Intelligence artificielle et réseaux neuronaux

THEODON Léo[1], sous la direction de C. FINOT

Résumé
En physique, comme dans tout autre domaine, l'utilisation de solutions analytiques pour résoudre un problème peut s'avérer difficile, si ce n'est impossible. Ce peut être parce qu'il n'existe aucune description analytique complète, ou parce que le calcul numérique se révèle extrêmement coûteux en temps ou en mémoire. L'utilisation de l'intelligence artificielle et des réseaux neuronaux [1] peut alors permettre dans certains cas de proposer des solutions alternatives, pour une très faible complexité spatiale et temporelle. Dans ce rapport, nous exposerons le travail réalisé sous *MATLAB* à l'aide d'une *toolbox* dédiée (Deep Learning Toolbox) [2] avec notamment l'étude de la diffraction d'un bord d'écran.

Mots-clefs
Réseaux-neuronaux — Intelligence Artificielle — MATLAB — Optique

[1] *Département de Physique, Université de Bourgogne - Franche Comté, Dijon, France*

Table des matières

Introduction 1
1 Réseaux neuronaux 1
2 Diffraction d'un bord d'écran infini 2
2.1 Résolution analytique 2
2.2 Utilisation d'un réseau neuronal 3
Configuration et entraînement • Amélioration du modèle
2.3 Problème inverse 5
Avec un réseau neuronal • Critique des résultats
3 Performances des réseaux neuronaux 6
3.1 Taille des échantillons 6
3.2 Nombre de neurones 7
3.3 Analyse 7
Conclusion 7
Remerciements 8
Références 8

Introduction

L'utilisation de la machine pour résoudre des problèmes est devenue au cours des dernières décennies un véritable automatisme, les capacités de calcul offertes tant par les super-calculateurs que par nos ordinateurs personnels étant largement supérieures à celles du cerveau humain. Néanmoins, certaines tâches peuvent s'avérer complexes, même pour les machines les plus puissantes, et nécessiter plusieurs jours de calcul, ainsi que des capacités en mémoire phénoménales. Les réseaux neuronaux offrent alors une solution à la fois simple et légère, ces derniers ne nécessitant que peu de mémoire et donnant des résultats en un temps extrêmement rapide [1].

1. Les calculs effectués par un réseau neuronal à une couche peuvent être réduits à de simples calculs matriciels modulés par une fonction d'activation.

Pour illustrer ce propos, nous nous proposons d'entraîner un réseau neuronal à reconnaitre des figures observées lors de la diffraction de la lumière sur un bord d'écran infini pour différentes longueurs d'onde. Nous tenterons alors d'apprécier dans quelle mesure ce dernier se montrera capable de prédire une figure de diffraction lorsque l'on lui donne une longueur d'onde et une distance en entrée. Nous discuterons aussi de la robustesse du réseau lorsqu'il est confronté à un signal bruité, de la possibilité d'effectuer l'opération inverse (reconnaissance de la longueur d'onde en fonction de la figure de diffraction), et de l'influence qu'ont les caractéristiques propres du réseau (nombre de nœuds, de couches, algorithme de descente, ...) sur ses performances.

Faire appel à un réseau neuronal pour traiter ce genre de problèmes n'a que peu d'intérêt pratique étant donné que le calcul de la solution analytique peut s'effectuer de manière extrêmement rapide. Néanmoins, il permet de montrer sa viabilité lorsque ce même calcul est coûteux ; le réseau permettant alors d'interpoler l'ensemble des solutions à partir d'un nombre restreint de données pré-calculées sur lesquelles il aura pu s'entraîner.

1. Réseaux neuronaux

Un réseau neuronal peut être vu comme un graphe (Fig. 1) prenant un certain nombre d'éléments en entrée (*inputs*) afin d'offrir un ou plusieurs résultats obtenus en sortie (*output*). Les nœuds le composant sont appelés les *neurones*. De fait, on peut constater que sa conception s'inspire du fonctionnement des neurones biologiques, les liaisons entre chaque couches (*layer*) de neurones étant l'analogue des synapses du cerveau. Tout comme le cerveau humain, le réseau neuronal doit s'entrainer sur un ensemble de données avant de pouvoir produire des résultats cohérents et être utilisé. Ainsi, il est particulièrement utile pour la reconnaissance de motifs, en

particulier la reconnaissance de caractères manuscrits, mais pas uniquement. La principe est simple. Chaque arrête du graphe va être affectée d'une certaine valeur (ou *poids*) permettant de déterminer la valeur attribuée à un neurone de la couche cachée à partir des données d'entrée que l'on module en fonction de ces mêmes poids avant de leur appliquer une fonction d'activation [2]. La sortie est alors obtenue par combinaison linéaire de la valeur de chaque neurone, modulée à son tour par une fonction d'activation.

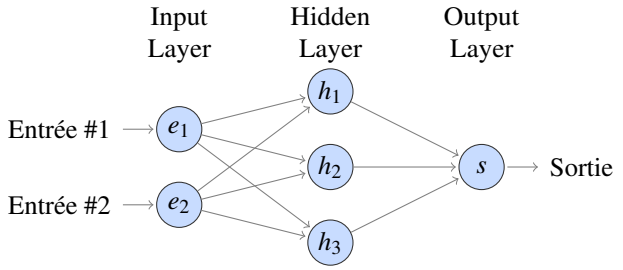

FIGURE 1. Représentation d'un réseau neuronal.

