République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Amar Thelidji- Laghouat

FACULTÉ: GENIE CIVIL ET D'ARCHITECTURE

DÉPARTEMENT : GENIE CIVIL

MÉMOIRE DE MASTER

Présenté par : CHETTIH Maroua

DOMAINE : Sciences et Techniques

FILIERE : Hydraulique

OPTION : Ressources Hydrauliques

Thème

Deep Learning pour la prévision des précipitations et des débits dans les bassins côtiers Algériens

Jury de soutenance :

Nom et Prénom	Grade	qualité
Bouache Mohamed	MAA	Président
Tadj Walid	MCB	Examinateur
Chettih Mohamed	Pr.	Rapporteur

Promotion : 2020

الملخص

الهدف من عملنا هو تقدير تنبؤات هيدرولوجية تعتمد على مفهوم التعلم العميق. تم استخدام ثلاثة مناهج تعتمد على الشبكات العصبية المتكررة من نوع LSTM القادرة على تعلم التبعيات طويلة الأجل والاحتفاظ بالمعلومات على مدى فترات أطول للتنبؤ بالتدفقات اليومية والأمطار الشهرية في بعض الأحواض الجزائرية. كما تم تمديد التطبيق إلى مؤشرات التذبذب في الغلاف الجوي على نطاق واسع في خطوات شهرية ويومية. أظهر التنبؤ بالتدفقات اليومية أداءً عالياً للغاية حيث وصل معامل الارتباط بين التدفقات المرصودة والتدفقات المحاكاة إلى 0.97. بالنسبة للأمطار الشهرية ، يبدو أن التنبؤ متوسط الجودة ، حيث لم يتجاوز معامل الارتباط القيمة 0.64. يمكن ربط هذه النتيجة بهيكل سلسلة هطول الأمطار ، على الرغم من وجود الموسمية ، فإنها تحتوي دائمًا على جزء كبير من العشوائية. تتصرف مؤشرات تذبذبات الغلاف الجوي NAO و MO في خطوات شهرية كعملية عشوائية ليس لها بنية أو انتظام أو قاعدة تنبؤ يمكن تحديدها ، وقد ثبت أن التنبؤ بها مستحيل. ومع ذلك، على أساس يومي، فإن التنبؤ بالمؤشرات يبدو واعدًا للغاية ويعطي نتائج جيدة جدًا. في نهاية المطاف ، النتائج التي تم الحصول عليها في هذا العمل مشجعة للغاية وذات أهمية كبيرة ولها مساهمة كبيرة حيث تم توضيح أداء التعلم العميق مع شبكات LSTM بوضوح مقارنة بالنتائج التي تم الحصول عليها بواسطة نموذج مع شبكات من الخلايا العصبية الاصطناعية الكلاسيكية من نوع المستقبلات متعدد الطبقات.

الكلمات المفتاحية: *التعلم العميق ، LSTM ، التنبؤ ، المطر ، التدفق ، مؤشر تذبذب المناخ ، الجزائر .*

Summary

The objective of our work is to make hydrological forecasts based on the concept of Deep Learning. Thus, three approaches based on recurrent neural networks of LSTM type capable of learning long-term dependencies and retaining information over longer periods were used for the forecasts of daily flows and monthly rains in some Algerian basins. The application was also subsequently extended to large-scale atmospheric oscillation indices at monthly and daily steps. The forecast of daily flows showed a very high performance where the correlation coefficient between the observed flows and the simulated flows reached 0.97. For the monthly rains, the forecast seems of average quality, as the correlation coefficient did not exceed the value 0.64. This result could be linked to the structure of the rainfall series, despite the presence of seasonality; they always contain a considerable part of stochasticity. Atmospheric oscillations indices NAO and MO at monthly steps behave like a random process that has no structure, regularity or identifiable prediction rule, their prediction proves impossible. However, on a daily basis, the forecasting of the indices looks very promising and gives very good results. Ultimately, the results obtained in this work are very encouraging and very significant and of a substantial contribution where the performance of Deep Learning with LSTM networks has been clearly demonstrated compared to the results obtained by a classic model of artificial neural networks of the Multilayer Perceptron type.

Keywords: *Deep Learning, LSTM, Forecast, Rain, Flow, Climate Oscillation Index, Algeria.*

Résumé

L'objectif de notre travail est de réaliser des prévisions hydrologiques sur la base du concept du Deep Learning. Ainsi, trois approches basées sur les réseaux de neurones récurrents de type LSTM capables d'apprendre les dépendances à long terme et conservent les informations sur des périodes plus longues ont été utilisées pour les prévisions des débits journaliers, des pluies mensuelles dans quelques bassins Algériens. L'application a été aussi étendue par la suite aux indices d'oscillation atmosphériques à grande échelle aux pas mensuels et journaliers. La prévision des débits journaliers a montré une très haute performance où le coefficient de corrélation entre les débits observés et les débits simulés a atteint 0.97. Pour les pluies mensuelles, la prévision semble de moyenne qualité, car le coefficient de corrélation n'a pas dépassé la valeur 0.64. Ce résultat pourrait être lié à la structure des séries de pluies, malgré la présence de saisonnalité, elles renferment toujours une part considérable de stochasticité. Les indices d'oscillations atmosphériques NAO et MO au pas mensuel se comportent comme un processus aléatoire qui n'a aucune structure ni régularité ni règle de prédiction identifiable, leur prévision s'avère impossible. Cependant, au pas journalier, la prévision des indices semble très prometteuse et donne de très bons résultats. En définitive, les résultats obtenus dans ce travail, s'avèrent très encourageants et très significatifs et d'un apport substantiel où la performance du Deep Learning avec des réseaux LSTM a été nettement mise en évidence comparés aux résultats obtenus par un modèle à réseaux de neurones artificiels classique de type Perceptron Multi-couches.

Mots clés : *Deep Learning, LSTM, Prévision, Pluie, Débit, Indice d'Oscillation Climatique, Algérie.*

Dédicaces

Je dédie ce modeste travail à :

Mon père : Mon précieux Aucune dédicace ne saurait exprimer l'estime, le dévouement et le respect que j'ai toujours eu pour lui. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.

Ma mère : Affable, honorable, aimable : Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

A mes sœurs : Mes chères sœurs vous étiez présent dans tous mes moments d'examens par vos soutiens moraux.

A ma famille : *Mes grands-mères, mes tentes et mes oncles qui mon beaucoup encouragé et qui me donnent de la vivacité.*

A tous mes amis : *Qui m'ont toujours encouragé, et à qui je souhaite plus de succès dans leurs vies.*

Remerciements

Table Des Matières

Liste des figures

Liste des tableaux

Remerciements

Tout d'abord, je tiens à remercier Allah, le tout puissant et miséricordieux, qui m'a donné la force, l'intelligence et la patience d'accomplir ce modeste travail.

Je voudrais remercier mon directeur de mémoire M. Chettih, professeur à l'université de Laghouat, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Mes vifs remerciements s'adressent à Monsieur le Président de jury et à tous les membres de jury qui m'ont fait l'honneur d'examiner ce travail.

Je remercie également toute l'équipe pédagogique de l'université de Laghouat qui a contribué à ma formation.

Je tiens à témoigner toute ma reconnaissance aux personnes suivantes, pour leur aide dans la réalisation de ce mémoire :

- Ma sœur, pour avoir lu, relu et corrigé mon exposé. Ses conseils ont été très précieux ;
- Mes parents, pour leur soutien constant et leurs encouragements.

En fin, je tiens à remercier toutes les personnes qui ont contribué de loin ou de près à la réalisation de ce modeste travail.

Table Des Matières

Liste des figures

Liste des tableaux

Introduction Générale

La prévision est l'un des sujets les plus importants en hydrologie, car elle permet de prendre les bonnes décisions, de bien planifier et gérer convenablement les ressources en eau et d'assurer le bon fonctionnement et la protection des infrastructures hydrauliques.

A ce titre, la modélisation mathématique constitue l'outil incontournable pour la prévision hydrologique. L'objectif principal de la modélisation d'une série temporelle hydrologique est la prédiction de l'évolution future de la série à partir de celles qui ont été observées. Néanmoins, les séries hydrologiques sont réputées d'être non-linéaires et non stationnaires (Labat, 2000). Au cours de ces des dernières décennies, un grand nombre d'approches automatisées ou informatisées ont été mises en œuvre pour modéliser ce type de séries.

L'avènement de l'informatique et l'évolution technologique de la puissance de calcul ont beaucoup facilité la formulation d'approches non linéaires et permis d'avoir d'autres alternatives fiables et efficaces. Ainsi, des modèles empiriques non-linéaires issus de l'intelligence artificielle sont apparus pour constituer une nouvelle approche pour modéliser des systèmes complexes tels que les processus hydrologiques.

Parmi ces méthodes, les réseaux neuronaux, qui présentent une technique de calcul pour la prévision hydrologique (Abrahart, 2004),Les réseaux de neurones fournissent une voie de modélisation qui peut être utile quand il ya suffisamment de données (Abrahart, 2004). Malgré leur efficacité, les réseaux de neurones sont souvent considérés comme des solutions de boîte noire. Mais en revanche, ils peuvent produire des résultats très utiles.

Par ailleurs, la combinaison de raisonnement probabiliste, logique floue, réseaux de neurones et algorithmes évolutionnaires, ce que nous appelons communément : le " *Soft Computing* ", constitue une nouvelle technique pour la construction de systèmes hybrides intelligents capables de raisonnement et d'apprentissage dans un environnement incertain et imprécis. L'application des systèmes hybrides intelligents en hydrologie a montré de bonnes performances (Nayak, 2007), (Tayfur, 2012).

Cependant, la complexité des régimes hydrologiques (non-linéaires, non-stationnaires, multi-échelles) requiert à recourir à des outils plus spécifiques et mieux adaptés (Sivakumar et al., 2001). Ainsi, le Deep Learning issu de l'intersection des réseaux de neurones, de l'intelligence artificielle, de la modélisation graphique, de l'optimisation, de la reconnaissance des formes et du traitement du signal trouve toute son importance pour la prévision des séries temporelles.

Par conséquent, la méthode du Deep Learning la plus utilisée pour des problèmes similaires à ceux de l'hydrologie est celle du réseau récurrent à mémoire court et long terme (Hu Caihong et al., 2018) que nous proposons d'utiliser dans cette étude pour la prévision hydrologique.

L'objectif de cette étude est de construire des modèles de type Data Driven basés sur les réseaux de neurones récurrents du Deep Learning permettant de simuler et de prédire les phénomènes hydrologiques à partir des données disponibles aux pas journaliers et mensuels.

Ce travail est composé de trois chapitres, d'une introduction générale, d'une conclusion générale et d'une liste bibliographique. L'introduction générale donne un aperçu du sujet de ce mémoire, expose la problématique et décrit la méthodologie du travail.

Le chapitre 1 est dédié à l'état de l'art du Deep Learning, Machine Learning, à la description des différents types de Réseaux de neurones, aux types du Deep Learning et aux applications du Deep Learning.

Le chapitre 2 est consacré aux réseaux de longue mémoire à court terme, à leur architecture, à leurs propriétés et à leur fonctionnement. Ce chapitre décrit les algorithmes des réseaux, la structure des codes de calculs et quelques applications pour valider les programmes de calculs.

Le chapitre 3 est consacré à la présentation des sites d'étude et des données, ainsi qu'à la présentation des résultats et à leurs interprétations. Pour vérifier l'efficacité des modèles élaborés leurs résultats seront confrontés aux résultats d'autres modèles.

La conclusion générale synthétise les principaux apports de ce mémoire et dégage quelques perspectives qui pourront faire suite à ce travail.

Chapitre 1

Introduction au Deep Learning

I.1 Introduction

Le Deep Learning est basé sur le concept de Machine Learning et le concept d'Intelligence Artificielle. Il peut être défini comme une forme d'Intelligence Artificielle, dérivée du Machine Learning. Le Deep Learning constitue un sous-ensemble du Machine Learning qui est lui-même un sous-ensemble de l'intelligence artificielle et des statistiques.

L'intelligence artificielle est un vaste domaine représentant l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler certains traits de l'intelligence humaine. (Beghdad et Ouserir, 2018) La recherche sur l'intelligence artificielle a commencé peu de temps après la seconde guerre mondiale. Les premiers travaux reposaient sur la connaissance de la structure du cerveau, de la logique propositionnelle et de la théorie de calcul de Turing. L'intelligence artificielle a beaucoup évolué grâce notamment à l'émergence du Cloud Computing et du Big Data, soit d'une puissance de calcul peu coûteuse et de l'accessibilité à un grand nombre de données. (Turing, 1950)

Le concept de Machine Learning date du milieu du 20ème siècle. Dans les années 1950, le mathématicien Alan Turing imagine une machine capable d'apprendre. Au cours des décennies suivantes, différentes techniques d'Apprentissage Machine ont été développées pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome. Le Machine Learning est capable de reproduire un comportement grâce à des algorithmes, eux-mêmes alimentés par un grand nombre de données. Confronté à de nombreuses situations, l'algorithme apprend quelle est la décision à adopter et créé un modèle. (Turing 1950)

Ainsi, avec l'essor des techniques avancées d'apprentissage machine (Beghdad et Ouserir, 2018). l'intelligence artificielle se trouve propulsée au-devant de la scène et le Deep Learning issu de l'intersection des réseaux de neurones, de l'intelligence artificielle, de la modélisation graphique, de l'optimisation, de la reconnaissance des formes et du traitement du signal va prendre toute son importance.

La figure ci-dessous (Fig.01), montre la conception de la relation entre les différentes approches. De l'intelligence artificielle basique au deep Learning, il aura fallu attendre des décennies. Chaque approche évolue de son côté, et permet de renforcer l'autre dans le cas où l'on va plus profondément dans l'apprentissage. Les structures, les fonctionnements, les concepts sont constamment remis en question.

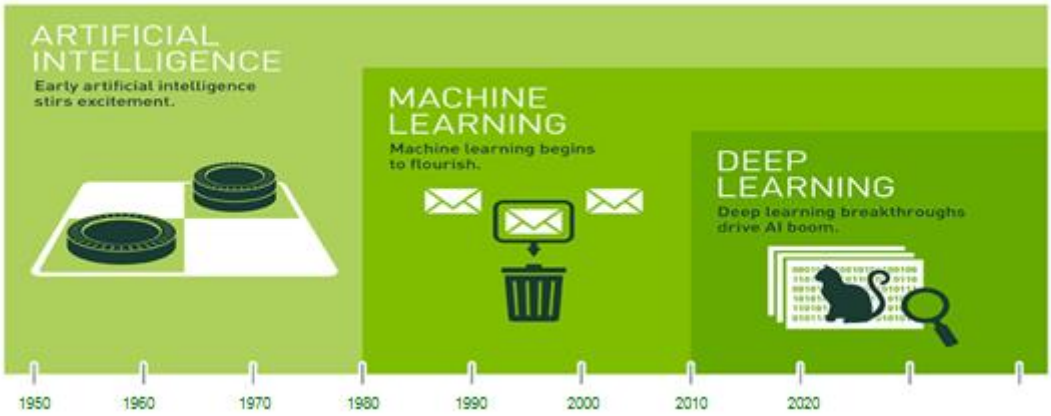


Fig. 01 : - Conception de la relation entre l'Intelligence Artificielle, Machine Learning et le Deep Learning

Les domaines d'application du Deep Learning sont nombreux, on cite à titre d'exemple : la reconnaissance visuelles, la robotique, la bio-informatique, la reconnaissance ou la comparaison de formes, la sécurité, la santé, l'informatique pédagogique, l'art, la traduction et la liste n'est pas exhaustive.

La technologie du Deep Learning est commercialement utilisée dans l'industrie des soins de santé, le traitement d'images médicales, le traitement des langues naturelles et la publicité pour améliorer les taux de clics. Microsoft, Google, IBM, Yahoo, Twitter, Baidu, Paypal et Facebook exploitent tous le Deep Learning pour comprendre les préférences des utilisateurs afin qu'ils puissent recommander des services et des produits ciblés.

I.2 Historique

C'est le développement le plus excitant du monde de l'intelligence artificielle en ce moment. Jetons un coup d'œil à certains des principaux développements de l'histoire de l'apprentissage automatique. Il a parcouru un long chemin en peu de temps.

De toute évidence, pour que la machine et le Deep Learning fonctionnent, nous avions besoin d'une compréhension établie des réseaux neuronaux du cerveau humain.

Walter Pitts, un logicien, et Warren McCulloch (Culloch et Pitts, 1943), un neuroscientifique, nous ont donné cette pièce du puzzle en 1943 lorsqu'ils ont créé le premier modèle mathématique d'un réseau neuronal. Publiés dans leur ouvrage fondateur : *A logical calculus of the ideas immanent in nervous activity*, ils ont proposé une combinaison de mathématiques et d'algorithmes visant à imiter les processus de la pensée humaine.

Ce premier modèle computationnel de neurone artificiel appelé neurones de McCulloch-Pitts est toujours la norme d'aujourd'hui, bien qu'il ait évolué au fil des années.

Alan Turing, un mathématicien britannique, est peut-être le plus connu pour son implication dans le décryptage des codes pendant la seconde guerre mondiale. Par ailleurs, ses contributions aux mathématiques et aux sciences ne s'arrêtent pas uniquement là. En 1947, il prédit le développement de l'apprentissage automatique, allant même jusqu'à décrire l'impact qu'il pourrait avoir sur les emplois. (Turing, 1947)

En 1950, Turing a proposé une telle machine, faisant même allusion à des algorithmes génétiques, dans son article : *Computing Machinery and Intelligence*. Dans ce document, il a conçu ce qui a été surnommé le test de Turing, bien qu'il l'ait lui-même appelé le jeu d'imitation - pour déterminer si un ordinateur peut penser. (Turing, 1950)

En 1952, Arthur Samuel a inventé l'apprentissage automatique ainsi que l'expression : *apprentissage automatique*. En rejoignant le laboratoire Poughkeepsie d'IBM, Arthur Samuel allait créer les premiers programmes d'apprentissage informatique. (Samuel, 1952)

Frank Rosenblatt, un psychologue, a soumis un document intitulé : *Le perceptron, un automate percevant et reconnaissant*, au Cornell Aeronautical Laboratory en 1957. Il a déclaré qu'il construirait un système électronique ou électromécanique qui apprendrait à reconnaître les similitudes ou les identités entre les modèles d'informations optiques, électriques ou tonales, d'une manière qui peut être étroitement analogue aux processus perceptifs d'un cerveau biologique.(Rosenblatt, 1957)

En 1959, les neurophysiologistes et lauréats du prix Nobel David H. Hubel et Torsten Wiesel ont découvert deux types de cellules dans le cortex visuel primaire: les cellules simples et les cellules complexes. De nombreux réseaux de neurones artificiels s'inspirent d'une manière ou d'une autre de ces observations biologiques. Bien qu'il ne s'agisse pas d'un jalon pour le Deep Learning en particulier, il en est certainement un qui a fortement influencé le domaine. (Hubel et Wiesel, 1959)

En 1960, Henry J. Kelley publie un article intitulé : *Gradient Theory of Optimal Flight Paths*, un article important et largement reconnu dans le domaine. Beaucoup de ses idées sur la théorie du contrôle, le comportement des systèmes avec les entrées, et comment ce comportement est modifié par le Feedback, ont été appliquées directement à l'intelligence artificielle et aux réseaux de neurones artificiels au fil des années. Ils ont été utilisés pour développer les bases d'un modèle de rétro propagation continue utilisé dans l'apprentissage des réseaux de neurones. (Kelley, 1960)

Le mathématicien Ivakhnenko et ses associés, ont sans doute créé les premiers réseaux du Deep Learning en 1965, appliquant ce qui n'avait été que des théories et des idées jusqu'à ce moment-là. Ivakhnenko a développé la méthode de groupe de traitement des données (GMDH du *Group Method of Data Handling*), définie comme une famille d'algorithmes inductifs pour la modélisation mathématique par ordinateur d'ensembles de données multi-paramétriques et qui propose une optimisation structurelle et paramétrique entièrement automatique des modèles et l'appliquée aux réseaux de neurones. Ses algorithmes d'apprentissage ont utilisé des perceptrons multicouches à action directe approfondie en utilisant des méthodes statistiques à chaque couche pour trouver les meilleures fonctionnalités et les transmettre à travers le système. (Velluz et Grosjean, 1965). En utilisant la technique GMDH, Ivakhnenko a pu créer en 1971 un réseau profond à 8 couches, et il a également démontré avec succès l'efficacité du processus d'apprentissage dans un système d'identification informatique.(Ivakhnenko, 1971)

Innovateur reconnu dans les réseaux de neurones, Kunihiko Fukushima est mieux connu pour la création du Neocognitron, un réseau neuronal artificiel multicouche hiérarchisé proposé par Fukushima en 1979. Il a été utilisé pour la reconnaissance de caractères manuscrits et d'autres tâches de reconnaissance de formes, et a servi d'inspiration pour les réseaux de neurones convolutionnels.(Barber et al., 1979)

Son travail qui a été fortement influencé par Hubel et Wiesel a conduit au développement des premiers réseaux de neurones convolutifs, qui sont basés sur l'organisation du cortex visuel trouvée chez les animaux. Ce sont des variantes de perceptrons multicouches conçues pour utiliser des quantités minimales de prétraitement.

