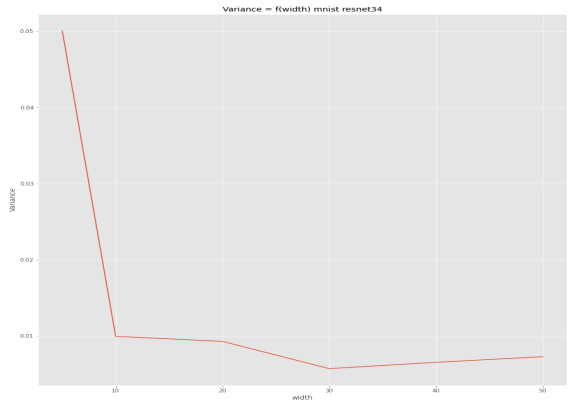


Figure 6. Variation de la variance en fonction de la largeur

4.1.3. MNIST

Pour les tests réalisés sur les célèbres données MNIST, nous avons utilisé les algorithmes implémenté dans le github de l'article scientifique : nous avons utilisé l'architecture DNN avec la fonction de perte MSE. Comme on peut le voir sur les figures ci-dessus, nos simulations valident les résultats de l'article scientifique. On peut constater que le biais diminue en fonction de la complexité du modèle alors que la variance suit une progression uni-modale. Remarquons que pour les données MNIST, l'entraînement des modèles est beaucoup plus rapide et nécessite moins d'epoch : nous avons configure le nombre d'epoch à 10 avec un trial de 2 pour obtenir la convergence.

4.2. Modèle basé sur les algorithmes

Dans cette partie, nous avons utilisé différents algorithmes de classification sur les données cifar10.
Avec les différents algorithmes de ResNet (ResNet18, ResNet34, ResNet50 et VGG) nous avons fait varier la largeur des filtres de convolution. On a ainsi, étudie la variation du biais, de la variance et de la fonction de perte en fonction de la largeur des filtres de convolutions qu'on a fait varier de 10 à 50.

4.2.1. RESNET18

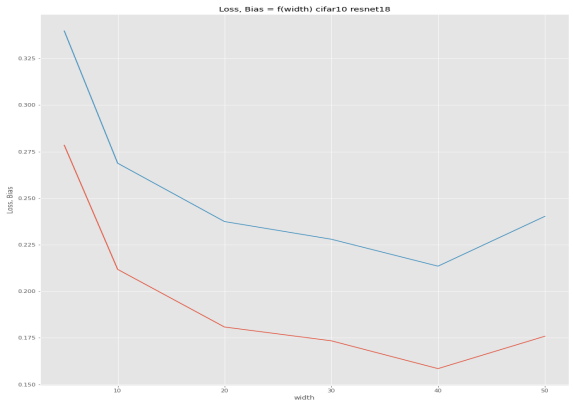


Figure 7. Variation du biais et de la fonction de perte en fonction de la largeur

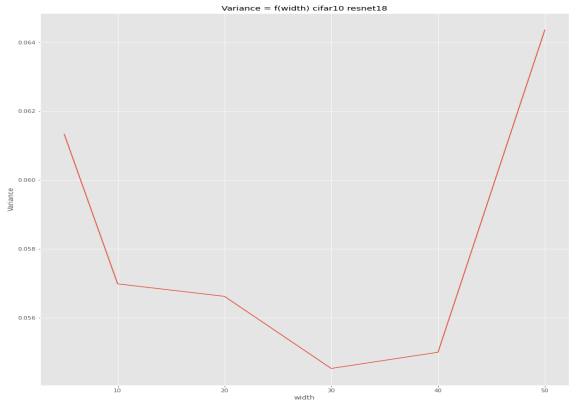


Figure 8. Variation de la variance en fonction de la largeur

Ici, on a utilisé les données cifar10 avec l'architecture Resnet18 auquel on a fait varier la largeur des filtres de convolutions de 10 à 100. On a obtenu les résultats comme représenté dans les deux figures. Comme on peut le constater, le biais diminue en fonction de la largeur du modèle, par contre la variance ne décrit pas exactement une courbe en forme de cloche, mais suit une progression uni-modale. On constate que pour une largeur de 50, on voit une augmentation de la variance, mais cela n'affecte pas les résultats globaux.

4.2.2. VGG

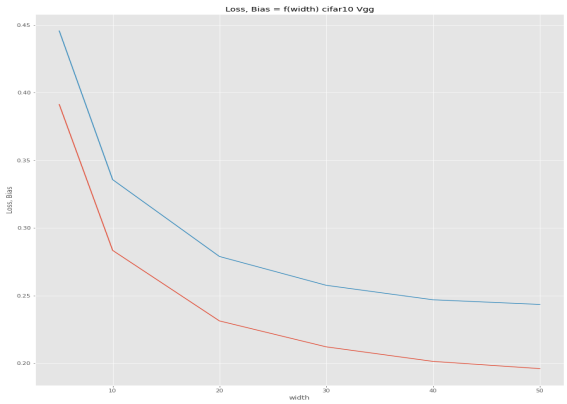


Figure 9. Variation du biais et de la fonction de perte en fonction de la largeur

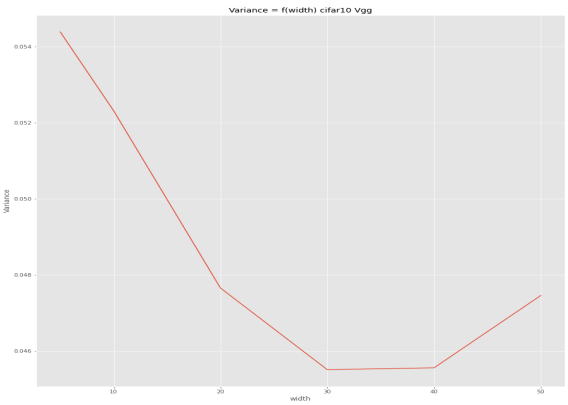


Figure 10. Variation de la variance en fonction de la largeur

Pour l'architecture VGG avec les données cifar10, nous obtenons une courbe de la variance uni-modale et le biais qui diminue en fonction de largeur des filtres de convolutions. Ces résultats confirment encore nos données de l'article.

5. Conclusion

La reproduction des expériences de l'article scientifique est extrêmement difficile et coûteuse en ressources informatique. En effet, il faut un GPU pour faire tourner le CUDA et cela nous n'avons pas des machines dotées de GPU. De ce fait, on a utilisé google colab mais cela ne nous a permi non plus de faire les tests dont on voulait. En effet, google colab limite le temps de GOU qu'on peut utiliser par jour. De ce fait, il nous a fallu des fois arrêter nos algorithmes et les faire tourner le jour suivant.

References

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.