Afin de s'améliorer, le réseau neuronal détermine l'erreur commise entre les résultats obtenus en sortie et les données qui lui sont fournies afin de s'entrainer. Il modifie ensuite les poids affectés à chaque branche par *back-propagation* (ou rétro-propagation du gradient) selon différentes méthodes. Une fois tous les poids corrigés, il effectue un nouveau cycle, et ce, jusqu'à ce que la condition d'arrêt soit atteinte (une erreur seuil, une absence de progression, etc ...).

Deep Learning Toolbox *MATLAB* offre quant à lui une toolbox permettant de générer un réseau neuronal de façon totalement transparente pour l'utilisateur. En effet, ce dernier n'a qu'à spécifier les données à partir desquelles le réseau doit s'entrainer, à savoir l'ensemble des entrées ainsi que les valeurs de sortie correspondantes. Il suffit ensuite de choisir les caractéristiques du réseau, comme le nombre de neurones souhaité pour la couche cachée, de spécifier un algorithme préféré pour la descente du gradient (à savoir la minimisation de l'erreur) tel que Levenberg-Marquardt, Gauss-Newton, Gradient conjugué, etc... et de choisir la taille des échantillons utilisés pour l'entrainement du réseau ainsi que pour la validation des résultats [3]. Une fois ces paramètres définis, il ne reste plus qu'à commencer l'entrainement du réseau. Celui-ci se termine lorsque l'erreur minimale fixée est atteinte, ou lorsque la progression devient constante. Il est possible de fixer d'autres critères d'arrêts comme un temps maximum, ou un nombre de cycle à ne pas dépasser afin d'éviter les boucles infinies (bien que l'utilisateur puisse à tout moment stopper l'entraînement s'il le souhaite). Enfin, le réseau alors entraîné est directement utilisable dans l'environnement *MATLAB* sous la forme d'une fonction. Il est néanmoins également possible de produire un script permettant de mieux contrôler les paramètres du réseau, et en particulier le nombre de couches cachées.

2. Une fonction d'activation est « en général » une fonction de classe au moins C_1 à valeur dans $[0;1]$, monotone et croissante. Ses propriétés peuvent varier selon l'effet souhaité. La fonction arctan, la sigmoïde ou encore la fonction de Heaviside sont toutes trois des fonctions d'activation.

3. En effet, le réseau neuronal s'entraine et se teste sur deux ensemble distincts afin d'obtenir une valeur plus pertinente de l'écart type, et surtout éviter un phénomène de sur-entraînement qui conduit à la création d'un réseau très fiable pour les valeurs d'entrées qu'il connaît, et peu performant pour tout ce qui lui est inconnu. Typiquement, nous avons choisi de conserver 70% de l'échantillon pour l'entrainement du réseau, et 30% pour les tests et la validation des résultats (les valeurs de validation étant également utilisées dans la phase de rétro-propagation afin d'améliorer les performances).

2. Diffraction d'un bord d'écran infini

2.1 Résolution analytique

Afin d'apprendre à maîtriser la *toolbox* de *MATLAB* permettant de générer des réseaux neuronaux, nous allons nous pencher sur un problème simple et bien connu des étudiants de L3, à savoir la figure de diffraction obtenue par un bord d'écran infini.

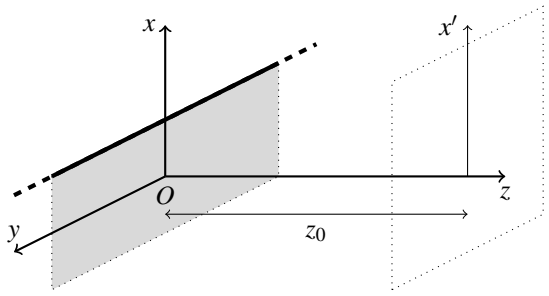

FIGURE 2. Dispositif d'observation de la figure d'interférence sur un écran placé à distance z_0 de la pupille de diffraction.

Sur la figure (2), on rappelle le dispositif permettant l'observation d'une figure de diffraction liée à un bord d'écran infini dans la direction O_y sur un écran placé à une distance z_0 de la pupille. Bien que l'intensité de la figure de diffraction puisse être obtenue par le calcul de la transformée de Fourier de la fonction de Heaviside dans le domaine de Fraunhofer [4], il convient de rappeler que cette dernière peut être décrite de manière complète à l'aide des intégrales de Fresnel.

$$I(x') = \frac{1}{2} I_0 \left[\left(\frac{1}{2} + C_f(u) \right)^2 + \left(\frac{1}{2} + S_f(u) \right)^2 \right] \qquad (1)$$

Dans l'équation (1), les fonctions C_f et S_f sont les fonctions de Fresnel et on a $u = \sqrt{\frac{2}{\lambda z_0}}(x - x')$. Il est donc possible de calculer grâce à *MATLAB* les courbes de diffraction obtenues pour différentes longueurs d'ondes λ et pour plusieurs valeurs de z_0. Nous ferons le choix d'un bord d'écran placé à l'origine $(x = 0)$, de choisir $\lambda \in [400\text{ nm}; 800\text{ nm}]$ et $z_0 \in [0{,}8\text{ m}; 1\text{ m}]$.

4. Le domaine de Fraunhofer, ou région de champ lointain, permet d'approximer l'expression de l'amplitude du champ électromagnétique. Cette dernière devient alors proportionnelle à la transformée de Fourier de la fonction pupillaire. Cette approximation est valable lorsque le nombre de Fresnel F est supérieur à $\frac{1}{2}$, c'est-à-dire que la distance entre plan objet et image est suffisamment grande. Nous ne feront pas cette approximation ici.