En 1982, John Hopfield a créé et popularisé le système qui porte désormais son nom. Le réseau Hopfield est un réseau de neurones récurrent qui sert de système de mémoire adressable au contenu ; il reste un outil de mise en œuvre populaire pour le Deep Learning au 21^{e} siècle. (Hopfield, 1982).

Le neuroscientifique computationnel Terry Sejnowski a utilisé sa compréhension du processus d'apprentissage pour créer le NETtalk en 1985. Le programme a appris à prononcer les mots anglais de la même manière qu'un enfant et a pu s'améliorer avec le temps tout en convertissant le texte en parole.(Hinton, 1985).

Dans un article publié en 1986 intitulé : *Learning Representations by Back-propagating Errors*, par David Rumelhart, Geoffrey Hinton et Ronald J. Williams décrivent plus en détail le processus de rétropropagation. Ils ont montré comment cela pouvait considérablement améliorer les réseaux de neurones existants pour de nombreuses tâches telles que la reconnaissance de forme, la prédiction de mots, etc… (Rumelhart, Hinton, et Williams, 1986).

Malgré quelques revers après ce succès initial, Geoffrey Hinton a poursuivi ses recherches pour atteindre de nouveaux niveaux de succès. Il est considéré par beaucoup dans le domaine comme le parrain du Deep Learning.

Yann LeCun, une autre rock star de l'univers de l'intelligence artificiel et du Deep Learning a combiné des réseaux de neurones convolutifs avec des théories de rétropropagation récentes pour lire des chiffres manuscrits en 1989. Son système a finalement été utilisé pour lire les chèques manuscrits et les codes postaux par la NCR corporation (National Cash Register) et d'autres sociétés, traitant de 10 à 20% des chèques encaissés aux États-Unis à la fin des années 90 et au début des années 2000.(LeCun et al.,1989)

Christopher John Cornish Hellaby Watkins a présenté sa thèse de doctorat au King's College en 1989 intitulée : *Learning from Delayed Rewards*. Dans ce document, il a introduit le concept du Q-Learning, qui améliore considérablement l'aspect pratique et la faisabilité de l'apprentissage par renforcement dans les machines. Ce nouvel algorithme a suggéré qu'il était possible d'apprendre directement un contrôle optimal sans modéliser les probabilités de transition ou les récompenses attendues du processus de décision de Markov. (Watkins, 1989)

En 1993, l'informaticien allemand Jürgen Schmidhuber a résolu une tâche *d'apprentissage très profond* nécessitant plus de 1000 couches dans le réseau neuronal récurrent. Ce fut un énorme bond en avant dans la complexité et la capacité des réseaux de neurones.(Kolboom et Weisenfeld, 1993)

Les machines à vecteurs de support ou séparateurs à vaste marge (SVM : *Support Vector Machine*) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Ces techniques existent depuis les années 1960, peaufinées et affinées par de nombreux auteurs au fil des décennies. Le modèle standard actuel a été conçu par Corinna Cortes et Vladimir Vapnik en 1993 et présenté en 1995. Un SVM est essentiellement un système de reconnaissance et de mapping data, et peut être utilisé dans la catégorisation de texte, la reconnaissance de caractères manuscrits et la classification d'images en ce qui concerne le Machine Learning et le Deep Learning. (Drucker, 1994)

La mémoire à court terme à long terme (LSTM) est une architecture de réseau neuronal récurrent artificiel proposé par Jürgen Schmidhuber et Sepp Hochreiter en 1997. Ils améliorent à la fois l'efficacité et l'aspect pratique des réseaux de neurones récurrents en éliminant le problème de dépendance à long terme. Les réseaux LSTM peuvent *mémoriser* ces informations pendant une période plus longue. Raffinés au fil du temps, les réseaux LSTM sont largement utilisés dans les cercles du Deep Learning.(Hochreiter et Schmidhuber, 1997)

Yann LeCun a joué un rôle déterminant dans un autre progrès dans le domaine du Deep Learning lorsqu'il a publié son article: *Gradient-Based Learning Applied to Document Recognition* en 1998. L'algorithme de descente de gradient stochastique combiné à l'algorithme de rétropropagation est l'approche préférée et de plus en plus efficace du Deep Learning.(LeCun et al., 1998).

Professeur et chef du laboratoire d'intelligence artificielle à l'Université de Stanford, Fei-Fei Li a lancé ImageNet en 2009. Il s'agit d'une base de données très vaste et gratuite de plus de 14 millions d'images disponibles pour les chercheurs, les enseignants et les étudiants. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) est une compétition annuelle où des équipes de recherche évaluent leurs algorithmes de traitement d'images sur le jeu de données d'ImageNet .Les années 2010 ont vu des progrès spectaculaires dans le domaine du traitement d'images. En 2011, les plus faibles taux d'erreur de classification de la compétition ILSVRC étaient d'environ 25 %. En 2012, la révolution du Deep Learning a permis de faire baisser subitement ce record à 16 %.(Deng et al., 2009)

Entre 2011 et 2012, Alex Krizhevsky a remporté plusieurs concours internationaux de Machine Learning et Deep Learning avec sa création AlexNet, un réseau de neurones convolutionnels. AlexNet s'est enrichi et amélioré sur LeNet5 construit par Yann LeCun des années plus tôt. Il ne contenait initialement que huit couches - cinq convolutionnelles suivies de trois couches entièrement connectées, et il a renforcé la vitesse et le décrochage (ou l'abandon) à l'aide d'unités linéaires rectifiées. Son succès a donné le coup d'envoi à une renaissance du réseau neuronal convolutionnel dans la communauté du Deep Learning. (Krizhevsky Sutskever et Hinton, 2012)

Remarque : Le décrochage, ou abandon, est une technique de régularisation pour réduire le surajustement dans les réseaux de neurones.

Développé et publié en 2014, le système du Deep Learning du géant des médias sociaux, surnommé DeepFace (DeepFace est un système de reconnaissance faciale d'apprentissage profond créé par un groupe de recherche de

Facebook) utilise des réseaux de neurones pour identifier les visages avec une précision de 97,35%. C'est une amélioration de 27% par rapport aux efforts précédents, et un chiffre qui rivalise avec celui des humains (qui serait de 97,5%). Google Photos utilise un programme similaire. (Baron, 2014).

Les Réseaux Antagonistes Génératifs (GAN pour Generative Adversarial Networks) sont une classe d'algorithmes d'apprentissage non-supervisé. Ces algorithmes ont été introduits par une équipe de chercheurs dirigée par Ian Goodfellow en 2014. Ce type de réseau est l'idée la plus intéressante des dix dernières années dans le domaine de l'apprentissage automatique. (Goodfellow et all, 2014)

A partir de 2016, la Cray Incorporated une société qui conçoit et fabrique des superordinateurs, ainsi que de nombreuses autres entreprises comme elle, sont désormais en mesure d'offrir des machines puissantes et des produits et des solutions du Deep Learning. En utilisant le Neural-Network Software de Microsoft sur ses superordinateurs XC50 avec 1000 unités de traitement graphique Nvidia Tesla P100, on peut effectuer des tâches du Deep Learning sur les données en une fraction de temps alors qu'ils prenaient des heures auparavant. (Ismail et al. , 2016)

Aujourd'hui, le Deep Learning est présent dans nos vies d'une manière que nous ne pouvons même pas considérer : la reconnaissance vocale et d'image de Google, les moteurs de recommandation Netflix et Amazon, Siri d'Apple, les réponses automatiques par e-mail et texte, les chatbots, etc… Avec les données qui nous entourent, il y a plus d'information pour que ces programmes puissent être analysés et améliorés. (import.io)

I.3 Réseaux de Neurones

Les Réseaux de Neurones sont un moyen populaire de mettre en œuvre l'intelligence de la machine. L'idée est qu'ils se comportent comme les neurones d'un cerveau. Le terme de « réseau de neurones » suggère un lien fort avec la biologie. Ce lien existe : les méthodes mathématiques qui seront décrites par la suite, ont été appliquées avec succès à la modélisation des systèmes nerveux vivants. Néanmoins, le terme est plus métaphorique que scientifique : si le lien avec la biologie a constitué une motivation majeure des pionniers du domaine, les réels développements des réseaux de neurones sont de nature purement mathématique et statistique ; leurs applications se situent dans des domaines qui n'ont généralement aucun rapport avec la neurobiologie. (Dreyfus, 2002).

Un modèle linéaire statique est de la forme :

$$g(x, w) = \sum_{i=1}^{p} w_i f_i(x) \qquad 1$$

Où le vecteur w est le vecteur des paramètres du modèle, et où les fonctions $f\ i\ (x)$ sont des fonctions non paramétrées, ou à paramètres fixés et connus, des variables x. Les réseaux de neurones entrent dans la catégorie des modèles non linéaires en leurs paramètres. La forme la plus courante de réseau de neurones statique est une extension simple de la relation précédente :

$$g(x, w) = \sum_{i=1}^{p} w_i f_i(x, w') \qquad 2$$

où les fonctions $f_i(X, W')$, appelées « neurones », sont des fonctions paramétrées.

- **Définition**

Un neurone est une fonction non linéaire, paramétrée, à valeurs bornées. Suivant en cela l'usage, on utilisera fréquemment, par abus de langage, le terme de « neurone linéaire » pour désigner une fonction paramétrée linéaire ou affine (qui n'est donc pas bornée).

Les variables sur lesquelles opère le neurone sont souvent désignées sous le terme d'entrées du neurone, et la valeur de la fonction sous le terme de sortie. Il est commode de représenter graphiquement un neurone comme indiqué sur la figure 02. Cette représentation est le reflet de l'inspiration biologique qui a été à l'origine de la première vague d'intérêt pour les neurones formels, dans les années 1940 à 1970. (McCulloch et Pitts, 1943) (Minsky et Papert, 1969)

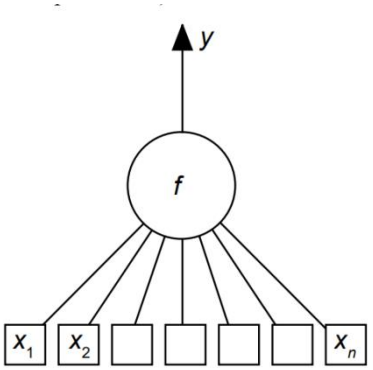


Fig. 02 : - Un neurone réalise une fonction non linéaire paramétrée bornée $y = f(x, w)$ où les composantes du vecteur x sont les variables et celles du vecteur w sont les paramètres

La fonction f peut être paramétrée de manière quelconque. Deux types de paramétrage sont fréquemment utilisés :

- les paramètres sont attachés aux variables du neurone : la sortie du neurone est une fonction non linéaire d'une combinaison des variables$\{x_i\}$, pondérées par les paramètres $\{w_i\}$, qui sont alors souvent désignés sous le nom de « poids » ou, en raison de l'inspiration biologique des réseaux de neurones, « poids synaptiques ». Conformément à l'usage (également inspiré par la biologie), cette combinaison linéaire sera appelée « potentiel » dans tout cet ouvrage. Le potentiel v le plus fréquemment utilisé est la somme pondérée, à laquelle s'ajoute un terme constant ou « biais » :

$$v = w_0 + \sum_{i=1}^{n} w_i x_i \qquad 3$$

La fonction f est appelée fonction d'activation. Pour des raisons qui seront exposées plus loin, il est recommandé d'utiliser pour f une fonction « sigmoïde » (c'est-à-dire une fonction en forme de « s ») symétrique par rapport à l'origine, telle que la tangente hyperbolique ou la fonction Arctangente. Ainsi, dans la très grande $m_0$ajorité des applications qui seront décrites dans ce chapitre, la sortie d'un neurone a pour équation :

$$y = th[w_0 + \sum_{i=1}^{n} w_i x_i] \qquad 4$$

Le biais w_0 peut être considéré comme le produit du paramètre w_0 par la constante 1, de sorte qu'il est commode d'introduire une variable égale à 1 dans le vecteur des variables. La relation précédente peut alors s'écrire :

$$y = th(w.x) \qquad 5$$

où le symbole $\cdot$ désigne le produit scalaire de deux vecteurs ;

- les paramètres sont attachés à la non-linéarité du neurone : ils interviennent directement dans la fonction f ; cette dernière peut être une fonction radiale ou RBF (en anglais Radial Basis Function), ou encore une ondelette ; la première tire son origine de la théorie de l'approximation (Powell, 1987), la seconde de la théorie du signal (Mallat, 1989).

  Par exemple, la sortie d'un neurone RBF à non-linéarité gaussienne a pour équation :

$$y = \exp\left[-\frac{\sum_{i=1}^{n}(x_i - w_i)^2}{2w_{n+1}^2}\right] \qquad 6$$

où les paramètresw_i, i = 1 à n sont les coordonnées du centre de la gaussienne, et w_{n+1} est son écart-type. Dans les compléments théoriques et algorithmiques, en fin de chapitre, d'autres exemples de neurones sont présentés. La différence pratique essentielle entre les deux types de neurones qui viennent d'être décrits est la suivante : les neurones tels que les RBF ou les ondelettes ont des non-linéarités locales, qui tendent vers

zéro dans toutes les directions de l'espace des variables ; leur zone d'influence est donc limitée dans l'espace, ce qui n'est pas le cas des neurones à fonction d'activation sigmoïde.

- **Réseaux de neurones statiques ou réseaux non bouclés**

Définition :

Un réseau de neurones non bouclé réalise une ou plusieurs fonctions de ses entrées par composition des fonctions réalisées par chacun des neurones.

Un réseau de neurones non bouclé peut donc être imaginé comme un ensemble de neurones connectés entre eux, l'information circulant des entrées vers les sorties sans retour en arrière.

On peut alors représenter le réseau par un graphe acyclique dont les nœuds sont les neurones et les arêtes les «connexions entre ceux-ci. La représentation de la topologie d'un réseau par un graphe est très utile, notamment pour les réseaux bouclés. Les neurones qui effectuent le dernier calcul de la composition de fonctions sont les neurones de sortie ; ceux qui effectuent des calculs intermédiaires sont les neurones cachés fig. 03.

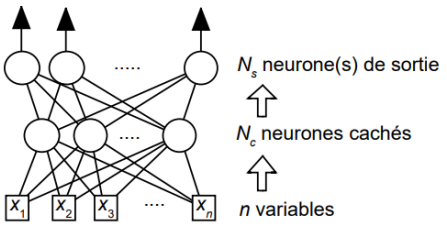


Fig. 03 : - Un réseau de neurones à n variables, une couche de N_c neurones cachés et N_s neurones de sortie.

Réseau à couches :

Ce réseau réalise N_c fonctions algébriques des n variables du réseau ; chacune des sorties est une fonction, réalisée par le neurone de sortie correspondant, des fonctions non linéaires réalisées par les neurones cachés. Le temps ne joue aucun rôle fonctionnel dans un réseau de neurones non bouclé, si les variables sont indépendantes du temps, les sorties le sont également. Le temps nécessaire pour le calcul de la fonction réalisée par chaque neurone est négligeable et, fonctionnellement, on peut considérer ce calcul comme instantané. Pour cette raison, les réseaux non bouclés sont souvent appelés *réseaux statiques*, par opposition aux réseaux bouclés ou *dynamiques*.

Les réseaux de neurones non bouclés à couches, dont les neurones cachés ont une fonction d'activation sigmoïde, sont souvent appelés *Perceptrons multicouche* (ou MLP pour Multi-Layer Perceptron). Pour les réseaux à une couche cachée de sigmoïdes et un neurone de sortie linéaire, l'extension la plus naturelle des modèles linéaires de la forme :

$$g(x, w) = \sum_{i=1}^{p} w_i f_i(x) \qquad 7$$

est une combinaison linéaire de fonctions paramétrées :

$$g(x, w) = \sum_{i=1}^{p} w_i f_i(x, w') \qquad 8$$

c'est la forme la plus utile de modèle neuronal : une combinaison linéaire de fonctions non linéaires paramétrées des variables. Le modèle représenté sur la figure 04 a pour expression:

$$g(x, w) = \sum_{i=1}^{N_c} \left[w_{N_{c+1,i}} th\left(\sum_{j=1}^{n} w_{ij} x_j + w_{i0} \right) \right] + w_{N_{c+1,0}} = w_2 . f(W_1 x) \qquad 9$$

où $\boldsymbol{x}$ est le vecteur des variables (de dimension $\boldsymbol{n+1}$), $\boldsymbol{w2}$ est le vecteur des paramètres de la deuxième couche de connexions (de dimension $\boldsymbol{Nc+1}$), $\boldsymbol{W1}$ est la matrice des connexions de la première couche (de dimension $(\boldsymbol{Nc+1, n+1})$, $\boldsymbol{et}\ \boldsymbol{f}(.)$ est le vecteur (de dimension $\boldsymbol{Nc+1}$) constitué du biais et des fonctions réalisées par les neurones cachés : $\boldsymbol{f0=1}$,

$$f_i = \mathrm{th}\left(\sum_{j=0}^{n} w_{ij}x_j\right) \qquad 10$$

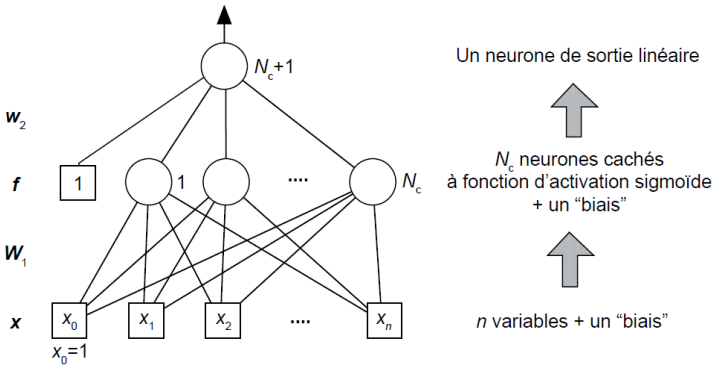


Fig. 04 : - Un réseau de neurones à n variables, un biais, une couche de Nc neurones cachés à fonction d'activation sigmoïde et un neurone de sortie linéaire.

Les neurones cachés sont numérotés de 1 à N_c et le neurone de sortie est numéroté N_c+1. Par convention, le paramètre W_{ij} est relatif à la connexion allant du neurone (ou de l'entrée) j vers le neurone i.

Si la relation que l'on cherche à réaliser entre les variables et les sorties présente une importante composante linéaire, il peut être utile d'ajouter, à la structure de réseau à couches qui vient d'être décrite, des termes linéaires, parfois appelés *termes directs*, qui se traduisent, dans la représentation graphique du réseau, par des connexions directes entre les entrées et le neurone de sortie (Fig.05).

Par exemple, pour un réseau dont les fonctions d'activation sont des sigmoïdes, le modèle devient :

$$\begin{aligned} g(x,w) &= \left[\sum_{i=1}^{N_c} w_{N_{c+1,i}}\,\mathrm{th}\left(\sum_{j=1}^{n} w_{ij}x_j + w_{i0}\right)\right] \\ &\quad + w_{N_{c+1,0}} + \sum_{k=1}^{n} w_{N_{c+1,k}}x_k \qquad 11 \\ &= w_2.f(W_1x)+w.x' \end{aligned}$$

Où W_3 est un vecteur de dimension n et x^1 est le vecteur de composantes$\{X_1, \dots, X_n\}$, c'est-à-dire le vecteur x dépourvu du biais.

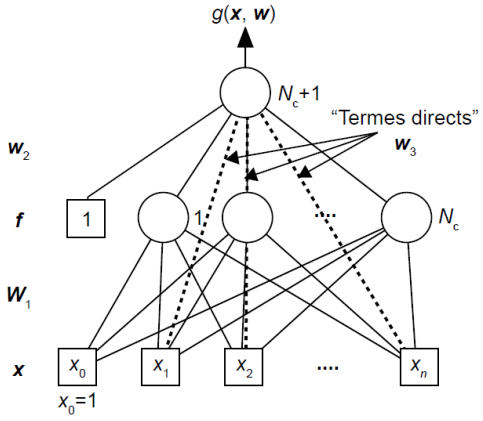


Fig. 05 : - Représentation graphique d'un réseau de neurones à couches comportant des termes directs.

Réseaux de RBF (fonctions radiales de base) ou d'ondelettes

Dans ce cas, comme indiqué plus haut, les paramètres relatifs aux RBF sont attachés à la non-linéarité elle-même ; en revanche, le neurone de sortie (linéaire) réalise une somme pondérée des sorties des neurones cachés. La sortie du réseau a donc pour expression (pour des fonctions radiales gaussiennes) :

$$g(x,w) = \sum_{i=1}^{N_c} w_{N_{c+1,i}} \exp\left(-\frac{\sum_{j=1}^{n}(x_j - w_{ij})^2}{2w_i^2}\right) \qquad 12$$

où x est le vecteur des entrées du réseau (de dimension n) et w est le vecteur des paramètres du réseau (de dimension $(n+2)\ N_b+1$) (Broomhead et Lowe, 1988) (Moody et Darken, 1989). les neurones cachés sont numérotés de 1 à N_c , et le neurone de sortie porte le numéroN_c+1.

Les connexions de la première couche ont toutes des paramètres égaux à 1. Dans ces réseaux, la sortie est une fonction linéaire des paramètres de la dernière couche de connexions, et elle est une fonction non linéaire des paramètres des gaussiennes. Les réseaux d'ondelettes ont exactement la même structure, l'équation de la gaussienne étant remplacée par celle d'une ondelette multidimensionnelle. Les paramètres attachés à la non-linéarité sont alors les centres et les dilatations des ondelettes (Blanche-Benveniste, 1994) (Oussar et Dreyfus, 2000) .

- **Réseaux de neurones dynamiques ou réseaux bouclés (ou récurrents)**

L'architecture la plus générale, pour un réseau de neurones, est celle des « réseaux bouclés », dont le graphe des connexions est cyclique : lorsque l'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est désigné sous le terme de « cycle »). La sortie d'un neurone du réseau peut donc être fonction d'elle-même ; ceci n'est évidemment concevable que si la notion de temps est explicitement prise en considération.

Définition

Un réseau de neurones bouclé à temps discret réalise une (ou plusieurs) équation(s) aux différences non linéaires, par composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions.

Propriété

Tout cycle du graphe des connexions d'un réseau de neurones bouclé doit comprendre au moins une connexion de retard non nul. La figure 06 présente un exemple de réseau de neurones bouclé. Les chiffres dans les carrés indiquent le retard attaché à chaque connexion, exprimé en multiple de l'unité de temps (ou période d'échantillonnage) T. Ce réseau contient un cycle qui part du neurone 3 et revient à celui-ci en passant par le neurone 4 ; la connexion de 4 vers 3 ayant un retard non nul, ce réseau est causal.

Explications

A l'instant kT : le neurone 3 calcule$y_3(kT)$ en fonction de $y_4[(k-1)T]$, $u_1(kT)$, $u_2[(k-1)T]$ où k est un entier positif et $yi\ (kT)$ désigne la sortie du neurone i à l'instant(kT). Le neurone 4 calcule $y_4(kT)$ en fonction de $y_3(kT)$ et $u_2(kT)$. Le neurone 5 calcule la sortie du réseau de neurones, $g(kT)$en fonction de$y_3(kT)$, $y_4[(k-1)T]$, et $u_1(kT)$.. Les équations récurrentes qui gouvernent le réseau sont donc :

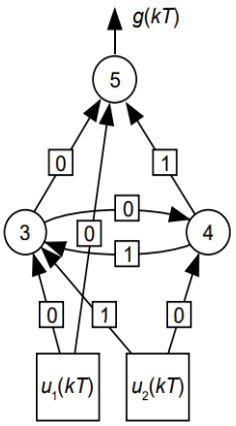


Fig. 06 : - Un réseau de neurones bouclé à deux variables

$$y_3(k) = f_3[y_4(k-1)u_1(k), u_2(k-1)] \qquad 13$$

$$y_4(k) = f_4[y_3(k), u_2(k)] \qquad 14$$

$$g(k) = f_5[y-(k), y_4(k-1), u_1(k)] \qquad 15$$

où, pour alléger les notations, la période d'échantillonnage T a été omise. f_3, f_4, f_5 sont les fonctions non linéaires réalisées par les neurones 3, 4 et 5 respectivement.

Forme canonique des réseaux de neurones bouclés

Dans la mesure où les réseaux de neurones bouclés réalisent des équations récurrentes non linéaires, il est utile d'examiner les liens entre ces modèles non linéaires et les modèles dynamiques linéaires, utilisés notamment en automatique des systèmes linéaires.

La description la plus générale d'un système linéaire est la description d'état :

$$x(k) = Ax(k-1) + Bu(k-1) \qquad 16$$

$$g(k) = Cx(k) + Du(k) \quad 17$$

où $x(k)$ est le vecteur des variables d'état à l'instant (discret) kT, $u(k)$ est le vecteur des variables de commande à l'instant kT, $g(k)$ est le vecteur des prévisions du modèle à l'instant kT, $et\ A, B, C, D$ sont des matrices. Rappelons que les variables d'état sont un ensemble de variables, en nombre minimal, telles que l'on peut calculer leurs valeurs à l'instant$(k+1)T$ si l'on connaît leurs valeurs initiales et si l'on connaît les valeurs des variables de commande à tout instant compris entre$0\ et\ kT$. Le nombre de variables d'état est appelé ordre du système. De manière analogue, on définit la forme canonique d'un système non linéaire à temps discret par les équations suivantes :

$$x(k) = \Phi(x(k-1), u(k-1)) \quad 18$$

$$g(k) = \Psi(x(k-1), u(k-1)) \quad 19$$

Où Φ et Ψ sont des fonctions non linéaires (des réseaux de neurones, par exemple), et x est le vecteur des variables d'état. Là encore, les variables d'état sont un ensemble de variables, en nombre minimal, permettant de décrire complètement le système à l'instant k si l'on connaît leurs valeurs initiales et si l'on connaît les valeurs des variables de commande à tout instant compris entre$0\ et\ k-1$. Tout réseau de neurones peut être mis sous une forme canonique (Fig.07) où le symbole q^{-1} représente un retard d'une unité de temps.

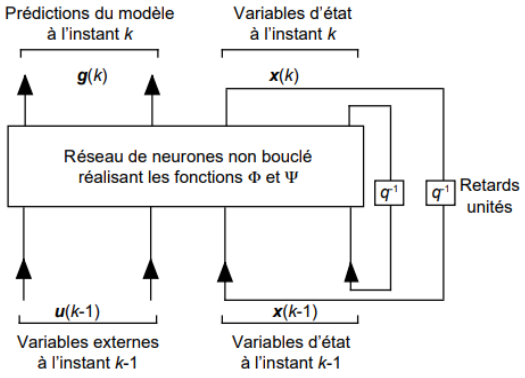


Fig.07 : - Forme canonique d'un réseau de neurones bouclé.

Propriété

Tout réseau de neurones bouclé, aussi complexe soit-il, peut être ramené à une forme canonique, comportant un réseau de neurones non bouclé dont certaines sorties (les variables d'état) sont ramenées aux entrées par des bouclages de retard unité (Nerrand et al., 1993). Par exemple, le réseau de neurones représenté sur la figure 06 peut être mis sous la forme canonique indiquée sur la figure 08. Ce réseau possède une seule variable d'état (il est donc du 1er ordre), qui est la sortie du neurone 3. Dans cet exemple, ce neurone est un neurone caché, mais un neurone de sortie peut être un neurone d'état.

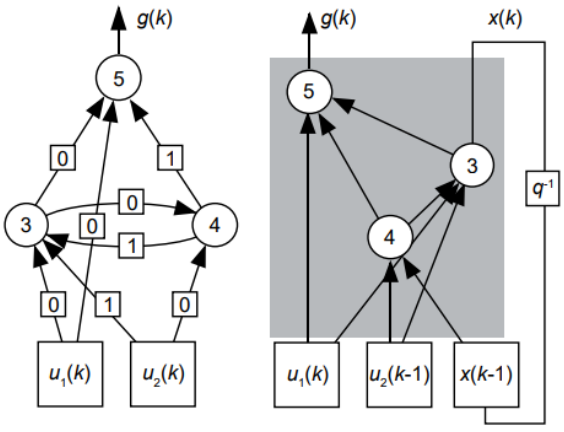


Fig. 08 : - Forme canonique à droite du réseau représenté à gauche. Ce réseau possède une variable d'état $x(kT)$ à la sortie du neurone 3.

Explications

À l'instant kT, le neurone 4 a pour variables $u2[(k-1)T]$ et $x[(k-1)T] = y3[(k-1)T])$: il calcule donc$y4[(k-1)T]$; comme dans la forme non canonique, le neurone 3 a pour variables $u1(kT), u2[(k-1)T], y4[(k-1)T]$: il calcule donc $y3(kT)$; le neurone 5 a pour variables $y3(kT), u1(kT)$ et $y4[(k-1)T]$: il calcule donc sa sortie, qui est la sortie du réseau de neurones, $g(kT)$.

Les deux réseaux sont donc bien équivalents fonctionnellement. On peut aussi montrer le résultat en comparant les équations qui régissent les deux réseaux, posant alors :

$$z_3 = f_3(z_4, u_2(k-1)) \quad 20$$
$$z_4 = f_4(z_3(k-1), u_2(k-1)) \quad 21$$

le modèle sous forme canonique s'écrit :

$$g(k) = (f_5 z_3 z_4 u_1(k)) \quad 22$$

Ces équations sont bien identiques à celles de la forme non canonique :

$$y_3(k) = f_3[y_4(k-1)u_1(k), u_2(k-1)] \quad 23$$
$$y_4(k) = f_4[y_3(k), u_2(k)] \quad 24$$
$$g(k) = f_5[y_3(k), y_4(k-1), u_1(k)] \quad 25$$
$$\text{En identifiant } z_3 = y_3(k) \text{et } z_4 = y_4(k-1).$$

I.4 Deep Learning et Données

Les systèmes d'apprentissage profond fonctionnent sur des données. Les données peuvent être organisées de plusieurs façons. Le formulaire de tableau pour les images par exemple implique une structure pour les données. Le même nombre de points peut être organisé en un seul vecteur. Les réseaux de neurones convolutifs sont souvent utilisés pour les données structurées d'images. (Paluszek et Thomas, 2020 b)

On pourrait aussi avoir un vecteur dont nous souhaitons apprendre une séquence temporelle. Dans ce cas, si chaque colonne est un échantillon de temps, nous pourrions avoir par exemple à examiner une séquence d'échantillons en cours et déterminer si un ensemble de *k* échantillons correspond à une séquence prédéterminée. Pour ce problème simple, le réseau neuronal apprendrait la séquence et serait ensuite alimenté en ensembles de plusieurs échantillons. (Paluszek et Thomas,b 2020)

Cependant,nous devons également distinguer ce que nous entendons par correspondance. Si tous nos chiffres sont exacts, le problème est relative ment simple. Dans les systèmes réels, les mesures sont souvent bruyantes. Dans ces cas, nous voulons correspondre avec une certaine probabilité. Cela conduit au concept de réseaux neuronaux statistiques. (Paluszek et Thomas,b 2020)

I.5 Types de Deep Learning

Il existe de nombreux types de réseaux d'apprentissage profond. De nouveaux types sont toujours en cours de développement. Dans ce paragraphe nous décrivons brièvement certains des principaux types.(Paluszek et Thomas, 2020.a)

1.5.1. Multilayer Neural Network

Le Perceptron Multicouche (MLP) est un type de réseau neuronal artificiel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau à propagation directe (*feedforward*).

Chaque couche est constituée d'un nombre variable de neurones, les neurones de la dernière couche (dite de sortie) étant les sorties du système global. Le perceptron a été inventé en 1957 par Frank Rosenblatt. (Rosenblatt, 1957)

Le perceptron est organisé en trois parties :

- La couche d'entrée (input layer) correspond à un ensemble de neurones qui portent le signal d'entrée.
- La couche cachée (hidden layer) ou plus souvent plusieurs couches cachées (couche cachée 1, couche cachée 2, …). Il s'agit du cœur de notre perceptron, là où les relations entre les variables vont être mises en exergue.
- La couche de sortie (output layer) : cette couche représente le résultat final de notre réseau. (Gülen, 2018)

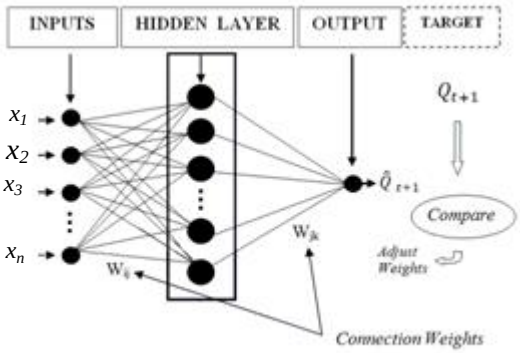


Fig. 09 : - Structure d'un perceptron multicouche

1.5.2. Convolutional Neural Networks (CNN)

Les réseaux de neurones convolutionnels (CNN) est une séquence de couches où chaque couche transforme un volume d'activations en un autre par une fonction différentiable. Les réseaux de neurones convolutifs visent à limiter le nombre d'entrées tout en conservant la forte corrélation spatialement locale des images naturelles (Beghdad et Ouserir, 2018). Les CNN ont les traits distinctifs suivants :

1. ***Connectivité locale*** : grâce au champ récepteur qui limite le nombre d'entrées du neurone ;(Gülen, 2018)
2. ***Poids partagés*** : dans les réseaux de neurones convolutifs, les paramètres de filtrage d'un neurone sont identiques pour tous les autres neurones d'un même noyau ;
3. ***Invariance à la translation*** : comme tous les neurones d'un même noyau (filtre) sont identiques, le motif détecté par ce noyau est indépendant de la localisation spatiale dans l'image.

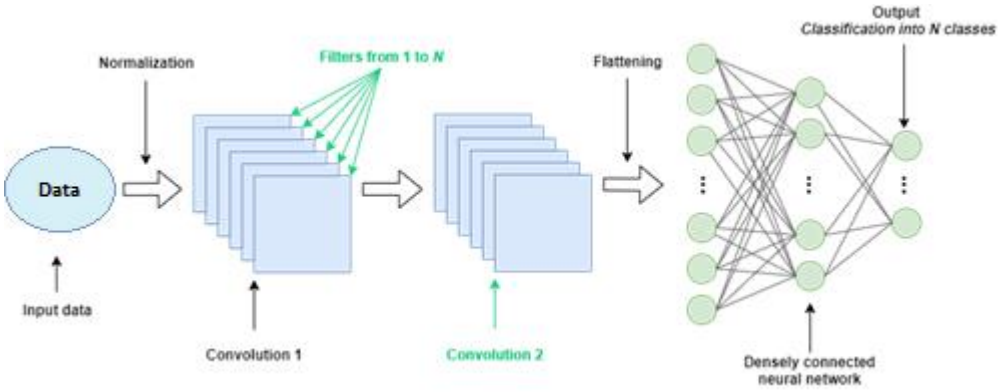


Fig. 10 : - L'architecture d'un réseau de neurones convolutifs

Une architecture de réseau de neurones convolutifs est formée par un empilement de couches de traitement :

- la couche de convolution (CONV) qui traite les données d'un champ récepteur ;
- la couche de pooling (POOL), qui permet de compresser l'information (souvent par sous-échantillonnage) ;
- la couche de correction (ReLU), souvent appelée par abus « ReLU » en référence à la fonction d'activation (Unité de rectification linéaire) ;
- la couche « entièrement connectée » (FC), qui est une couche de type perceptron ;
- la couche de perte (LOSS).

1.5.3. Recurrent Neural Network (RNN)

Un réseau de neurones récurrents est un réseau de neurones artificiels présentant des connexions récurrentes. Un réseau de neurones récurrents est constitué d'unités (neurones) interconnectées interagissant

non-linéairement et pour lequel il existe au moins un cycle dans la structure. Les unités sont reliées par des arcs (synapses) qui possèdent un poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées.

Les réseaux de neurones récurrents sont utilisés pour les problèmes dépendant du temps. Ils combinent les données du dernier pas de temps avec les données de la couche cachée ou intermédiaire, pour produire une représentation du pas de temps actuel.

Les architectures des réseaux récurrents varient d'entièrement connectées à partiellement connectés, y compris les réseaux multicouches à couches d'entrée et de sortie distinctes.

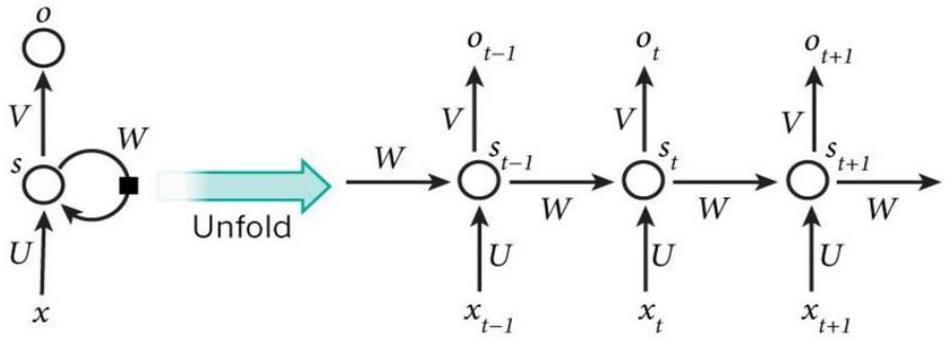


Fig. 11 : - Un réseau de neurones récurrent(d'apres Britz ,2015)

Les techniques d'entraînement du réseau sont les mêmes que pour les réseaux classiques (rétropropagation du gradient), néanmoins les réseaux de neurones récurrents se heurtent au problème de disparition du gradient pour apprendre à mémoriser des évènements passés. Des architectures particulières répondent à ce dernier problème, on peut citer en particulier les réseaux Long short-term memory.

1.5.4. Long Short-Term Memory Networks (LSTMs)

Un réseau Long short-term memory (LSTM), en français réseau récurrent à mémoire court et long terme ou plus explicitement réseau de neurones récurrents à mémoire court-terme et long terme, est l'architecture de réseau de neurones récurrents la plus utilisée en pratique qui permet de répondre au problème de disparition de gradient.

Les réseaux de mémoire à court terme visent à surmonter le problème des gradients de disparition en utilisant les portes pour conserver sélectivement les informations pertinentes et oublier les informations qui ne le sont pas. Une sensibilité plus faible à l'écart temporel rend les réseaux LSTM meilleurs pour l'analyse des données séquentielles que les RNN simples.(MathWorks)

L'architecture d'un bloc LSTM est illustrée ci-dessous. Un bloc LSTM a généralement une cellule de mémoire, une porte d'entrée, une porte de sortie et une porte d'oubli en plus de l'état caché dans les RNN traditionnelss.(MathWorks)

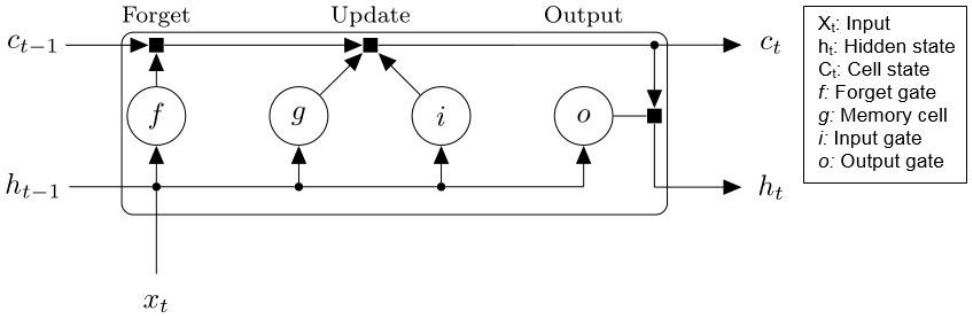


Fig. 12: - Architecture d'un bloc simple LSTM.(MathWorks)

L'idée associée au LSTM est que chaque unité computationnelle est liée non seulement à un état h caché mais également à un état c de la cellule qui joue le rôle de mémoire. Le passage de c_{t-1} à c_t se fait par transfert à gain constant et égal à 1.