La figure (3) permet d'illustrer la forme de la courbe d'intensité que l'on obtient pour $\lambda = 500$ nm et $z_0 = 1$ m.

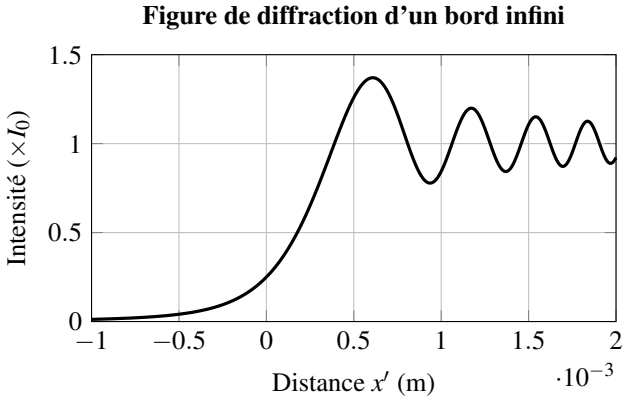

FIGURE 3. Intensité en fonction de la distance x' (m) pour $\lambda = 500$ nm et $z_0 = 1$ m.

2.2 Utilisation d'un réseau neuronal

2.2.1 Configuration et entraînement

Nous savons à présent calculer de manière analytique les courbes de diffraction pour différentes longueurs d'onde λ et distances z_0. Il convient alors d'utiliser la *toolbox* proposée par *MATLAB* afin d'entrainer un réseau neuronal à associer ces courbes à des couples (λ, z_0). L'objectif est de pouvoir obtenir une courbe en sortie pour n'importe quel couple une fois que le réseau neuronal aura été suffisamment bien entrainé. Pour ce faire, nous allons générer un premier échantillon sous la forme d'une matrice R de taille (100×100), chaque ligne contenant une courbe de diffraction, et une matrice VAL de dimension (100×2) contenant quant à elle les couples (λ, z_0) de sorte à ce que la courbe $R(i,-)$ corresponde au couple de valeurs de la ligne i de la matrice VAL. Nous produisons donc un échantillon de taille 100 sur lequel notre réseau neuronal va s'entraîner. En l'occurrence, nous allons choisir dix longueurs d'onde différentes entre 400 nm et 800 nm et dix distances z_0 différentes entre 80 cm et 1 m. Pour ce premier essai, nous choisissons une couche cachée contenant 10 neurones, et nous aurons donc deux entrées (les lignes de la matrice VAL) et 100 sorties (chaque courbe étant échantillonnée par 100 points). L'algorithme utilisé pour la minimisation de l'erreur et la rétropropagation sera celui de Levenberg-Marquardt, préféré pour sa rapidité. Dans ces conditions, le réseau arrête de s'entraîner au bout de 12 itérations, la condition d'arrêt étant qu'il a cessé de progresser sur l'échantillon de test (représentant 15% du total) depuis 6 itérations successives, avec un coefficient de corrélation [5] linéaire $r = 0,9934$ et une erreur moyenne (mse [6]) égale à $0,158$.

On peut constater (Fig. 4) que le réseau est capable de fournir des courbes cohérentes avec la théorie pour des longueurs d'ondes et des distances appartenant aux intervalles sur lesquels il a été entraîné. En revanche, une réelle différence apparait lorsque l'on sort de cet intervalle, en choisissant $\lambda = 1000$ nm par exemple (Fig. 4b), même si l'allure générale ressemble toujours à l'intensité obtenue pour une figure d'interférence sur un bord infini. De plus, on peut constater un certain crénelage de la courbe. Cela peut être dû à deux facteurs en particulier, à savoir l'échantillonnage de courbes (100 échantillons étant peut-être trop peu) et le nombre de nœuds choisi pour la couche cachée du réseau, à savoir 10. Nous discuterons de ces paramètres plus tard en augmentant le nombre de nœuds et en utilisant des réseaux multi-couches.

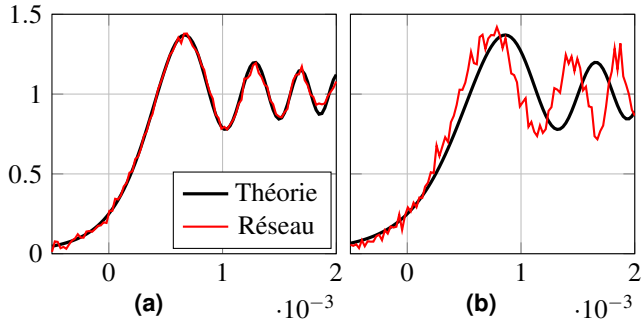

FIGURE 4. Comparaison entre théorie et réseau neuronal avec $z_0 = 1$ m et $\lambda = 500$ nm (4a) et pour $\lambda = 1000$ nm (4b).

2.2.2 Amélioration du modèle

Après ce premier essai, nous nous sommes posé la question de la pertinence de l'entrainement du réseau sur un échantillon *parfait*. En effet, un signal, quel qu'il soit, est toujours au moins légèrement bruité. Il convenait alors de tester la robustesse du réseau neuronal lorsqu'il serait confronté à des échantillons bruités. Plusieurs possibilités s'offraient à nous :

1. Lisser l'échantillon bruité avant que le réseau neuronal ne commence à s'entrainer.
2. Laisser le réseau neuronal s'entrainer sur les courbes bruitées et n'effectuer l'opération de lissage qu'en sortie afin de pouvoir comparer avec la solution analytique attendue.
3. Adopter un lissage en effectuant une moyenne pondérée par rapport aux plus proches voisins, simple ou multi-passe, ou en appliquant un filtre passe-bas dans l'espace de Fourier [7]. La première solution sera retenue, l'utilisation de la TF n'étant plus efficace que pour un signal fortement échantillonné. Étant donné le nombre important de calculs effectués par nos machines, nous ne pouvions nous permettre de choisir un nombre de points trop élevé pour la définition de nos courbes, et de fait, un calcul de moyenne se révèlerait suffisant pour un lissage acceptable.