Les pondérations et les biais de la porte d'entrée contrôlent la mesure dans laquelle une nouvelle valeur s'écoule dans la cellule. De même, les pondérations et les biais de la porte d'oubli et de la porte de sortie contrôlent la mesure dans laquelle une valeur reste dans la cellule et la mesure dans laquelle la valeur dans la cellule est utilisée pour calculer l'activation en sortie du bloc LSTM, respectivement.(MathWorks)

Recursive Neural Network

Un réseau de neurones récursif est une sorte de réseau de neurones profond créé en appliquant le même ensemble de poids de manière récursive sur une entrée structurée, pour produire une prédiction structurée sur des structures d'entrée de taille variable, ou une prédiction scalaire sur celle-ci, en parcourant une structure donnée dans l'ordre topologique
Ces réseaux ont déjà été appliqués avec succès pour modéliser la compositionnalité en langage naturel à l'aide de représentations structurelles basées sur un arbre d'analyse.(Irsoy et Cardie, 2014)

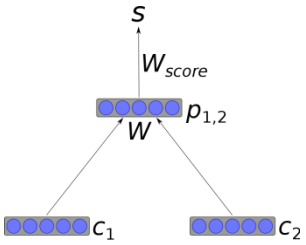


Fig. 13 : - Architecture d'un réseau de neurones récursif simple

1.5.5. Temporal Convolutional Machines (TCM)

Le TCM est une architecture convolutive conçue pour apprendre des séquences temporelles (Lockett et Miikkulainen, 2009). Les TCM sont particulièrement utiles pour la modélisation statistique des séquences temporelles. La modélisation statistique est appropriée lorsque les données entrantes sont bruyantes. Lockett et Miikkulainen, 2009)

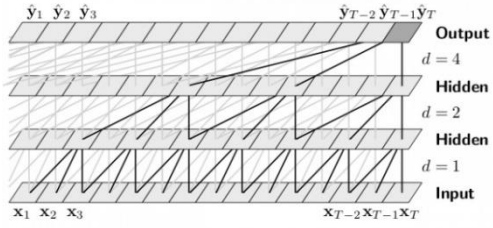


Fig. 14 : - Structure d'un réseau convolutif temporel.

1.5.6. Stacked Autoencoders

Un auto-encodeur, ou auto-associateur est un réseau de neurones artificiels utilisé pour l'apprentissage non supervisé de caractéristiques discriminantes.

L'objectif d'un auto-encodeur est d'apprendre une représentation (encodage) d'un ensemble de données, généralement dans le but de réduire la dimension de cet ensemble. Récemment, le concept d'auto-encodeur est devenu plus largement utilisé pour l'apprentissage de modèles génératifs.(Paluszek et Thomas, 2020.b)

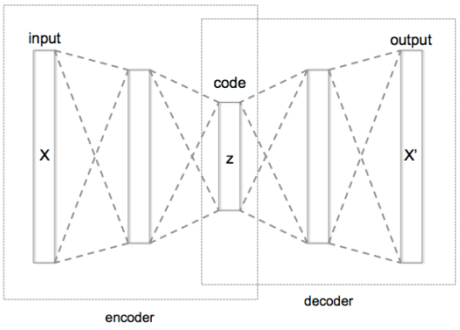


Fig. 15 : -Schéma simplifié d'un réseau neuronal auto-encodeur (Stacked auto-encoder).

1.5.7. Extreme Learning Machine (ELM)

En apprentissage automatique, le terme **extreme learning machine** fait référence à un type de réseau de neurones. Sa spécificité est de n'avoir qu'une seule couche de nœuds cachés, où les poids des entrées de connexion de nœuds cachés sont répartis au hasard et jamais mis à jour. Ces poids entre les nœuds

cachés d'entrée et les sorties sont appris en une seule étape, ce qui revient essentiellement à l'apprentissage d'un modèle linéaire. Le nom "extreme learning machine" (ELM) a été donné à ces modèles par Guang-Bin Huang, mais le principe était déjà connu.(Extreme_learning_machine)

Ces modèles peuvent produire une bonne performance de généralisation et avoir un processus d'apprentissage beaucoup plus rapide que les réseaux formés en utilisant la rétropropagation du gradient.

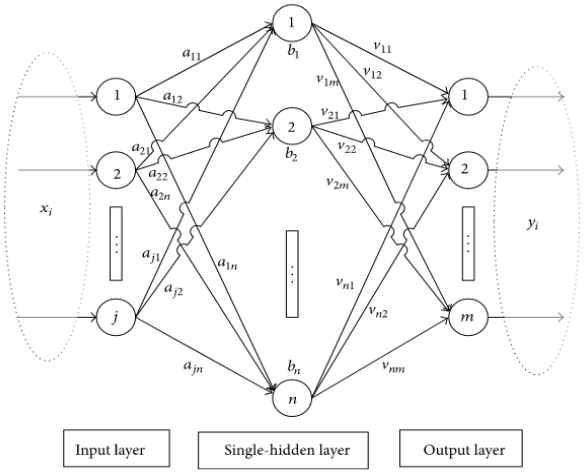


Fig. 16 : -Exemple d'un réseau simple ELM : Réseau neuronal Feedforward à couche cachée unique.

1.5.8. Recursive Deep Learning

Les réseaux neuronaux récursifs sont une sorte de réseau neuronal profond. Ils sont généralement créés après avoir appliqué le même ensemble de pondérations de manière récursive sur les entrées structurées. Cela se produit à chaque nœud pour la même raison. Ils font partie d'une classe d'architecture qui fonctionne sur des entrées structurées, et en particulier, sur des graphes acycliques dirigés. (medium.com (convolutional-recursive-deep-learning-for-3d-object-classification))

Ils ont une structure arborescente profonde. La topologie en forme d'arbre permet des connexions de branchement et une structure hiérarchique. On peut faire la classification des approches de réseaux de neurones récursifs en deux comme ci-dessous :

• **Approche interne** - Cette approche consiste généralement à effectuer une récursivité à l'intérieur du graphique sous-jacent et l'objectif est généralement atteint en avançant lentement autour des bords du graphique.

• **Approche externe** - Cette approche consiste généralement à effectuer une récursion en dehors du graphique sous-jacent et à agréger les informations sur des distances progressivement plus longues dans une direction rectangulaire.

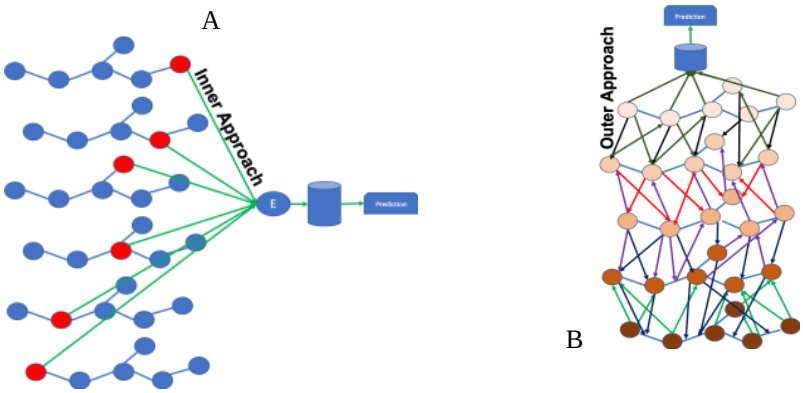


Fig. 17 :-Exemple de deux structures de réseaux neuronaux récursifs :
(A) – Approche interne, (B) – Approche externe

Ces réseaux sont utilisés pour prédire les sorties structurées sur des structures d'entrée de taille variable, parfois une prédiction scalaire en parcourant une structure donnée dans l'ordre topologique. Les réseaux de neurones récursifs répondent non seulement aux entrées, mais également au contexte. Ils traitent chaque entrée de série chronologique séparément.

1.5.9. Generative Deep Learning

Un modèle génératif décrit comment un ensemble de données est généré, en termes de modèle probabiliste. En échantillonnant à partir de ce modèle, nous sommes en mesure de générer de nouvelles données. (Paluszek et Thomas, 2020.b) (Foster, 2019).

Le modèle génératif doit comprendre la distribution à partir de laquelle les données sont obtenues et doit ensuite utiliser cette compréhension pour effectuer la tâche de classification. Les modèles génératifs ont également la capacité de créer des données similaires aux données d'apprentissage qu'ils ont reçues car ils ont appris la distribution à partir de laquelle les données sont fournies (Scisnack.com (26/11/2013)).Pour illustration et pour plus d'information, on renvoi le lecteur au livre de David Foster édité par O'Reilly Madia Inc. en 2019 intitulé : Generative Deep Learning : *Teaching Machines to Paint, Write, Compose, and Play.*

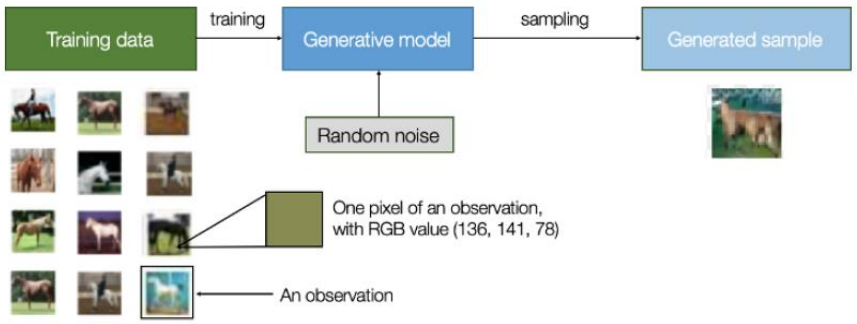


Fig. 18 : - Processus de modélisation générative dans le cas du traitement d'image (D'après Foster 2019)

I.6 Applications du Deep Learning

L'apprentissage profond est utilisé dans de nombreuses applications aujourd'hui. En voici quelques exemples.

I.6.1. Reconnaissance d'images

Il s'agit sans doute de l'utilisation la plus connue et la plus controversée de l'apprentissage en profondeur. Un système d'apprentissage en profondeur est formé avec des photos de gens. Les caméras sont distribuées partout et les images capturées. Le système identifie ensuite les faces et les correspondances individuelles contre sa base de données formée. Même avec des variations d'éclairage, les conditions météorologiques, et les vêtements, le système peut identifier les gens dans les images.

I.6.2. Reconnaissance de la parole

On a rarement un être humain au téléphone. Tu es le premier présenté avec un auditeur robotisé qui peut identifier ce que vous dites, du moins dans contexte limité de ce qu'il attend. Quand un humain écoute un autre humain, l'auditeur ne se contente pas d'enregistrer le discours, il ou elle devine ce que la personne va dire et combler les vides de mots brouillés et confondre grammaire. Les auditeurs robotisés ont quelques-uns dès les mêmes capacités. Un auditeur robotique est l'incarnation du "test de Turing". Vous en avez un que vous pensiez être humain ? Ou d'ailleurs, avez-vous jamais atteint un humain que vous pensiez être un robot ?

I.6.3. Analyse de l'écriture manuscrite

Il y a longtemps, vous obteniez des formulaires dans lesquels vous aviez des boîtes dans lesquelles écrire des chiffres et des lettres. Au début, ils devaient être des capitales de blocs ! Un système robotisé d'écriture

manuscrite pourrait trouver les lettres dans ces boîtes de manière fiable. Des années plus tard, bien qu'il y ait de nombreuses années, le ministère des Postes américain a introduit des systèmes de lecture de code postal. Au début, vous deviez placer le code postal sur une partie spécifique de l'enveloppe. Ce système a évolué pour qu'il puisse trouver des codes postaux n'importe où. Cela a rendu le système zip + 4 vraiment précieux et une forte augmentation de la productivité

I.6.4. Traduction automatique

Traduction automatique: Google traductions fait un assez bon travail car il peut traduire presque toutes les langues du monde. C'est un exemple de système de formation en ligne. Vous voyez que lorsque vous tapez une phrase et que la traduction a une coche en regard de celle-ci, car un être humain a dit qu'il avait raison.

I.6.5. Ciblage

En ciblant nous voulons dire trouver ce que vous **voulez**. C'est peut-être un film, un un vêtement, ou un livre.

Des systèmes d'apprentissage profonds recueillent des informations sur ce que vous aimez et décidez ce que vous seriez le plus intéressé à acheter. Ceci est d'il y a quelques années. Peut-être des danseurs de ballet comme Star Wars! D'autres applications incluent le jeu, la conduite autonome, la médecine et bien d'autres. Juste à propos de toute activité humaine peut être une application de l'apprentissage profond.

I.7. Conclusion

L’utilisation du Deep Learning dans plusieurs domaines techniques et scientifiques constitue une motivation justifiée pour le tester à travers son réseau LSTM pour la prévision en hydrologie. Les séries hydrologiques ont le plus souvent des fréquences de coupures correspondant à de très longues mémoires et un comportement non linéaire, ceci constitue une perspective motivante faisant l’objet de ce travail.

Chapitre 2

Réseaux Long Short-Term Memory (LSTM)

II.1. Introduction

Les réseaux de neurones récurrents (RNN) sont largement utilisés en intelligence artificielle dès lors qu'une notion temporelle intervient dans les données comme c'est le cas des données Hydrologiques.

Un réseau de neurones récurrents est un réseau de neurones très répandu en Deep Learning. Néanmoins, comme pour le perceptron en son temps, le RNN souffre du problème de dissipation du gradient, c'est-à-dire pour "*apprendre*", un RNN utilise la méthode de la descente du gradient afin de mettre à jour les poids entre ses neurones. Cela repose sur la formule suivante :

$$\omega_{k+1} = \omega_k - \alpha_k \cdot F_\omega \tag{26}$$

Avec :

- ω un poids du réseau
- α la vitesse d'apprentissage du réseau
- F_ω le gradient du réseau par rapport au poids ω

Le problème est que la mise à jour des poids se fait de droite à gauche. A mesure que l'on avance vers la gauche, le produit $\alpha_k \cdot F_\omega$ devient très petit et les poids des premières couches de neurones ne sont quasiment pas modifiés. Ainsi, ces couches n'apprennent strictement rien, par conséquent, le RNN peut *facilement oublier* des données un petit peu anciennes lors de la phase d'apprentissage : *sa mémoire est courte.*

Cependant, ces réseaux souffrent dans leur structure interne de limitations qui les rendent inopérants dans beaucoup de situations. Si une cellule d'un RNN est finalement très simple, ce n'est pas le cas du LSTM au premier abord.

A ce titre, les blocs ou cellules LSTM interviennent plus efficacement dont les performances n'ont d'égales que leur complexité. Les LSTM ont été créés comme méthode permettant de gérer efficacement la mémoire à court et long terme grâce à leurs systèmes de portes. S'il en existe de nombreuses variantes, les versions d'origine sont encore très largement utilisées dans les meilleurs modèles de Deep Learning pour la prévision des séries temporelles, pour le traitement automatique du langage naturel, mais aussi pour la génération de texte ou l'étude de marchés…

II.2. Réseaux LSTM

Les chercheurs allemands, Hochreiter et Schmidhuber, ont introduit l'idée de réseaux de longue mémoire à court terme dans un article publié en 1997. Le LSTM est un type unique de réseau neuronal récurrent (RNN) capable d'apprendre les dépendances à long terme, ce qui est utile pour certains types de prédiction qui obligent le réseau à conserver les informations sur des périodes plus longues (Hochreiter et Schmidhuber, 1997).

Les LSTM sont conçus pour surmonter le problème du gradient de fuite et leur permettre de conserver les informations pendant de plus longues périodes par rapport aux RNN traditionnels. Les LSTM peuvent maintenir une erreur constante, ce qui leur permet de continuer à apprendre sur de nombreux pas de temps et de se propager à travers le temps et les couches.

Les LSTM permettent de se souvenir de leurs intrants sur une longue période de temps. Ils contiennent leurs informations dans une mémoire, le réseau LSTM peut lire, écrire et supprimer des informations de sa mémoire.

Cette mémoire peut être vue comme une *"cellule gated"*, qui signifie que la cellule décide de stocker ou de supprimer des informations en fonction de l'importance qu'elle attribue à l'information. L'attribution de l'importance se fait à travers des poids, qui sont également appris par l'algorithme. Cela signifie simplement qu'il apprend avec le temps quelle information est importante et laquelle ne l'est pas.

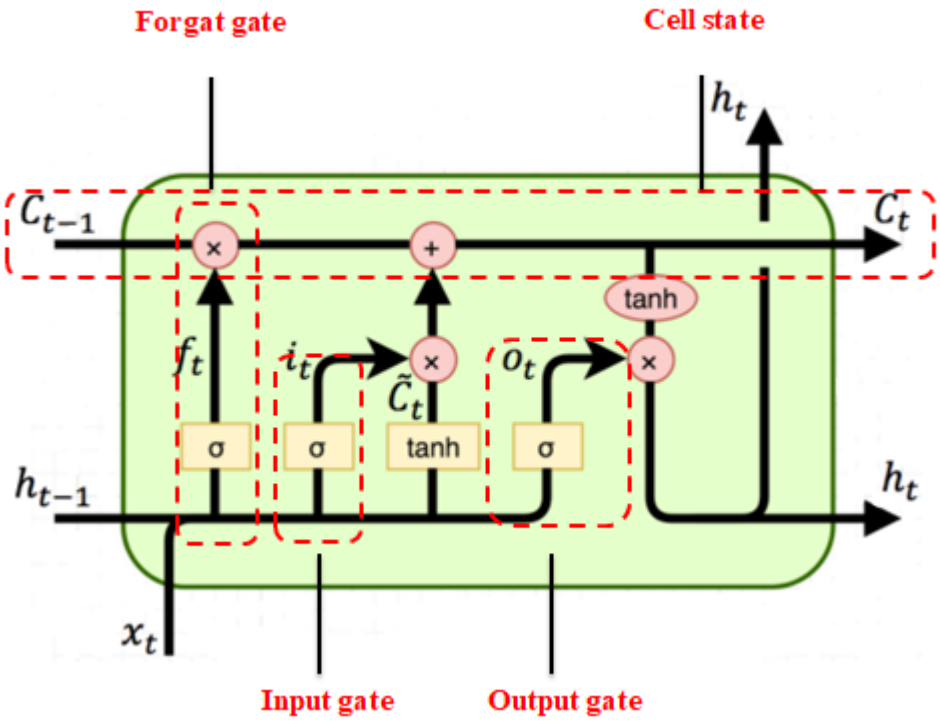


Fig. 19 : - Schéma fonctionnel d'une cellule LSTM.

Une cellule LSTM est composée de trois portes, ceux sont des zones de calculs qui régulent le flot d'informations :

- Input gate (porte d'entrée) : la porte d'entrée responsable de l'ajout d'informations aux cellules ;
- Output gate (porte de sortie) : cette porte sélectionne et produit les informations nécessaires ;
- Forget gate (porte d'oubli) : cette porte supprime les informations qui ne sont plus nécessaires à l'achèvement de la tâche. Cette étape est essentielle pour optimiser les performances du réseau.

On a également deux types de sorties généralement nommées états :

- Hidden state (état caché) ;
- Cell state (état de la cellule).

II.2.1. Architecture de réseau LSTM

L'architecture en forme de chaîne du LSTM lui permet de contenir des informations pendant des périodes plus longues, résolvant des tâches difficiles que les RNN traditionnels ont du mal ou ne peuvent tout simplement pas résoudre. La figure suivante représente une chaîne de trois cellules LSTM:

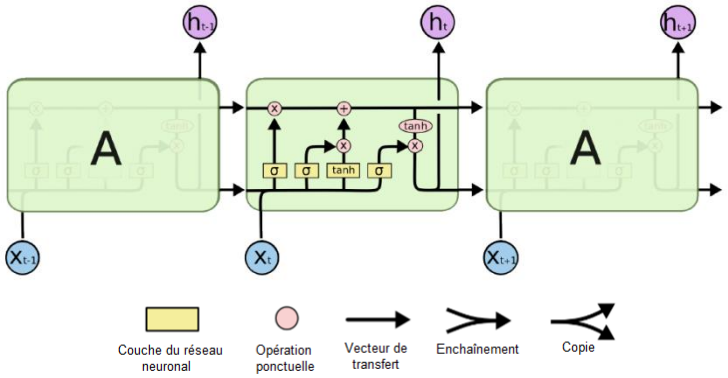


Fig. 20 : - Une chaine de cellules LSTM (colah.github.io, 2015).1

Dans le diagramme ci-dessus, chaque ligne porte un vecteur entier, de la sortie d'un nœud aux entrées des autres. Les cercles roses représentent des opérations ponctuelles, comme l'ajout de vecteurs, tandis que les cases jaunes sont des couches de réseau neuronal d'apprentissage. La fusion des lignes indique la concaténation (ou enchainement), tandis qu'une fourchette de ligne indique que son contenu est copié et que les copies vont à différents endroits.

La clé des LSTM est l'état de la cellule, la ligne horizontale qui traverse le haut du diagramme (Fig. 21). L'état de la cellule est un peu comme une bande transporteuse. Il parcourt toute la chaîne, avec seulement quelques interactions linéaires mineures. Il est très facile pour les informations de circuler inchangées.

Le LSTM a la capacité de supprimer ou d'ajouter des informations à l'état de la cellule, soigneusement régulé par des structures appelées portes. Les portes sont un moyen de laisser éventuellement passer des informations. Ils sont composés d'une couche de réseau neuronal sigmoïde et d'une opération de multiplication ponctuelle.

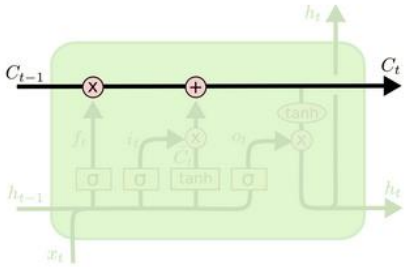
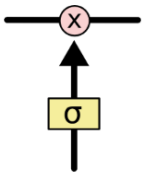


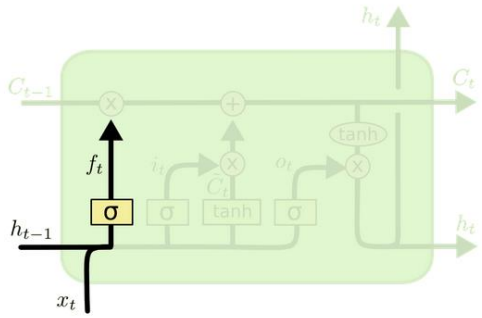
Fig. 21 : - Etat de la cellule LSTM.

La couche sigmoïde produit des nombres entre zéro et un, décrivant la quantité de chaque composant à laisser passer. Une valeur de zéro signifie *ne rien laisser passer*, tandis qu'une valeur de un signifie *tout laisser passer*.



II.2.2. Propriétés et fonctionnement du réseau LSTM

La première étape de notre LSTM est de décider quelles informations vont être rejetées de l'état de la cellule. Cette décision est prise par une couche sigmoïde appelée ou la couche de la porte d'oubli. On observe h_{t-1} et x_t , et on fait sortir un nombre entre 0 et 1 pour chaque nombre dans l'état de celluleC_{t-1}. Un 1 représente : *garder complètement ceci*, tandis qu'un 0 signifie : *se débarrasser complètement de ceci*.



Les calculs se déroulent comme suit :

$$f_t = \sigma(\omega_f . [h_{t-1}, x_t] + b_f) \qquad (27)$$

où :

h_{t-1} : La sortie a l'instant $t-1$.

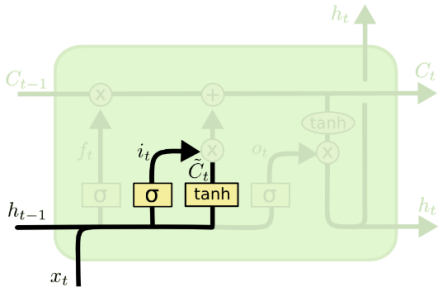
x_t: L'entrée courant a l'instant t.

b : C'est le bais.

ω : C'est le poids.

σ : C'est la fonction sigmoïde.

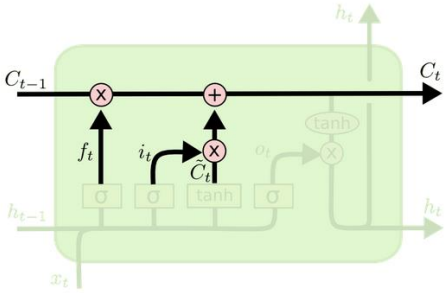
La prochaine étape consiste à décider quelles nouvelles informations nous allons stocker dans l'état de la cellule. Cela comporte deux parties. Tout d'abord, une couche sigmoïde appelée couche de porte d'entrée décide des valeurs que nous mettrons à jour. Ensuite, une couche $tangh$ crée un vecteur de nouvelles valeurs candidates,$\tilde{C}_t$ qui pourraient être ajoutées à l'état. À l'étape suivante, nous combinerons ces deux éléments pour créer une mise à jour de l'état.



$$i_t = \sigma(\omega_i . [h_{t-1}, x_t] + b_i) \qquad (28)$$

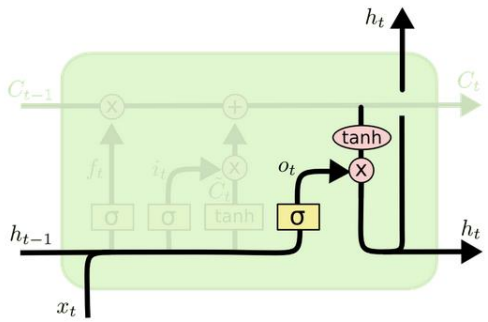
$$\tilde{C}_t = \tanh(\omega_c . [h_{t-1}, x_t] + b_c) \qquad (29)$$

Il est maintenant temps de mettre à jour l'ancien état de cellule, C_{t-1} , dans le nouvel état de cellule C_t . Les étapes précédentes ont déjà décidé ce qu'il fallait faire. On multiplie par la suite l'ancien état par f_t, et oubliant les choses que nous avons décidé d'oublier plus tôt. Puis on l'ajoute $i_t * \tilde{C}_t$.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \qquad (30)$$

Enfin, nous devons décider de ce que nous allons produire. Cette sortie sera basée sur l'état de notre cellule, mais sera une version filtrée. Tout d'abord, nous exécutons une couche sigmoïde qui décide des parties de l'état de la cellule que nous allons générer. Ensuite, nous mettons l'état de la cellule à travers $tangh$ (pour pousser les valeurs entre -1 et 1) et le multiplions par la sortie de la porte sigmoïde, de sorte que nous ne sortons que les parties que nous avons décidées.



$$o_t = \sigma(\omega_0[h_{t-1}, x_t] + b_0) \qquad (31)$$

$$h_t = o_t * \tanh(C_t) \qquad (32)$$

En définitive, le fonctionnement d'un réseau LSTM peut être synthétisé par la figure 22 qui résume les différentes étapes de calcul.

D'autres variantes des réseaux LSTM existent légèrement différentes du réseau LSTM présenté ci-dessus, mais les différences sont mineures. Cependant, pour les lecteurs, il convient d'en mentionner les réseaux proposés par Cho, et al. (2014), Koutnik, et al. (2014), Greff, et al. (2015), Jozefowicz, et al. (2015) et Yao, et al. (2015).

II.3. Algorithmes et Structure du Programme

II.3.1. Introduction

Dans ce travail, deux codes de calcul ont été réalisé sous Matlab pour prédire des séries temporelles. Un premier code assez simple a été réalisé pour se familiariser avec les réseaux LSTM et vérifier leur efficacité. Il a été appliqué par la suite à quelques exemples académiques pour en vérifier le bon fonctionnement, la stabilité lors de l'apprentissage et la convergence lors des tests.

Le deuxième code comprend trois cas, un cas à une seule couche LSTM, le deuxième cas à une couche BiSTLM et un troisième cas à deux couches LSTM.

II.3.2 Organigrammes et Programmes

- **Organigramme 1**

L'organigramme du premier code de calcul comprend huit principales étapes, ceci est illustré par la figure 23.

Les étapes de calcul peuvent être résumées ainsi :

- Après avoir chargé les données, la base contiendra une seule série chronologique avec des pas temps correspondant selon les cas étudiés mensuels ou journaliers et des valeurs correspondant à la dimension de la série. La sortie est un tableau où chaque élément est une étape de temps unique. Là, on redimensionne les données en vecteur de ligne.

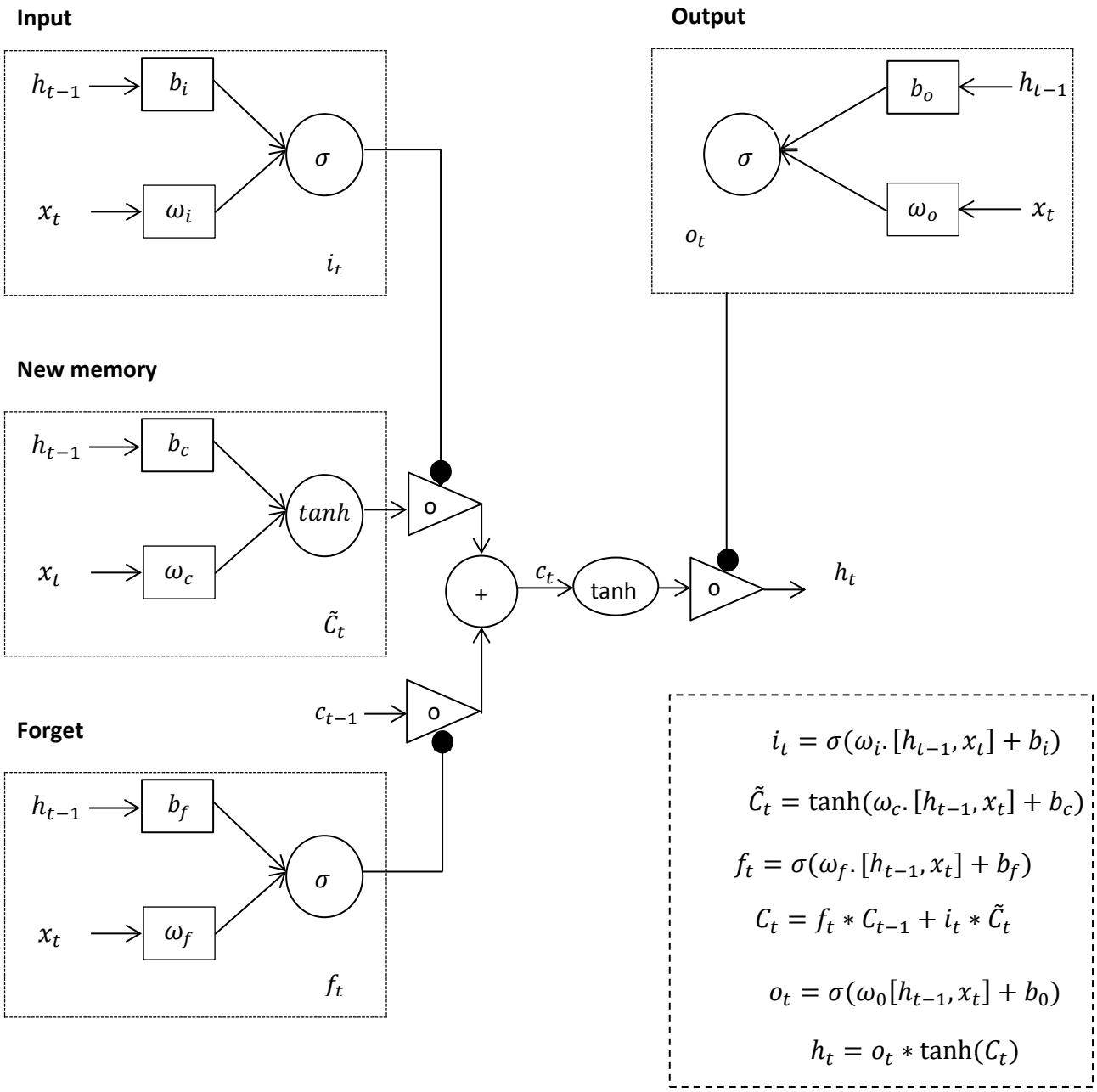


Fig. 22 : - Fonctionnement d'un block LSTM.

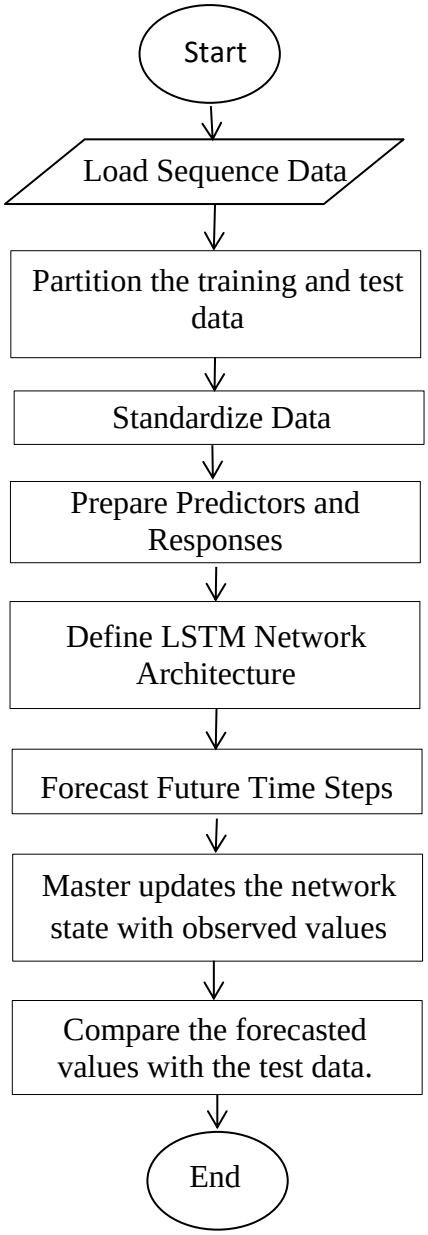


Fig. 23 : - Organigramme d'un code pour la prévision des séries temporelles à l'aide d'un réseau LSTM.

- Ensuite, on doit partitionner les données originales en une séquence d'entraînement ou d'apprentissage et en une séquence de test ou validation. L'apprentissage peut être effectué sur les 90% premiers de la séquence et tester sur les 10% derniers. Il est possible de choisir 80% ou 70% pour l'apprentissage et le reste (20% ou 30%) pour la validation.

- Pour un meilleur ajustement et pour éviter que l'entraînement ne diverge, On doit normaliser ou standardiser les données d'entraînement pour avoir une moyenne nulle et une variance unitaire. Cependant, au moment de la prédiction, on doit normaliser également les données de test en utilisant les mêmes paramètres que les données d'entraînement.

- Pour prévoir les valeurs des futurs pas de temps d'une séquence, on doit spécifier les réponses en tant que séquences d'apprentissage avec des valeurs décalées d'un pas de temps. C'est-à-dire qu'à chaque pas de temps de la séquence d'entrée, le réseau LSTM apprend à prédire la valeur du pas de temps suivant. Les prédicteurs sont les séquences d'entraînement sans le pas de temps final.

- Pour définir l'architecture réseau LSTM, on doit créer un réseau de régression LSTM dont on doit spécifier la couche LSTM pour avoir un certain nombre d'unités cachées. A titre d'exemple :

```
numFeatures = 1;
numResponses = 1;
numHiddenUnits = 200;

layers = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits)
    fullyConnectedLayer(numResponses)
    regressionLayer];
```

On doit spécifier les options de la phase d'apprentissage en réglant le solveur sur «**adam**» et lancer l'apprentissage pendant 250 itérations. On note cependant, que "adam" est un algorithme d'optimisation adaptative du taux d'apprentissage qui a été spécialement conçu pour entraîner les réseaux de neurones profonds (Diederik et al., 2015). Pour éviter toutes divergences ou arrêt du processus, il faut également spécifier le seuil de gradient à la valeur 1. Il faut spécifier la vitesse d'apprentissage initiale à 0.005 et réduire la vitesse d'apprentissage après 125 époques en multipliant par un facteur de 0.2. Toutes ces recommandations, on les retrouves dans le document : Deep Learning Toolbox User's Guide (Beale et al., 2019).

```
options = trainingOptions('adam', ...
    'MaxEpochs',250, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.005, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',125, ...
    'LearnRateDropFactor',0.2, ...
    'Verbose',0, ...
    'Plots','training-progress');
```

- Pour prévoir les valeurs de plusieurs pas de temps dans le futur, nous avons utilisé la fonction « **predictAndUpdateState »** pour prédire les pas de temps un par un et mettre à jour l'état du réseau à chaque prédiction. Pour chaque prédiction, nous utiliserons la prédiction précédente comme entrée de la fonction.

- Pour initialiser l'état du réseau, on commence par prédire sur la base des données de l'apprentissage *XTrain*. Ensuite, on doit faire première prevision en utilisant la dernière étape de la réponse d'apprentissage *YTrain*(end). On fait une boucle sur les prédictions restantes et on fait entrer la prédiction précédente dans "**predictAndUpdateState"** pour la mise à jour.

- Pour les grandes bases de données, les longues séquences ou les grands réseaux, les prédictions sur le GPU sont généralement plus rapides à calculer que les prédictions sur le CPU. Sinon, les prédictions sur le processeur sont généralement plus rapides à calculer. Pour les prédictions à pas de temps unique, on utilise le processeur. Pour utiliser le processeur pour la prédiction, on doit définir l'option "**ExecutionEnvironment**" de "**predictAndUpdateState**" sur "**cpu**"

```
net = predictAndUpdateState (net,XTrain) ;
[net, YPred] = predictAndUpdateState(net,YTain(end)) ;
```

```
numTimeStepsTest = numel(XTest) ;

for I = 2:numTimeStepsTest
[net,YPred(:,i)] = predictAndUpdateState(net,YPred(:,i-1),...
 'ExecutionEnvironment', 'cpu');
End
```

- Au cours de l'apprentissage, le graphique de progression d'apprentissage rapporte l'erreur quadratique moyenne (RMSE) calculée à partir des données standardisées. Après, le calage, on calcul le RMSE à partir des prédictions non standardisées ainsi que les autres paramètres d'évaluation.

- **Organigramme 2**

Le deuxième code de calcul est plus adapté aux séries temporelles irrégulières, bruitées et complexes à l'aide d'une couche LSTM qui apprend les dépendances à long terme entre les pas de temps de la série. Ici, on pondère automatiquement les données antérieures. Dans ce code, nous avons défini un générateur de nombre aléatoire sur sa valeur par défaut, afin qu'à chaque fois après l'exécution du script, nous aurons le même résultat. Le code à un réseau présentant trois cas différents qui seront détaillées plus loin. La figure 24, représente l'organigramme qui résume le code de calcul. Ce code est plus intéressant lorsque nous avons des dizaines ou des centaines de couches. Nous proposons également la possibilité d'essayer une couche BiLSTM et deux couches LSTM.

Les principales différences par rapport au premier code peuvent être résumées ainsi :

- Pour le premier cas à une seule couche LSTM, le réseau neuronal se compose de quatre couches:
  - sequenceInputLayer (numFeatures) ;
  - lstmLayer (numHiddenUnits) ;
  - fullyConnectedLayer (numResponses) ;
  - regressionLayer.

  1. **sequenceInputLayer** (inputSize) définit une couche d'entrée de séquence. inputSize est la taille de la séquence d'entrée à chaque pas de temps. Dans notre cas, la séquence n'est que la dernière valeur de la séquence temporelle, donc inputSize est 1. On peut avoir des séquences plus longues.
  2. **lstmLayer** (numHiddenUnits) crée une couche de longue mémoire à court terme. numHiddenUnits est le nombre d'unités cachées dans la couche. Le nombre d'unités cachées est le nombre de neurones dans la couche.
  3. **fullyConnectedLayer** crée une couche entièrement connectée avec une taille de sortie spécifiée. Elle multiplie l'entrée par une matrice de poids, puis ajoute un vecteur de biais.
  4. **regressionLayer** crée une couche de sortie de régression pour un réseau neuronal. La régression est un ajustement des données.

- Pour le deuxième cas, le réseau neuronal se compose de quatre couches dont une est la couche **bilstmLayer** :
  - sequenceInputLayer(numFeatures)
  - bilstmLayer(numHiddenUnits)
  - fullyConnectedLayer(numResponses)
  - regressionLayer

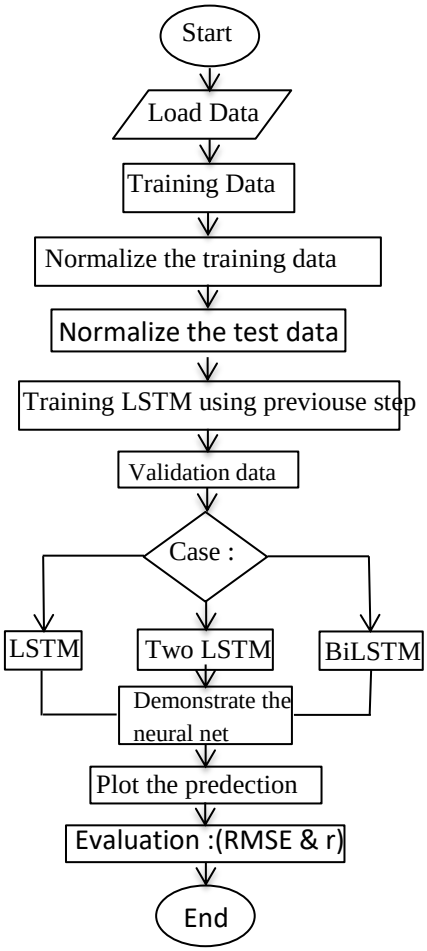


Fig. 24 : - Organigramme du deuxième code à trois cas : LSTM, Two LSTM et BiLSTM
Pour la prévision des séries temporelles.