5. Le coefficient de corrélation *linéaire* r (et de détermination r^2) est calculé à partir de l'ensemble des données de l'échantillon (valeurs d'entrainement, valeurs test et valeurs de validation).

6. *mse* = "*mean squared error*" ou "*erreur quadratique moyenne*".

7. Passer dans l'espace des fréquences en raisonnant sur la transformée de Fourier du signal spatial, et appliquer un filtre passe-bas, permet de supprimer les hautes fréquences qui sont responsables du bruit. Ainsi, en calculant la TF inverse du signal filtré, on récupère une courbe lissée.

4. Appliquer un niveau de bruit uniforme sur l'ensemble de l'échantillon ou définir des seuils minimum et maximum, et appliquer un niveau de bruit aléatoire. C'est cette seconde proposition qui sera privilégiée.

Nous procédons alors à la génération d'un nouvel échantillon dont nous indiquons les caractéristiques dans le tableau ci-dessous. Ce dernier sera constitué de courbes correspondant parfois au même couple $(\lambda; z_0)$ mais pour un niveau de bruit différent.

Caractéristiques	
Intervalle d'observation	$x' \in [-1\text{ mm}; 2\text{ mm}]$
Longueur d'onde	$300\text{ nm} \leq \lambda \leq 900\text{ nm}$
Distance à l'écran	$80\text{ cm} \leq z_0 \leq 1\text{ m}$
Échantillon d'entraînement	225 échantillons
Bruit additionnel	10% à 15%
Lissage	Deux passes, moyenne [8]
Échantillonnage des courbes	100 points par courbe

Nous choisissons ainsi de produire des courbes bruitées qui seront lissées avant que notre nouveau réseau ne les utilise pour s'entraîner. Nous relançons ensuite l'entraînement de notre réseau avec de nouveaux paramètres, indiqués dans le tableau suivant.

Paramètres du réseau neuronal	
Descente de gradient	Levenberg-Marquardt
Couches cachées	2
Neurones par couches	10 (sur chaque couches)
Condition d'arrêt	Erreur constante [9]
Arrêt	61 cycles
Coefficient de corrélation r	0,99553
Performance (mse)	0,002

Le choix d'avoir deux couches de dix neurones plutôt que simplement augmenter le nombre de neurones d'une unique couche cachée se justifie par deux arguments. Premièrement, pour un même nombre de neurones, avoir plusieurs couches semble dans la plupart des cas se révéler plus efficace (de manière exponentielle si l'on en croit la littérature). Nous en discuterons plus tard. Deuxièmement, les réseaux à une couche sont incapables de résoudre correctement les problèmes non-linéaires [10] tels que celui que nous cherchons à résoudre.

Le temps de calcul et le nombre d'itérations nécessaires à l'entraînement du réseau neuronal sont cette fois légèrement plus élevés que lors de notre premier essai. Néanmoins, la taille de l'échantillon test, ainsi que le nombre de neurones a également augmenté, et le fait d'avoir deux couches implique des calculs de rétro-propagation supplémentaires. Une fois cette étape terminée, le coefficient de corrélation obtenu et la performance annoncée semblent prometteurs [11] (cf. tableau précédent).

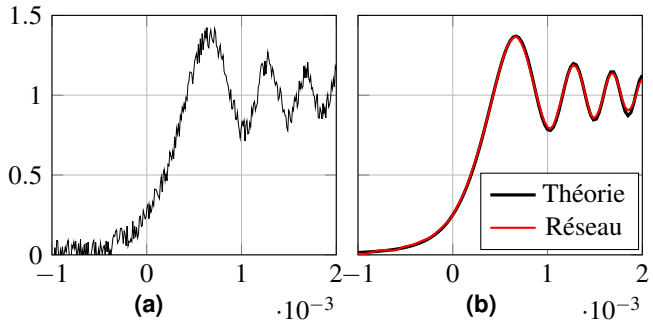

FIGURE 5. Exemple de signal bruité (5a) et courbe théorique face à la prédiction du réseau (5b) pour $\lambda = 655$ nm.

Sur la figure (5), on peut voir un exemple de signal bruité sur lequel le réseau s'est entraîné, et le résultat produit par ce dernier pour la même longueur d'onde. On peut constater que le lissage permet d'obtenir un résultat pratiquement identique à la courbe analytique.

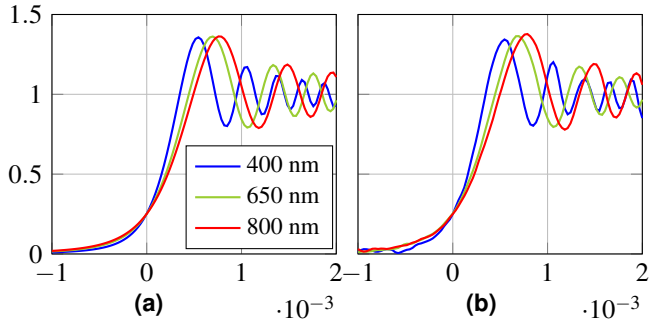

FIGURE 6. Courbes de diffraction théoriques (6a) et produites par le réseau neuronal (6b) pour trois longueurs d'ondes.

La figure (6) nous permet de comparer des figures de diffraction théoriques avec des prédictions du réseau neuronal pour trois longueurs d'ondes différentes du spectre visible. On peut constater que les résultats sont extrêmement proches, même si l'amplitude maximale varie légèrement pour le réseau par rapport à ce qui est attendu.

Surentraînement Un problème qui peut apparaitre lors de l'entraînement du réseau neuronal est le surentraînement. Pour bien comprendre ce phénomène, il faut rappeler comment procède le réseau afin d'améliorer ses performances. De manière schématique, l'échantillon contenant les données d'entraînement est scindé en trois groupes [12].

1. Un groupe d'entrainement (*training*) sur lequel le réseau va s'entraîner en se testant et dont il se sert pour cal-

8. Une première passe en prenant en compte les deux plus proches voisins, puis une deuxième en affectant les quatre plus proches voisins d'un coefficient différent.

9. Le réseau de neurones arrête l'entraînement si l'erreur sur les valeurs de validation ne progresse par après six cycles consécutifs.

10. La production d'un courbe de diffraction en fonction d'un couple $(\lambda; z_0)$ ne se résume pas à de simples opérations matricielles.

11. Plus l'erreur quadratique moyenne est faible, et plus le coefficient de corrélation est proche de 1, plus le réseau neuronal est performant.

12. Dans notre cas, la répartition était de 70%-15%-15%.

culer l'erreur moyenne et corriger les poids affectés à chacune des arrêtes.
2. Un groupe de validation (*valid*) que le réseau va utiliser uniquement en fin de cycle, après avoir calculé l'erreur moyenne sur le groupe *training*. Les résultats seront néanmoins utilisés dans la phase de rétro-propagation et afin d'arrêter l'entraînement lorsque le réseau cesse de s'améliorer.
3. Un groupe *test* que le réseau utilise uniquement pour calculer l'erreur totale à la fin de chaque cycle. Les données issues de ces calculs ne sont pas utilisées lors de la phase de rétro-propagation et permettent d'obtenir des mesures non-biaisées de la performance du réseau.

Ainsi, on comprend que l'entraînement prend évidemment fin si l'erreur ne progresse pas sur le groupe de validation, mais on voit aussi apparaitre un danger qui est ce phénomène de surentraînement. En effet, si le réseau continue de s'entraîner sans progresser significativement sur les valeurs de test, il peut devenir expert en ce qui concerne les données qu'il connait (les groupes *training* et *valid*), mais proposer des résultats totalement inutilisables et incohérents pour toute autre entrée. Il convient donc d'être vigilant et de tester le réseau sur des données autres que celles avec lesquelles il s'est entraîné. C'est notamment le cas sur la figure (6) où la longueur d'onde $\lambda = 650$ nm ne faisait pas partie des données d'entraînement. Néanmoins, on peut constater que les deux courbes (théorique, et calculée par le réseau), sont pratiquement identiques. En effectuant des tests similaires avec des valeurs d'entrée différentes, on peut se convaincre du fait que notre réseau neuronal n'est pas surentraîné.

2.3 Problème inverse

2.3.1 Avec un réseau neuronal

Après avoir été convaincu par la pertinence de l'utilisation d'un réseau neuronal afin d'obtenir l'intensité d'une figure de diffraction étant données la longueur d'onde λ et la distance à l'écran z_0, nous nous sommes penchés sur le problème inverse. En effet, un réseau neuronal saurait-il nous donner un couple (λ, z_0) étant donnée la courbe d'intensité d'une figure de diffraction d'un bord d'écran infini ? Le réseau neuronal aurait alors n entrées (où n représente le nombre de points avec lesquels la courbe est échantillonnée) et deux sorties, λ et z_0. Néanmoins, si l'on analyse le problème un peu plus finement, on peut sentir que cette approche ne fonctionne pas. En effet, la courbe d'intensité est décrite par l'équation (1). Dans cette équation, on peut voir que l'allure de la courbe ne va dépendre que du rapport entre les variables u et x', et que ce rapport ne dépend que du produit λz_0. Ainsi, à chaque courbe, on peut associer une infinité de couples (λ, z_0) différents. De fait, il convient de fixer l'un de ces deux paramètres afin de pouvoir entraîner le réseau neuronal dans de bonnes conditions, et nous avons décidé de prendre $z_0 = 1$ m. Nous avons par conséquent un réseau neuronal (Fig. 7) disposant de n entrées (le nombre de points nécessaires à la description d'une courbe) et une sortie (la longueur d'onde λ pour $z_0 = 1$ m fixé). Les caractéristiques de l'échantillon utilisé pour l'entraîner sont indiquées dans le tableau suivant.

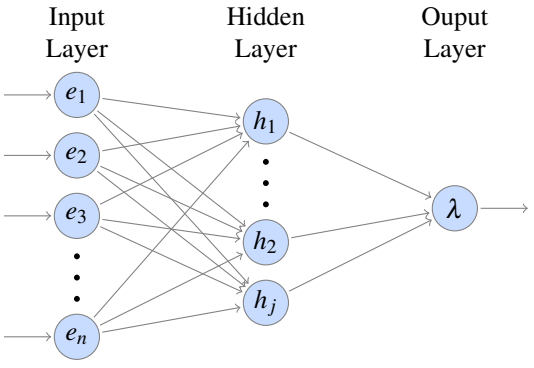

FIGURE 7. Structure du réseau pour le problème inverse.