La différence par rapport au premier cas, est que ce réseau dispose d'une couche **bilstmLayer** qui est une couche LSTM bidirectionnelle (BiLSTM), elle apprend les dépendances bidirectionnelles à long terme entre les pas de temps des séries temporelles ou des données de séquence. Ces dépendances peuvent être utiles lorsque nous souhaitons que le réseau apprenne de la série chronologique complète à chaque pas de temps.

Dans notre travail, l'application du cas d'une couche BiLSTM, été uniquement à titre comparatif avec les deux cas précédents. On a limité son application pour plusieurs raisons dont la plus importante raison est le temps de calcul, mais le code que nous avons établi permet des applications qui pourront être testées dans d'autres travaux qui feront suites à ce travail. Le schéma ci-dessous (fig. 25) montre le fonctionnement d'une couche BiLSTM.

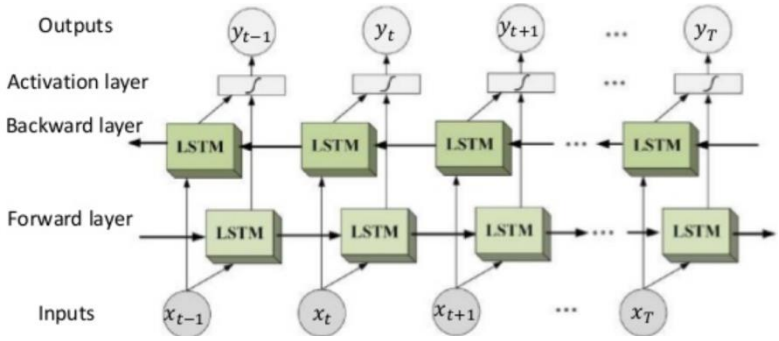


Fig. 25 : - Schéma fonctionnel d'une couche BiLSTM.

- Pour le troisième cas à deux couches LSTM, le réseau neuronal se compose de six couches:
  - sequenceInputLayer (numFeatures)
  - lstmLayer (numHiddenUnits)
  - reluLayer
  - lstmLayer (numHiddenUnits)
  - fullyConnectedLayer (numResponses)
  - regressionLayer.

Par rapport au premier cas, nous avons en plus une deuxième couche LSTM et une couche **reluLayer** qui effectue une opération de seuil sur chaque élément de l'entrée, où toute valeur inférieure à zéro est mise à zéro.

La partie centrale et la plus importante du programme de calcul est présentée ci-dessous.

```
switch layerSet
    case 'lstm'
        layers = [sequenceInputLayer(numFeatures)
            lstmLayer(numHiddenUnits)
            fullyConnectedLayer(numResponses)
            regressionLayer];
    case 'bilstm'
        layers = [sequenceInputLayer(numFeatures)
            bilstmLayer(numHiddenUnits)
            fullyConnectedLayer(numResponses)
            regressionLayer];
    case 'two lstm'
        layers = [sequenceInputLayer(numFeatures)
            lstmLayer(numHiddenUnits)
            reluLayer
            lstmLayer(numHiddenUnits)
            fullyConnectedLayer(numResponses)
            regressionLayer];
    otherwise
        error('Only 3 sets of layers are available');
end
```

II.3.3. Performance des algorithmes

La performance des méthodes utilisées ont été validées par des paramètres statistiques lors des phases d'apprentissage et de test. Les paramètres statistiques utilisés dans ce travail sont : L'erreur quadratique moyenne ***RMSE*** (*the root-mean-square error*), la fonction ***LOSS*** et le coefficient de corrélation ***r***. Ces paramètres sont donnés par les relations suivantes :

- **RMSE**

En statistiques, l'erreur quadratique moyenne est une mesure caractérisant la « précision » de l'estimateur. Elle est plus souvent appelée « erreur quadratique » ; elle est parfois appelée aussi « risque quadratique ».
L'erreur quadratique moyenne est définie par :

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(Xt_i - \hat{Y}t_i\right)^2}{N}} \tag{33}$$

où

Xt_i représentent les débits observés ;
Yt_i représentent les débits simulés.

- **Fonction LOSS**

Dans ce cas, on estime les paramètres du modèle en minimisant l'erreur entre la sortie du modèle et la réponse mesurée. Cette erreur, appelée « Loss Function » est une fonction positive des erreurs de prédiction $e(t)$. Ce paramètre n'a été estimé que dans la phase d'apprentissage pour en vérifier l'évolution de l'erreur et apprécier la convergence. En général, cette fonction est une somme pondérée des carrés des erreurs. Pour un modèle à ny sorties, la fonction Loss $V(\theta)$ a la forme générale suivante:

$$V(\theta) = \frac{1}{N}\sum_{t=1}^{N} e^{T}(t,\theta)\, W(\theta)\, e(t,\theta) \tag{34}$$

où:
N est le nombre d'échantillons de données.

$e(t,\theta)$ est un vecteur d'erreur ny-par-1 à un instant t donné, paramétré par le vecteur de paramètresθ.

$W(\theta)$ est la matrice de pondération, spécifiée comme une matrice semi-définie positive. Si W est une matrice diagonale, vous pouvez la considérer comme un moyen de contrôler l'importance relative des sorties lors des estimations multi-sorties. Lorsque W est un poids fixe ou connu, il ne dépend pas deθ.

- **Coefficient de corrélation (r)**

Le coefficient de corrélation entre deux variables aléatoires réelles X et Y ayant chacune une variance, noté $Cov(X,Y)$, ou parfois ou simplement r, est défini par :

$$r = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{35}$$

où $Cov(X,Y)$ désigne la covariance des variables X et Y.

II.3.4. Vérification des codes de calcul

Les codes de calcul ont testés sur des données académiques pour en vérifier l'efficacité. Une première fonction sinusoïdale bruitée légèrement a fait l'objet d'exemple de teste pour le premier programme. Pour le deuxième programme, on a choisi trois séries temporelles académiques pour les tests, une première série aléatoire, une deuxième série celle du processus Brownien et une troisième série chaotique de Lorenz.

- **Exemple d'une fonction bruitée**

Pour tester l'efficacité du premier programme, on a choisi une fonction sinusoïdale bruitée à 5 %. La fonction a la forme suivante :

$$X = \sin(t) + \cos\left(2 * 3.14 * \frac{t}{2}\right) + \varepsilon \tag{36}$$

La série est représentée par la figure 26 suivante :

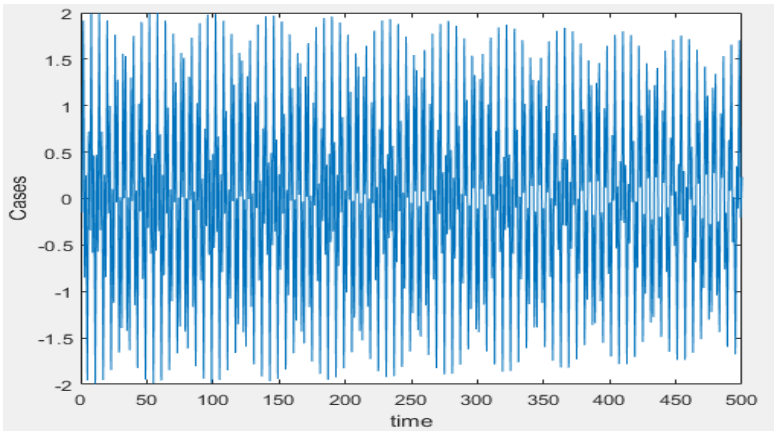


Fig. 26 : - Série temporelle.

Les résultats de la phase apprentissage sont résumés par la figure 27 qui illustre le processus d'apprentissage.

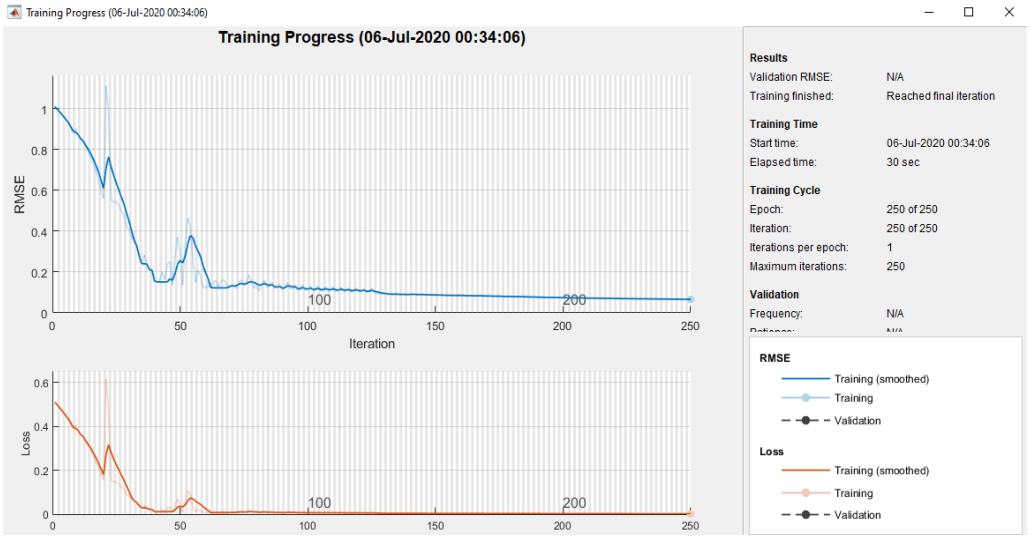


Fig. 27 : - Processus d'Apprentissage.

La prévision de la série chronologique à l'aide du premier programme a permis d'avoir les résultats ci-dessous.

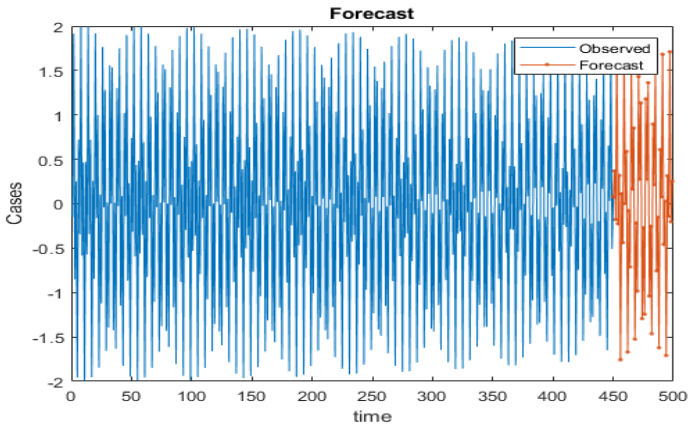
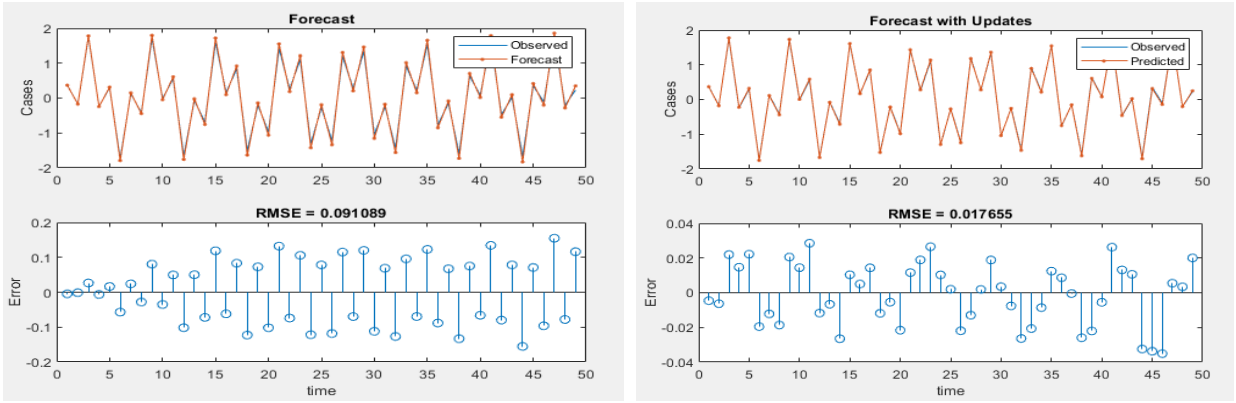


Fig. 28 : - Prévision avec une seule couche LSTM.



a **b**

Fig. 29 : - Prévisions : (a)- avant la mise à jour ; (b)- après la mise à jour.

On constate que les résultats obtenus pour ce premier exemple sont très encourageants et confirment l'efficacité du code de calcul. Un RMSE=0.01765, un coefficient de corrélation r=0.999 et des courbes de valeurs observées et simulées parfaitement confondues (Fig. 29) montrent bien la robustesse des algorithmes utilisés. Cependant, il reste à tester ces codes sur des données plus complexes pour confirmer la l'efficience du programme de calcul.

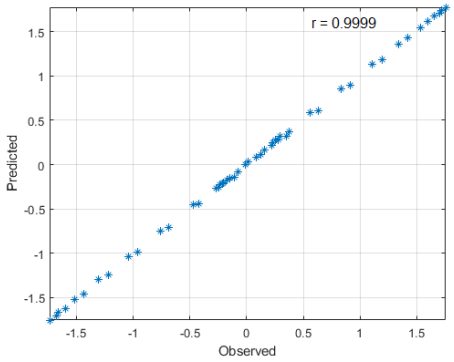


Fig.30 : - Comparaison des valeurs observés et des simulés dans la phase de test après la mise à jour.

- **Exemple d'une série chaotique**

Pour cet exemple, on a choisi le système de Lorenz exprimé sous la forme d'un système de trois équations différentielles d'ordre 1 :

$$\frac{dx}{dt} = \sigma\,(y - x) \quad et \quad \frac{dy}{dt} = \rho x - y - xz \quad et \quad \frac{dz}{dt} = -\beta z + xy \tag{37}$$

où $x,\ y,\ et\ z$ sont les variables dépendantes du temps et les paramètres $\sigma,\ \beta,\ et\ \rho$ sont connus.

Le système de Lorenz est un système d'équation non linéaire à cause des termes $xy\ et\ xz$. Pour cet exemple, on a choisi comme valeurs $\sigma = 16,\ \rho = 45.92\ et\ \beta = 4.0$, aves les conditions initiales $x(0) = y(0) = z(0) = 0.01$, il en résulte la série chronologique ci-contre :

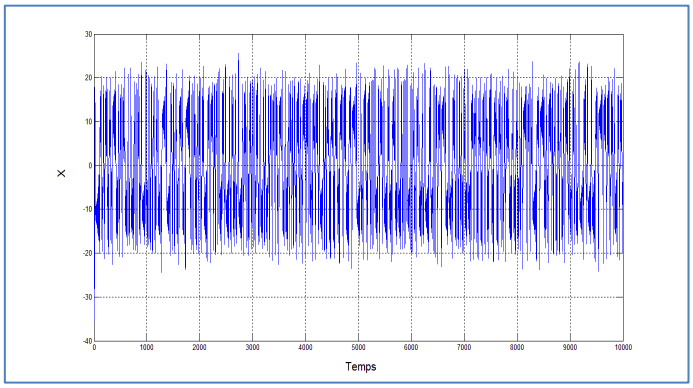


Fig.31 : - Série chronologique chaotique de Lorenz

En choisissant uniquement les 2000 premières valeurs de la série à cause de la durée et la lenteur du processus de calcul, l'application du premier code de calcul a permis d'avoir les résultats ci-dessous :

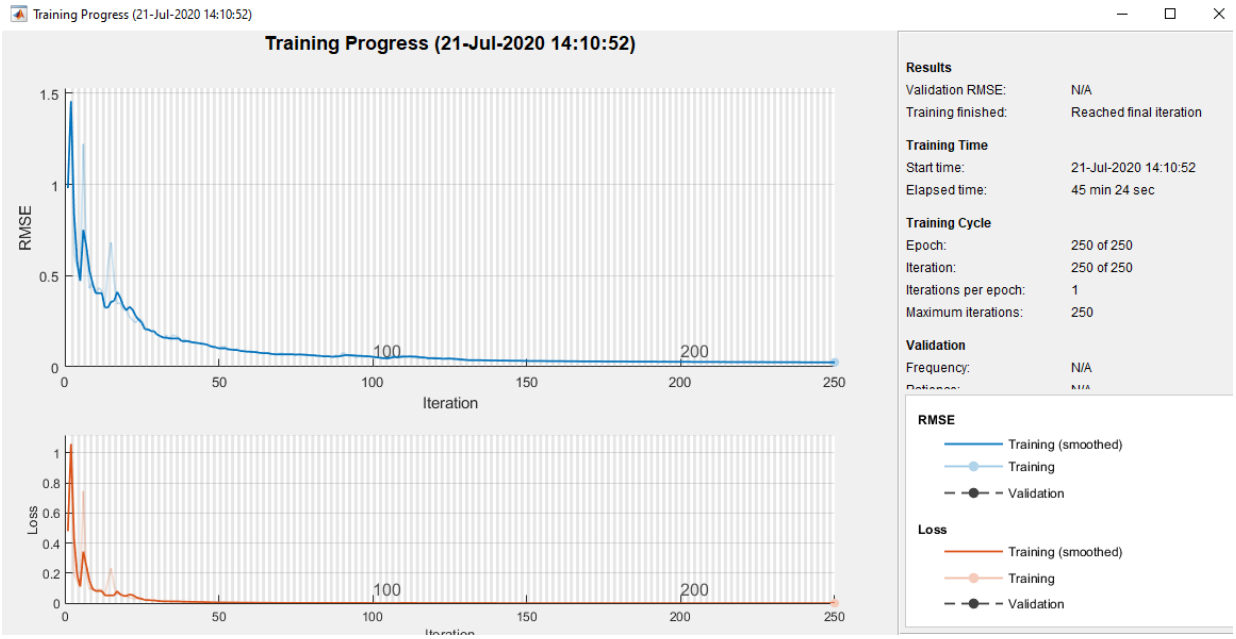


Fig.32 : - Processus d'apprentissage de la série de Lorenz.

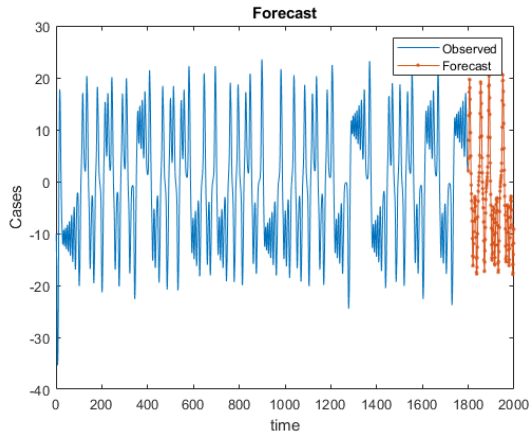


Fig.33 : - Prévision de la série de Lorenz.

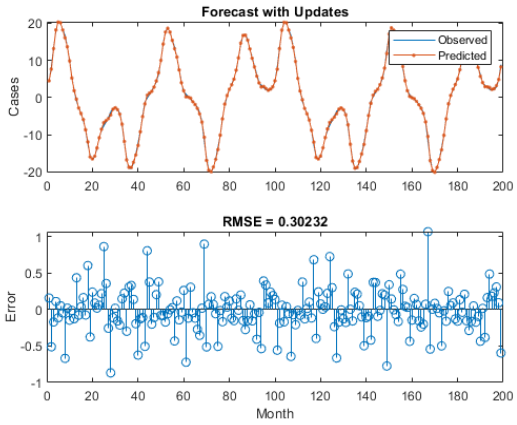


Fig.34 : - Prévision et comparaison des valeurs observées et simulées.

Nous constatons (Fig. 32) que le processus d'apprentissage converge rapidement et que le RMSE devient stable après 150 itérations. De même pour la fonction Loss est stable aux alentours de 50 itérations. La figure 33 montre les deux parties de la série : en bleu la chronique ayant été utilisée pour l'apprentissage et la partie en rouge prévue par le modèle. La figure 34 montre la comparaison des valeurs observées et simulées et leur erreur, la figure montre une parfaite correspondance. Les valeurs observées et simulées s'alignent parfaitement sur la figure 35 et le coefficient de corrélation r est 0.9996.

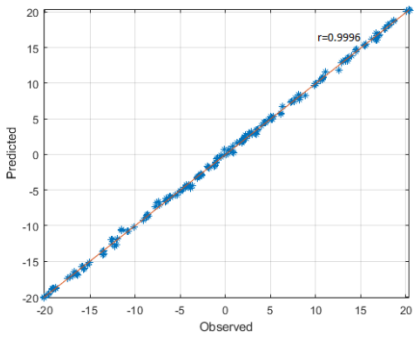


Fig. 35 : - Comparaison des valeurs observées et simulées de la série de Lorenz dans la phase de Test.

- **Exemple d'une série d'un processus Brownien**

L'exemple du mouvement Brownien a été choisi dans ce travail car les données hydrologiques ont une grande similitude avec ce processus.

Par définition, le mouvement brownien est une description mathématique du mouvement aléatoire d'une « grosse » particule immergée dans un fluide et qui n'est soumise à aucune autre interaction que des chocs avec les petites molécules du fluide environnant. Il en résulte un mouvement très irrégulier de la grosse particule (Fig. 36), qui a été décrit pour la première fois en 1827 par le botaniste Robert Brown en observant des mouvements de particules à l'intérieur de grains de pollen.

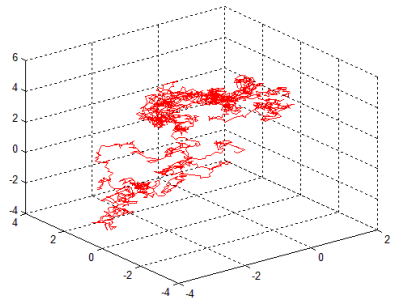


Fig.36 : - Mouvement brownien tridimensionnel par une marche aléatoire.

Le mouvement brownien unidimensionnel est un processus stochastique dépendant du temps et vérifiant certaines conditions parmi lesquelles l'accroissement devrait être indépendant du processus et que cet accroissement doit être une variable aléatoire normale de moyenne nulle.

Nous avons générés une série chronologique d'un processus Brownien sous Matlab (Fig. 37).

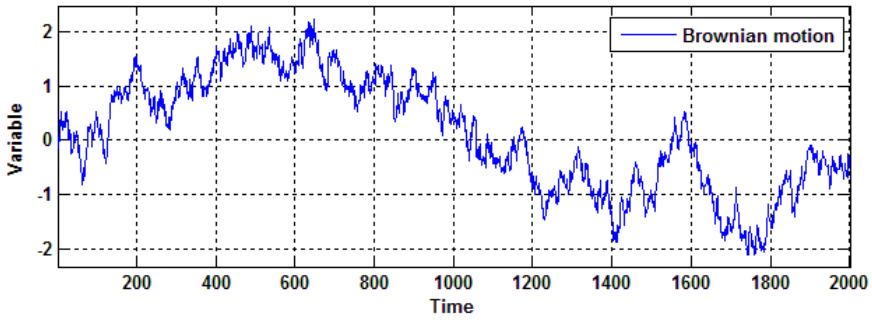


Fig. 37 : - Série chronologique d'un mouvement Brownien.

Les résultats de simulation à l'aide du premier code s'avèrent intéressants : RMSE = 0.1010 et un coefficient de corrélation r=0.9600. Cependant, les résultats obtenus à l'aide du deuxième code de calcul sont plus intéressants et plus précis, ils sont résumés par les figures suivantes :

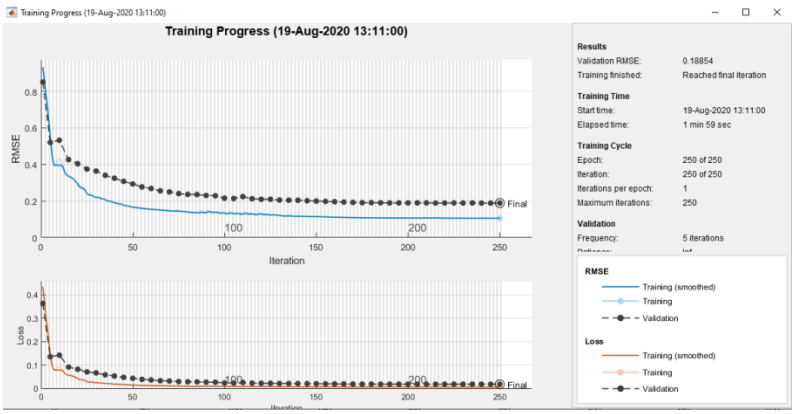


Fig. 38 : - Processus d'apprentissage de la série du mouvement Brownien.

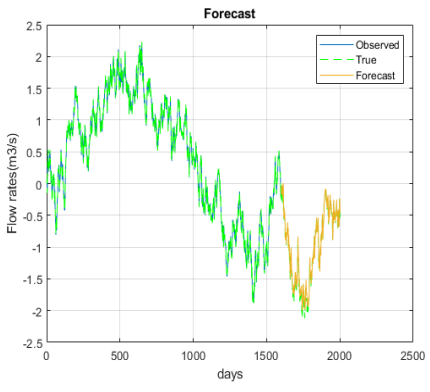


Fig. 39 : - Prévision de la série du mouvement Brownien.

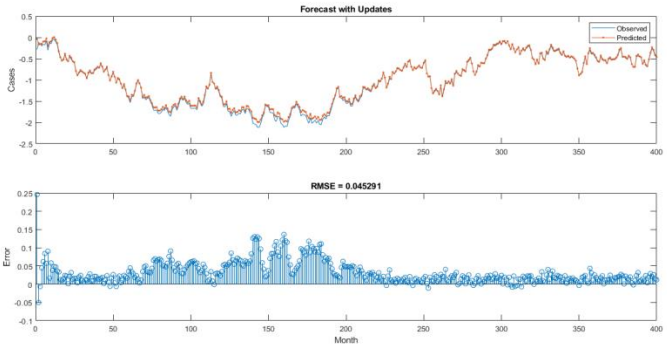


Fig. 40 : - Prévision et comparaison des variables observées et simulées dans la phase de test de la chronique du mouvement Brownien.

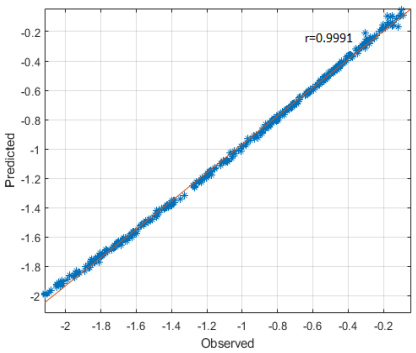


Fig. 41 : - Comparaison des valeurs observées et simulées de la série du mouvement Brownien dans la phase de Test.

A travers ces figures qui illustrent les résultats obtenus via le deuxième code à deux couches LSTM, pour une valeur de RMSE égale à 0.045 et un coefficient de corrélation égal à 0.9991, on peut conclure de l'efficacité du modèle.

- **Exemple d'une série d'un processus Aléatoire**

Sachant qu'en mathématique, une série aléatoire ne possède aucune structure ni régularité ni règle de prédiction identifiable, c'est-à-dire qu'aucune « stratégie effective ». Cet exemple permettra une fois de plus de confirmer la raisonnabilité du code de calcul, c'est-à-dire capable d'activité logique de raisonnement et conforme à la raison discursive aux règles du raisonnement.

D'un autre coté, de nombreux auteurs ont montré que les séries de pluies en Algérie ont le plus souvent un caractère aléatoire qui se traduit lors des analyses stochastiques par un comportement de bruit Gaussien fractionnaire (Chettih et Mesbah, 2006) (Taouti, 2012).

A ce titre, nous avons généré une série aléatoire (pseudo aléatoire) de 2000 valeurs sous Matlab présentée par la figure 42.

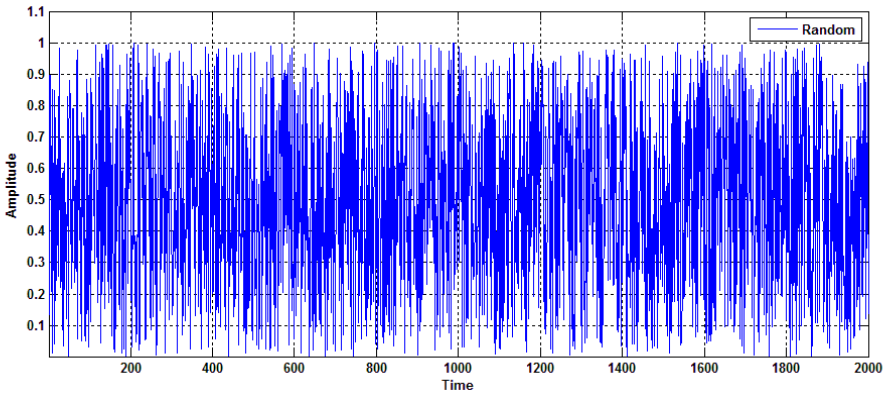


Fig. 42 : - Série chronologique d'un processus aléatoire.

Les résultats obtenus à l'aide des deux codes étaient très semblables, mais le temps de calcul était très long pour le deuxième code utilisant deux couches LSTM.

On se contentera des résultats du premier code de calcul à une seule couche LSTM, qui donne ses meilleurs résultats pour 1000 couches cachées et un temps raisonnable de calcul de 211 minutes pour un nombre d'itérations de 250. La figure 43 ci-dessous résume le processus d'apprentissage. La figure visualise l'évolution de RMSE et de Loss.

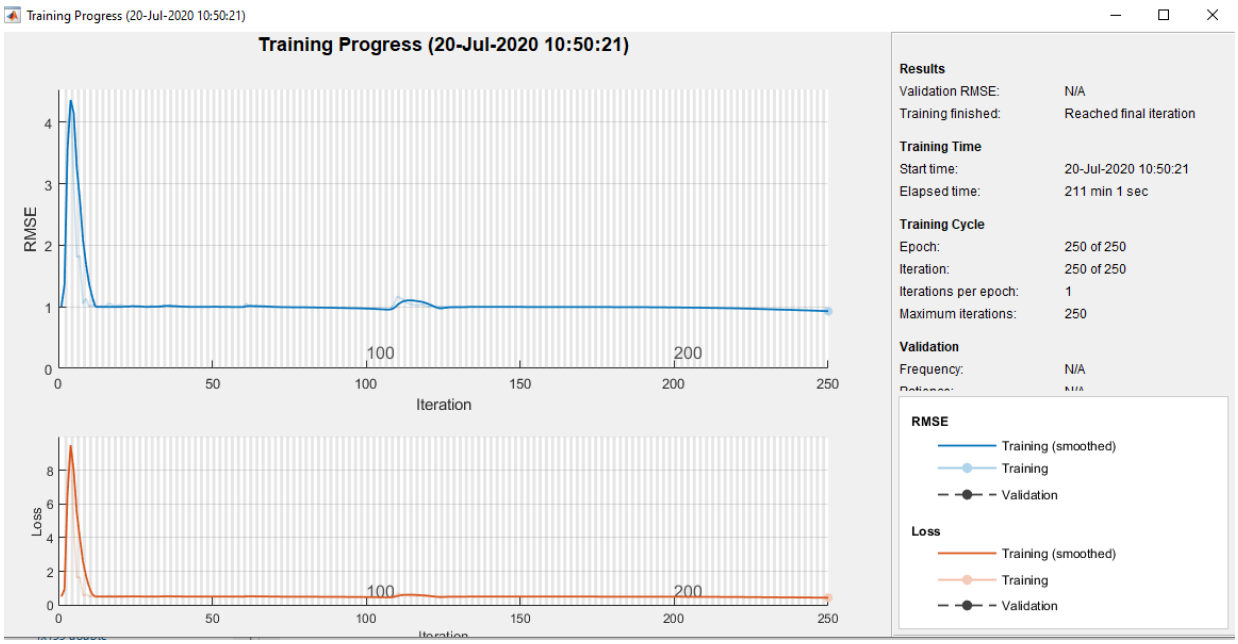


Fig. 43 : - Processus d'apprentissage de la série aléatoire.

La prévision est illustrée par la figure 44, où nous constatons une très mauvaise correspondance des valeurs observées et simulées en rouge ainsi que les erreurs sur les valeurs sur le graphique en dessous. La comparaison entre les valeurs observées et simulées est donnée par la figure 45 dont le coefficient de corrélation r n'est pas significatif de valeur égale à 0.0122. Ces mauvais résultats obtenus pour une série aléatoire confirment la raisonnabilité du code.

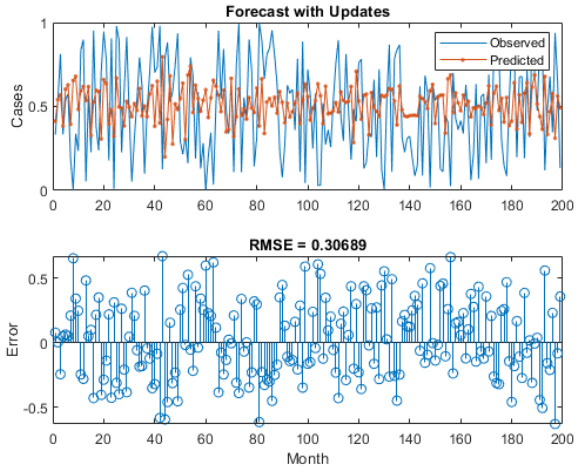


Fig. 44 : - Prévision et comparaison des variables observées et simulées dans la phase de test de la série aléatoire.

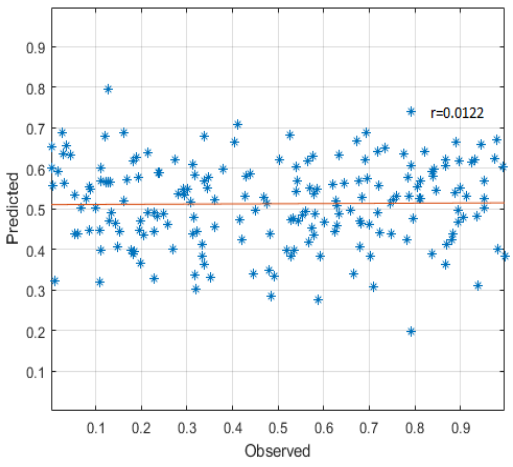


Fig. 45 : - Comparaison des valeurs observées et simulées de la série aléatoire dans la phase de Test.

II.4. Conclusion

Les algorithmes décrits dans ce chapitre et leur implémentation sous Matlab pour les prévisions des séries temporelles ont montré une très haute performance sur des cas théoriques (fonction sinusoïdale bruitée, série chaotique, mouvement brownien). Par ailleurs, ils ont confirmé leur raisonnabilité pour une série aléatoire qui ne possède aucune structure ni régularité ni règle de prédiction identifiable.

Le premier code de calcul à une seule couche LSTM a permis de se familiariser avec la méthode LSTM et le programme lui-même et de le tester en faisant varier certains paramètres comme le nombre d'unité cachée (numHiddenUnits), le nombre d'itération (MaxEpochs), ainsi que d'autres paramètres d'options.

Le deuxième code de calcul comprend trois cas : une seule couche LSTM, deux couches LSTM et une couche BiLSTM. Les deux derniers cas sont plus recommandés pour des séries très complexes où ils s'avèrent plus performants, mais cet avantage est au détriment du temps de calcul.

L'application de ces programmes aux séries hydrologiques de l'Algérie septentrionale permettra d'une part de tester une fois de plus l'efficacité des codes à la prévision hydrologique et de contribuer d'autre part à comprendre la complexité ou la simplicité du comportement hydrologique.

Chapitre 3

Deep Learning pour la prévision Hydrologique

III.1. Introduction

Le concept de Deep Learning s'est développé récemment grâce à l'avancée des performances de calcul des ordinateurs ce qui constitue une raison motivante pour son application. Ainsi, l'application du Deep Learning dans cette étude aux pluies, aux débits aux indices d'oscillations atmosphériques de quelques régions d'Algérie du Nord était principalement pour se familiariser avec cette nouvelle méthode. Ainsi, trois approches basées sur les réseaux LSTM ont été implémentées : une première à une seule couche LSTM, une deuxième à deux couches LSTM et une troisième BiLSTM.

Le choix des réseaux LSTM est justifié par le fait que les séries hydrologiques ont très souvent un effet mémoire très long, car les réseaux LSTM sont capables d'apprendre les dépendances à long terme et conservent les informations sur des périodes plus longues.

Cependant, l'un des inconvénients majeurs des réseaux LSTM est le temps de calcul considérable. Afin d'avoir un temps de calcul raisonnable et qui assure une bonne convergence, nous avons choisi d'utiliser une base de données uni-vectorielle et un nombre d'itération raisonnable. Le nombre de couches cachées a été choisi d'une façon expérimentale jusqu'à avoir le meilleur des résultats. L'apprentissage en profondeur permet aux modèles de calcul composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction. Le Deep Learning pourrait découvrir une structure complexe dans les ensembles de données et modifier ses paramètres internes à l'aide des algorithmes de rétropropagation. Ceci évite le va-et-vient dans les modèles classiques pour faire des modifications et assure un très bon apprentissage.

III.2. Présentation de la région d'étude

III.2.1. Situation géographique

L'Algérie du Nord de 150 000 km^2 environ de superficie est limitée au Nord par la mer Méditerranée et au Sud par l'Atlas Saharien (Fig. 46). C'est un pays de montagnes basses et moyennes, de hautes plaines et de plateaux. L'altitude moyenne ne dépasse pas 900 m. La partie côtière est occupée par des plaines basses petites et d'anciens massifs cristallins très démembrés. En limite Sud se trouve le système montagneux de l'Atlas Tellien, dont certains sommets atteignent ou dépassent 2000 m d'altitude. Ces montagnes sont coupées de gorges profondes où

s'intercalent d'importantes cuvettes.

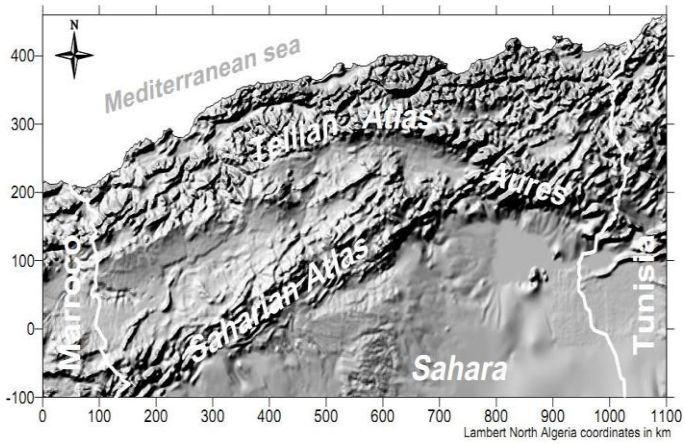


Fig. 46 : - Situation et orographie de l'Algérie du Nord.

C'est le secteur le plus pluvieux où sont situés les principaux Barrages, plus de 80 barrages sont opérationnels.

III.2.2. Climatologie

Le climat de l'Algérie du Nord est subtropical méditerranéen, se caractérise par un été très chaud et sec et un hiver relativement chaud et pluvieux. Dans les hauts-plateaux le climat est plus sec et plus continental. Le facteur prédominant dans la formation de l'écoulement des cours d'eau est évidemment la précipitation, qui est très irrégulière durant l'année.

L'altitude et l'orientation des chaînes montagneuses vis-à-vis des vents dominants, l'éloignement de la mer et la nature de la couverture végétale ont une grande influence sur les précipitations et leur répartition spatiale (Taouti, 2012).

C'est pour cela que les précipitations moyennes annuelles en Algérie septentrionale varient très fortement d'une région à l'autre (Fig. 47). Elles s'échelonnent de 1600 à 1900 mm sur les cimes des montagnes de l'Atlas Tellien jusqu'à 200 mm dans les régions intérieures du pays.

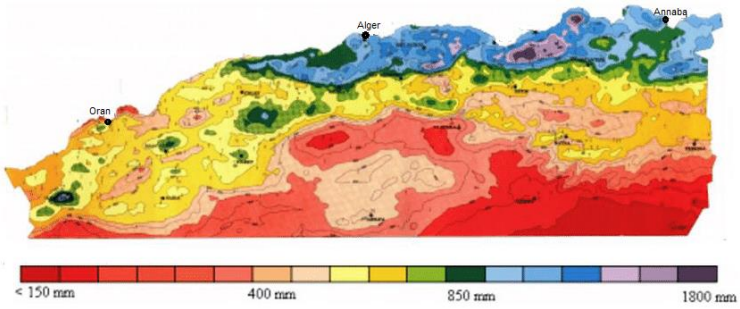


Fig. 47 : - Carte de la pluviométrie annuelle de l'Algérie du Nord (d'après l'ANRH 1993).