Caractéristiques	
Intervalle d'observation	$x' \in [-1\text{ mm}; 2\text{ mm}]$
Longueur d'onde	300 nm$\leq \lambda \leq$ 900 nm
Distance à l'écran	1 m (fixée)
Échantillon d'entraînement	225 échantillons
Bruit additionnel	Aucun
Lissage	Aucun
Échantillonnage des courbes	100 points par courbe

Entraînement du réseau Jusqu'à présent, nous avions toujours utilisé l'algorithme de Levenberg-Marquardt afin d'entraîner nos réseaux neuronaux. Curieusement, le premier algorithme s'est montré tout simplement incapable de fournir autre chose qu'un réseau surentraîné, donnant des valeurs complètement aberrantes sur les ensembles de test et de validation. Nous avons donc choisi un algorithme différent, appelé Bayesian Regularization, qui, bien que réputé plus lent, semble offrir de bien meilleures performances, dans la plupart des situations [3]. Ainsi, nous avons décidé d'entraîner un réseau composé d'une seule couche cachée contenant 30 neurones avec cette méthode.

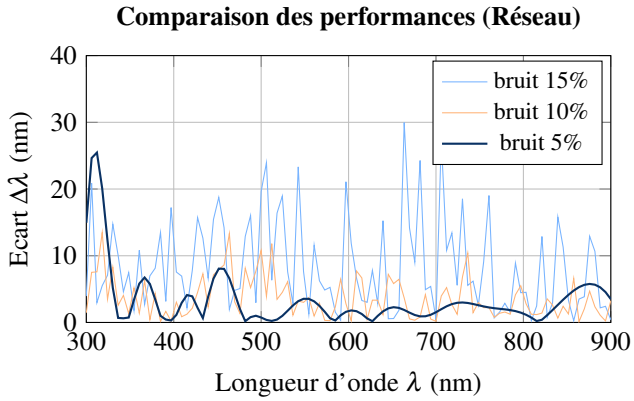

FIGURE 8. Comparaison de l'erreur commise par le réseau neuronal avec des courbes bruitées en entrée pour des longueurs d'onde comprises entre 300 nm et 900 nm.

Une fois notre réseau entraîné, nous avons souhaité le tester sur un large échantillon de courbes définies pour des lon-

gueurs d'onde situées entre 300nm et 900nm. Nous avons donc généré des ensembles de tests composés de 100 courbes chacun et correspondant à des niveaux de bruits différents (5%, 10% et 15%). Notre objectif était de mesurer la robustesse du réseau face à des courbes de plus en plus bruitées, et ne correspondant pas à ce sur quoi il s'était entraîné. On peut constater (Fig. 8) que pour un niveau de bruit relativement faible (5%), l'erreur est pratiquement toujours inférieure à 10nm (entre 1% et 3%) et voir même inférieure à 5nm.

Performances (Réseau neuronal)			
Niveau de bruit	5%	10%	15%
Erreur moyenne ($\Delta\lambda$) [13]	3,6 nm	8,1 nm	12 nm

On pourra regretter les forts écarts pour $\lambda < 350$nm, cela s'expliquant peut-être par la largeur de l'intervalle sur lequel l'étude est effectuée. Néanmoins, il apparait que la fiabilité diminue fortement avec des niveaux de bruit plus élevés (10% et 15%). Un pré-lissage avant de proposer la courbe au réseau neuronal permet toutefois d'atténuer cet effet.

2.3.2 Critique des résultats

Afin d'être plus à même de situer le niveau de performance de notre réseau neuronal, il a nous a a semblé opportun de le mettre en concurrence avec une autre méthode. Par conséquent, nous avons pensé à l'utilisation du produit de corrélation, défini par l'équation (2), afin de déterminer la longueur d'onde associée à chaque courbe de nos ensembles bruités en les comparant aux courbes de départ.

$$Cor_{s,r}(t) = s(x) \otimes r(x) = \int_{-\infty}^{+\infty} s(x) r^*(x-t) dx \qquad (2)$$

En particulier, puisque $Cor_{s,s}(t) < Cor_{s,s}(0)$ pour toute fonction $s(x)$, nous avons pu facilement obtenir la fonction proposant la meilleure corrélation par une simple recherche de maximum, et ainsi en déduire la longueur d'onde supposée du signal bruité.

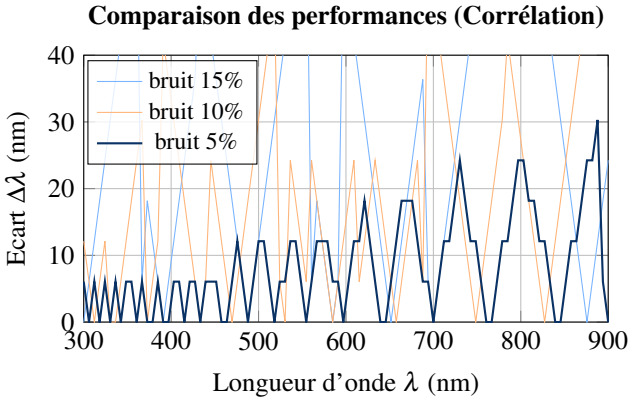

FIGURE 9. Comparaison de l'erreur commise avec la méthode du produit de corrélation.

Malheureusement, même pour des signaux non-bruités, il nous a fallu augmenter la taille de l'intervalle de définition et choisir $x' \in [-1mm; 4mm]$ afin d'obtenir une parfaite correspondance des longueurs d'onde. De plus, comme on peut le constater (Fig. 9), les résultats sont nettement moins satisfaisants qu'avec le réseau neuronal en ce qui concerne l'erreur moyenne. De surcroît, il faut y ajouter les temps de calcul nettement supérieurs.

Performances (Produit de corrélation)			
Niveau de bruit	5%	10%	15%
Erreur moyenne ($\Delta\lambda$)	8,3 nm	19,6 nm	32,6 nm

Dans ces conditions, il nous est apparu que le réseau neuronal était plus robuste, en particulier concernant le traitement de signal bruité. Néanmoins, les résultats sont bien moins spectaculaires que pour le premier problème, à savoir la génération d'un courbe étant donnés deux paramètres λ et z_0.

3. Performances des réseaux neuronaux

Après nous être familiarisés avec la manipulation des réseaux neuronaux, nous nous sommes intéressés à leurs performances, et en particulier, à l'influence que peuvent avoir différents paramètres tels que la taille de l'ensemble sur lequel le réseau effectue son entraînement, le nombre de neurones par couches ou encore que le nombre de couches. Nous avons donc programmé une fonction permettant de générer à la volée et en parallèle (afin d'optimiser le temps de calcul) des échantillons de taille variable, et d'entrainer des réseaux neuronaux pour un nombre de couches cachées allant de 1 à 4 et possédant chacune entre 4 et 20 neurones. Nous avons ainsi pu collecter un grand nombre de données que nous présentons dans les parties suivantes. L'indice de performance retenu sera la *mse*, calculée non pas sur l'ensemble sur lequel le réseau s'est entraîné, afin d'éviter le biais évident [14] découlant du fait que ce dernier sera toujours plus performant sur les ensembles connus, mais sur un groupe test de même taille que l'ensemble des données servant à l'entraînement du réseau. Enfin, le cas de test sera celui de la partie 2.2, à savoir la production d'une courbe en fonction de deux paramètres : λ et z_0.

3.1 Taille des échantillons

Dans un premier temps, il nous est apparu important de discuter de l'influence de la taille de l'échantillon avec lequel s'entraîne le réseau sur ses performances. Nous avons choisi de fixer le niveau d'échantillonnage des courbes à 100 points afin de ne pas surcharger le processeur, cette valeur offrant jusqu'à présent des résultats très satisfaisants. Des tests ont été effectués (Fig. 10) pour des réseaux possédant entre une et quatre couches cachées, avec dix neurones par couche. Dès lors, on peut en tirer deux conclusions notables.

1. Les performances augmentent toutes de manière significative avec la taille de l'ensemble des données jusqu'à

13. On a $\Delta\lambda = |\lambda_{network} - \lambda_{theorique}|$. (Écart moyen absolu)

14. Il est important de tester les performances du réseau sur un ensemble de données qui lui sont inconnues. Il faut donc s'assurer que cet ensemble a une intersection nulle avec les données d'entraînement et que sa taille est suffisamment importante pour donner des résultats significatifs. [4]

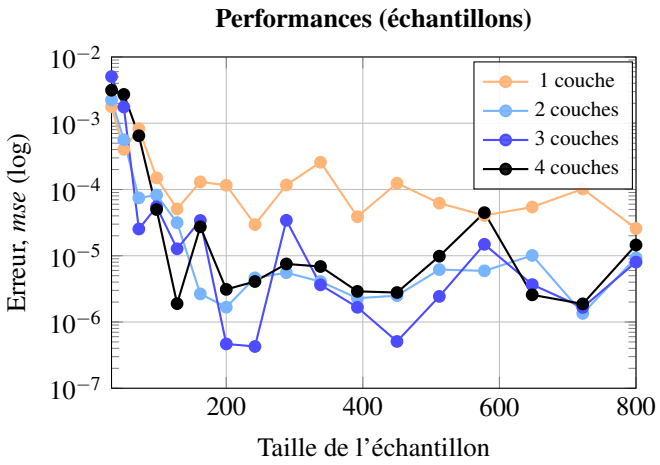

FIGURE 10. Erreur sur l'ensemble des données de test en fonction de la taille de l'échantillon (10 neurones par couche).

ce que celui-ci dépasse les 200 éléments à partir de quoi, ces dernières cessent de s'améliorer et restent confinées dans un intervalle compris entre 10^{-4} et 10^{-6}.

2. S'il apparait clairement qu'un réseau multi-couches propose des performances nettement supérieures à un réseau mono-couche, l'intérêt d'en superposer plus de deux semble limité.

3.2 Nombre de neurones

Pour tester l'influence de l'augmentation du nombre de neurones sur les performances des réseaux, il nous a fallu définir un ensemble de données de test, et notamment fixer sa taille. En accord avec les résultats de la partie précédente et pour limiter le temps de calcul, nous avons choisi de limiter ces ensembles à 128 éléments. De la même manière que précédemment, on peut dégager des résultats (Fig. 11) deux conclusions principales.

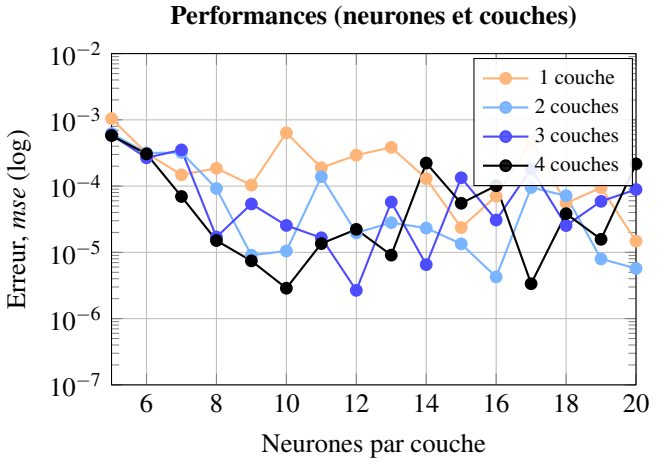

FIGURE 11. Erreur sur l'ensemble des données de test (128 éléments) en fonction du nombre de neurones par couches.

1. Les performances augmentent toutes de manière significative avec le nombre de neurones jusqu'à ce que celui-ci dépasse 10, à partir de quoi ces dernières cessent à nouveau de s'améliorer et oscillent entre 10^{-3} et 10^{-6}.
2. Encore une fois, il existe un écart très net entre l'utilisation d'une seule couche ou de plusieurs. Néanmoins, l'intérêt de l'augmentation du nombre de couches au delà de deux n'est pas évident.

3.3 Analyse

Nos simulations montrent de manière très claire l'influence du nombre de couches cachées, tout comme celle du nombre de neurones, sur les performances du réseau. De même, la taille de l'ensemble des données utilisé pour l'entraînement semble être le paramètre ayant l'influence la plus élevée. Le fait que les performances stagnent au delà de dix neurones, de deux couches cachées, ou d'un ensemble de données de cardinal égal à 200 peut s'expliquer de différentes manières. Premièrement, il ne faut pas oublier l'échantillonnage *limité* des données, à savoir 100 points par courbe, ce choix ayant été effectué pour limiter le temps de calcul. Deuxièmement, d'autres choix de ce type tels que la limitation du nombre de cycle d'entraînement des réseaux (fixé à 200) ou encore le temps de calcul par cycle (fixé à 600 secondes) ont probablement également une influence. Enfin, un élargissement de l'intervalle de définition des courbes améliorerait peut-être également les performances comme ce fut le cas pour la méthode du produit de corrélation (cf. partie 2.3.2). On pourrait aussi discuter du choix de l'algorithme utilisé afin d'entraîner le réseau étant donné que nous avons choisi de privilégier la vitesse au détriment de la précision (Levenberg-Marquardt contre Bayesian Regularization).

Conclusion

L'utilisation de réseaux neuronaux peut permettre de répondre à certains problèmes en proposant des solutions simples, très efficaces, peu coûteuses, que ce soit en terme de mémoire ou de temps de calcul. Néanmoins, il ne s'agit pas d'une solution miracle, comme nous avons pu le constater pour le problème de la détermination d'une longueur d'onde λ étant donnée une courbe d'intensité. De plus, son utilisation dans le cas étudié ne peut être considéré que comme un exemple servant à illustrer un concept, tant les solutions alternatives (comme le fait de passer par le calcul de la transformée de Fourier pour les figures de diffraction) sont efficaces.

En revanche, il nous a été donné d'appréhender un exemple de situation [5] où l'utilisation d'un réseau neuronal prend tout son sens. C'est en particulier le cas pour un problème de détermination des paramètres et conditions de propagation non-linéaire d'impulsions dans un milieu dispersif normal, à savoir une fibre optique. En effet, il n'existe pas de solution analytique, et les calculs numériques se révèlent extrêmement coûteux. L'idée est par conséquent de générer un ensemble de données conséquent par des méthodes numériques et d'utiliser un réseau neuronal afin d'interpoler l'ensemble des solutions en l'entraînant sur l'ensemble des données pré-calculées. On obtient ainsi un outil capable de fournir des résultats

en une fraction de seconde et occupant un espace mémoire négligeable une fois l'entraînement terminé. Cette méthode peut se généraliser à de nombreux problèmes où les solutions analytiques n'existent pas ou se révèlent incapables de produire des données exploitables avec un coût faible.

Enfin, on notera que nous n'avons fait qu'effleurer la surface des possibilités offertes par l'utilisation des réseaux neuronaux. En effet, il existe des réseaux bien plus sophistiqués, avec une construction plus minutieuse des couches cachées, ajoutant des couches de convolution, sélectionnant avec le plus grand soin les fonctions d'activation utilisées entre chaque couches, etc...En particulier, nous avons pu constater l'influence qu'ont chacun des paramètres du réseau, ainsi que de l'ensemble des données d'entraînement, sur les performances de ce dernier. C'est pourquoi il est nécessaire de bien analyser chaque situation et chaque problème afin de choisir les meilleures conditions possibles pour l'entraînement et la génération du réseau qui fournira alors des performances optimales sur des données inconnues.

Remerciements

Merci à Nicolas T. pour son sérieux, Guillaume J. pour sa curiosité, Alexandre O. pour ses nombreuses plaisanteries, M. Finot pour ses explications et l'Université de Bourgogne - Franche Comté pour nous offrir la possibilité de faire progresser la science chaque jour ainsi que la qualité des locaux mis à notre disposition.

Références

[1] N. THOME. Neural Networks and Deep Learning. `https://www.fun-mooc.fr/`.

[2] MathWorks. MATLAB, Deep Learning Toolbox. `https://fr.mathworks.com/products/deep-learning.html`.

[3] M. KAYRI. Predictive abilities of bayesian regularization and levenberg–marquardt algorithms in artificial neural networks : A comparative empirical study on social data. *Mathematical and Computational Applications*, may 2016.

[4] J. M. TWOMEY and A. E. SMITH. Performance measures, consistency, and power for artificial neural network models. *Mathematical and Computer Modelling*, 21 :243–258, 1995.

[5] K. HAMMANI C. FINOT, I. GUKOV and S. BOSCOLO. Nonlinear sculpturing of optical pulses with normally dispersive fiber-based devices. *Optical Fiber Technology*, 45 :306–312, 2018.