En hiver, l'épaisseur de la neige n'est jamais importante mais peut atteindre 30 cm et plus en montagne. Les températures moyennes annuelles de l'air sont de l'ordre de 18°C dans les régions côtières et de 14°C à l'intérieur. Le couvert végétal naturel est méditerranéen ou semi-désertique et obéit à la loi de l'étagement en montagne.

III.2.3. Hydrographie

L'Algérie du Nord se divise en deux régions hydrographiquement bien distinctes : ce sont les bassins versants dont les cours d'eau qui se jettent en mer Méditerranée correspondants aux bassins exoréiques de superficie de 130000 km², et les bassins endoréiques de superficie de 20000 km^2 (Fig. 48).

Selon la classification des régimes des rivières, les cours d'eau algériens entrent dans le type méditerranéen. Leur régime d'écoulement est conditionné par le caractère des précipitations météoriques caractérisées par une forte irrégularité saisonnière et interannuelle (Taouti, 2012).

Les cours d'eau algériens se caractérisent aussi par une grande quantité de transport solide. La turbidité moyenne annuelle de certains d'entre eux dépasse 30 kg/1 (Demmak, 1982).

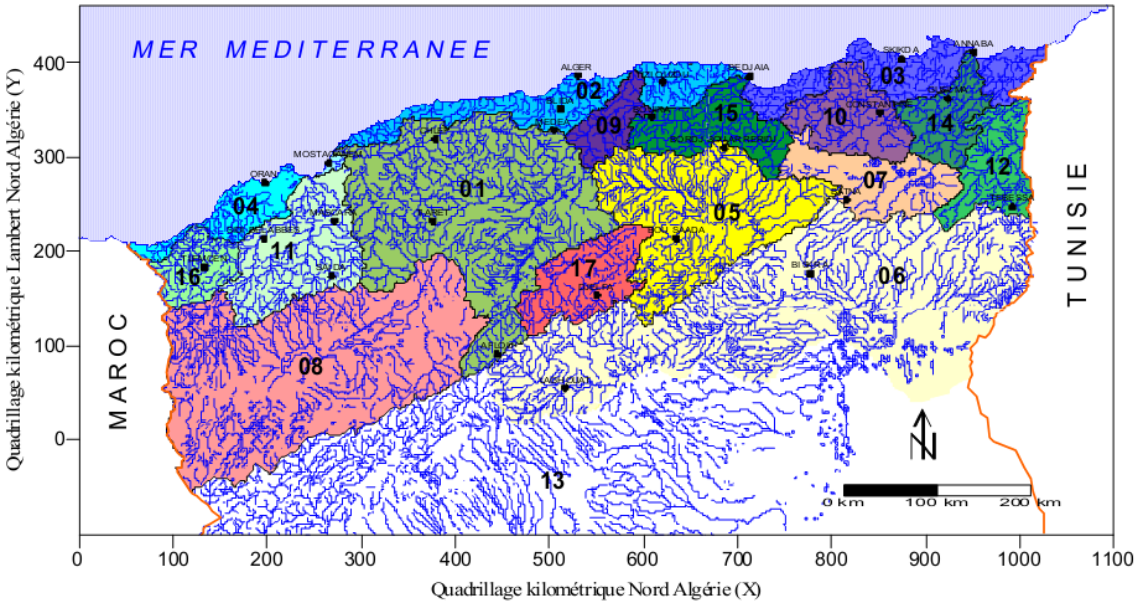


Fig. 48 : - Réseaux hydrographiques de l'Algérie du Nord.

III.3. Présentation des données

III.3.1. Précipitations

Nous nous intéressons dans ce travail aux données pluviométriques de l'Algérie Septentrionale au pas mensuel pour les stations d'Annaba, Dar El Beida et Oran (Fig. 49).

Au pas mensuel, les pluies présentent un processus complexe difficilement prévisible plus proche d'un processus aléatoire que d'un phénomène déterministe qui nous motive et fait tous l'intérêt de tester le Deep Learning.

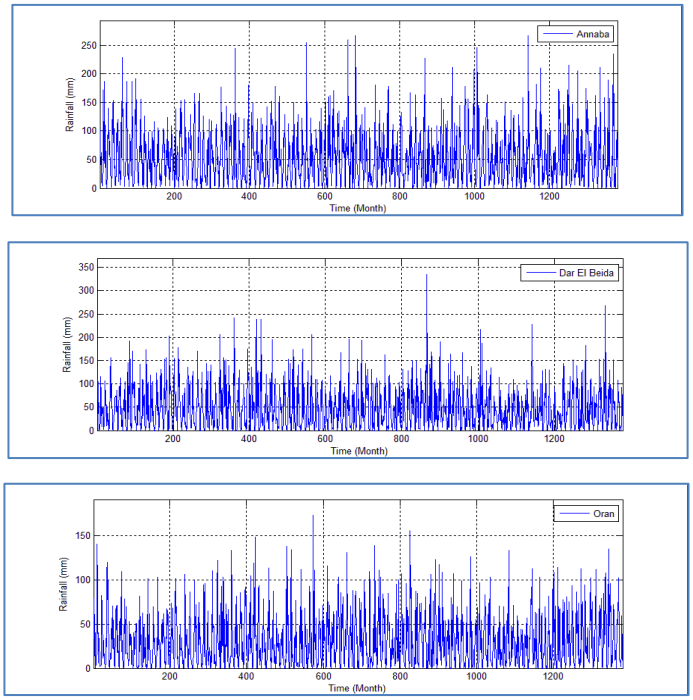


Fig. 49 : – Pluies mensuelles (1901-2015, Annaba, Dar El Beida et Oran).

Le tableau ci-dessous résume pour les stations pluviométriques citées précédemment les périodes d'observations, le type du pas d'échantillonnage, les coordonnées des stations et les sources ayant fournies ces données :

Stations	Périodes	Type	Coordonnées	Sources
Annaba	1901-2015	mensuel	36.89° N - 7.79° E	http://sdwebx.worldbank.org/climateportal/ ANRH & ONM
Dar El Beida	1901-2015	mensuel	36.54° N - 3.19° E	
Oran	1901-2015	mensuel	35.69° N - 0.66° O	

Tableau 01 : Stations d'observations et sources des données de pluies.

Les principales caractéristiques statistiques des pluies mensuelles sont résumées dans le tableau suivant :

Stations	Moyennes	Variances	Coef. Vari.	Min	Max
Annaba	54.3402	2.233*10^3	0.8697	0	266.60
Dar El Beida	47.8459	2.128*10^3	0.9642	0	334.60
Oran	31.0861	926.5115	0.9792	0	173.30

Tableau 02 : Caractéristiques statistiques des pluies mensuelles.

III.3.2. Indices d'oscillation climatiques

Les indices climatiques que nous avons choisis dans notre étude ont une influence sur le climat de l'Algérie surtout du Nord. Ils représentent une différence de pression calculée entre deux points : un de haute pression et l'autre de basse pression. Leur variabilité constitue un processus fractal très complexe (Lana et al., 2016). La difficulté de prédictibilité constitue un défi que nous souhaitons relever à l'aide du Deep Learning.

- **Oscillation Nord-Atlantique (NAO)**

L'Oscillation Nord-Atlantique NAO, pour North Atlantic Oscillation est un phénomène à la fois océanique et atmosphérique. Elle désigne un phénomène touchant le système climatique du nord de l'océan Atlantique. Ce phénomène fait référence aux mouvements de va-et-vient, selon un axe Nord-Sud, de masses d'air situées au-dessus de l'Arctique et de l'Islande en direction des Açores et de la péninsule Ibérique. La NAO influence le climat sur tout le pourtour de l'Atlantique Nord, principalement en Europe. Elle occasionne entre autres des changements de pression au sol qui conditionnent directement la position et l'intensité de l'anticyclone des Açores (Fig. 50). La NAO décrit les variations du régime océan-atmosphère sur la région et se mesure généralement comme la différence de pression atmosphérique entre l'anticyclone des Açores et la dépression d'Islande (Fig. 51).

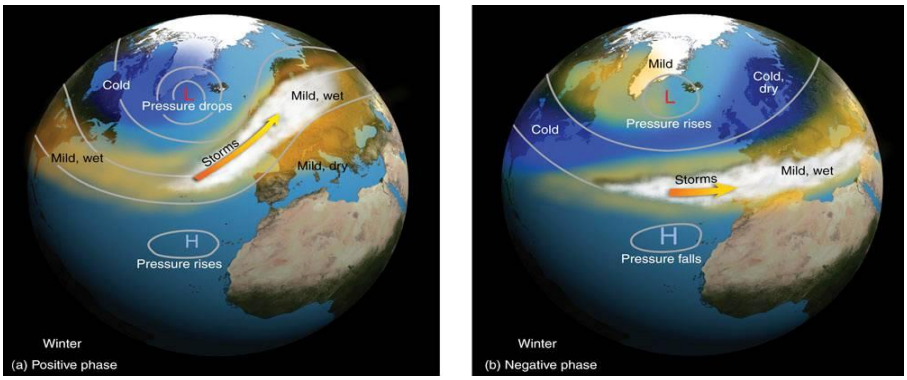


Fig. 50 : - Les deux phases de l'indice NAO (Scisnack, 2013).

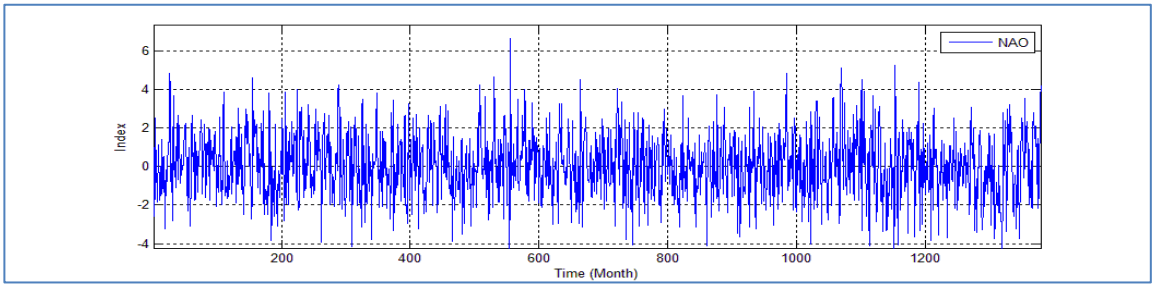


Fig. 51 : - Indice d'oscillation Nord-Atlantique 1901-2015

- **Oscillation Méditerranéenne de l'Ouest (WeMO)**

L'Oscillation de la Méditerranée Occidentale WeMO pour *Western Méditerranean Oscillation* n'est définie que dans le cadre du bassin occidental de la Méditerranée et de ses alentours. L'Oscillation de la Méditerranée Occidentale (WeMO) a été définie la première fois par Martin vide et Lopez Bustin en 2006 (Fig. 52), son indice est définit comme étant la différence des pressions atmosphériques de surface normalisées entre les stations de San Fernando en Espagne Padoue en Italie (Fig. 53).

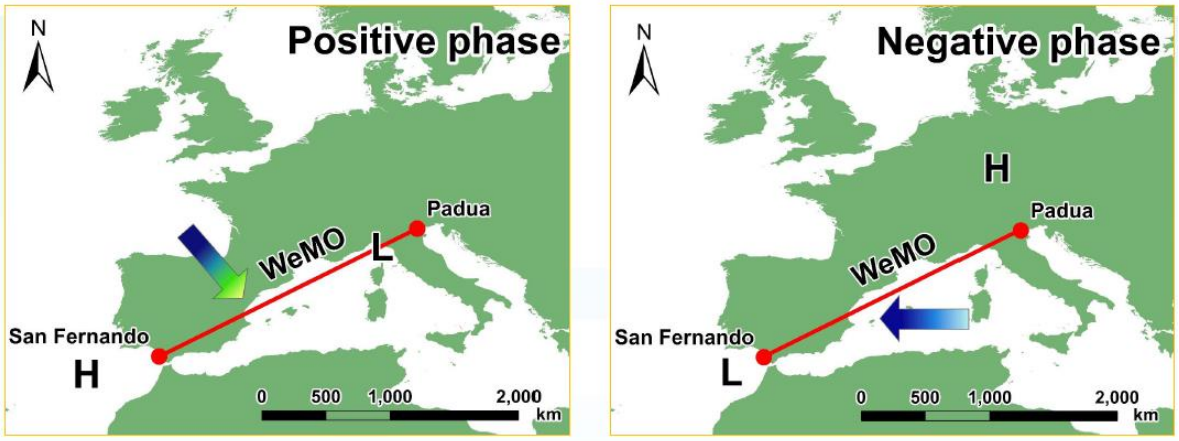


Fig. 52 : - Les deux phases de l'indice d'Oscillation Méditerranéenne Occidentale (d'après Lopez-Bustins et al., 2017)

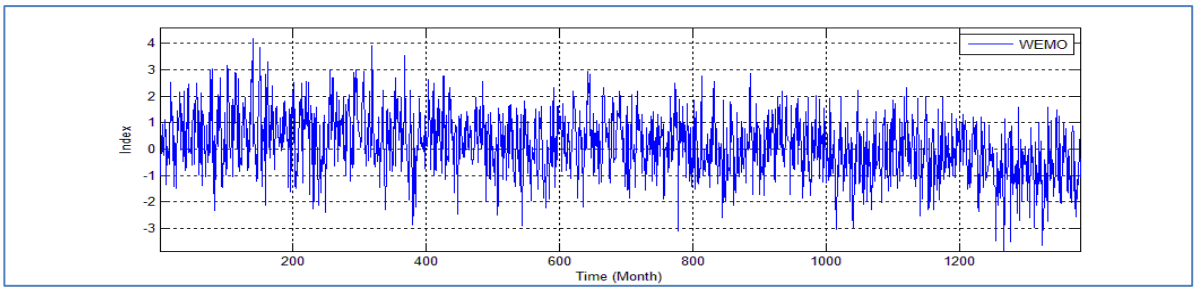


Fig. 53 : - Indice d'Oscillation Méditerranéenne Occidentale 1901-2015.

- **L'Oscillation Méditerranéenne**

L'Oscillation Méditerranéenne (MO pour *Mediterranean Oscillation*), a été défini par Conte et al. (1989) et par Palutikof et al. (1996).

Elle correspond au comportement opposé de la dynamique atmosphérique entre les sous bassins Est et Ouest de la Méditerranée. L'amplitude du bassin méditerranéen favorise la présence d'un comportement synchronisé mais opposé de la dynamique atmosphérique entre ses sous-bassins Est et Ouest. L'Indice de l'Oscillation Méditerranéenne est calculé comme étant la différence des pressions normalisée entre Alger et Le Caire en Egypte (Fig. 54). Nous disposons d'une série de valeurs journalières (Fig. 55) qui s'étend du 01 Janvier 1950 au 30 Septembre 2009 de la *Climatic Research Unit* (University of East Anglia) :(Crudata)

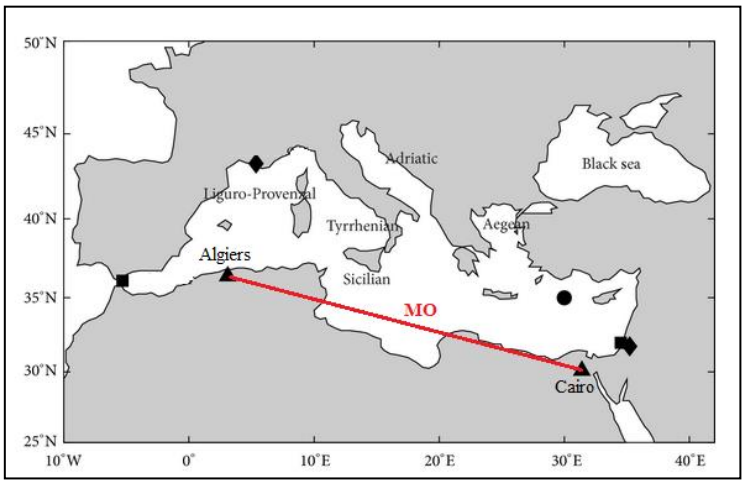


Fig. 54 : - MO : l'indice d'Oscillation Méditerranéenne est calculé comme la différence des pressions normalisée entre Alger et Le Caire.

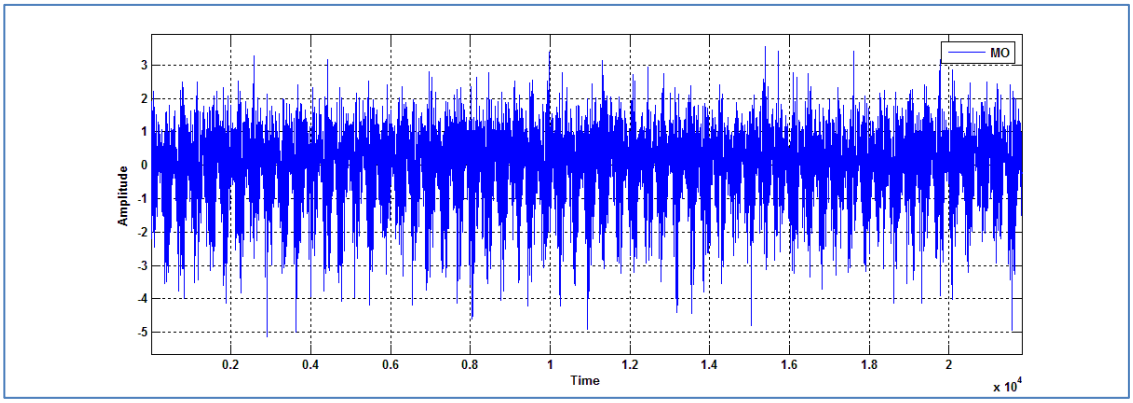


Fig. 55 : - Indice d'oscillation Méditerranéenne journalier : 1950-2009.

III.3.3. Débits journaliers

Trois bassins versants ont été pris en considération dans ce travail : Cheliff, Isser et Soummam (Fig. 56). Les deux tableaux ci-dessous donnent les principales caractéristiques des bassins et les caractéristiques statistiques des débits. On note cependant, que pour les séries de débit journalier, on a retenue qu'une partie des séries dont les périodes ne présentant aucune lacune d'observation. La figure 57 visualise la répartition temporelle des débits.

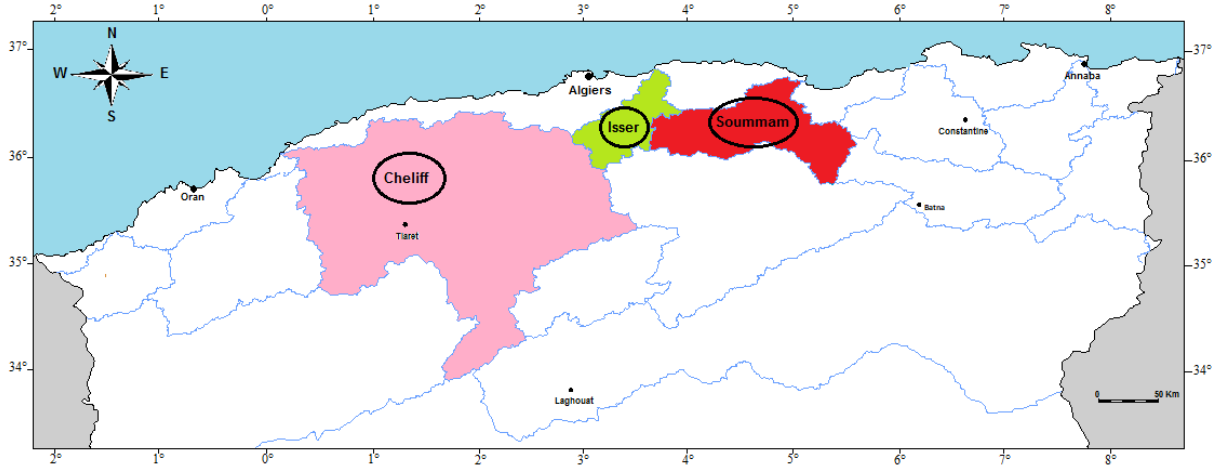


Fig. 56 : - Les bassins versants : Cheliff, Isser et Soummam.

Bassins	Périodes	Type	Superficies Km²	Sources
Cheliff	1978-2001	Journalier	43750.0	ANRH
Isser	1979-1999	Journalier	1118.0	
Soummam	1967-2005	Journalier	9125.0	

Tableau 03 : Bassins versants et sources des données de pluies

Bassins	Débit moyen (m^3/s)	Débit min (m^3/s)	Débit max (m^3/s)	Ecart-type Σ	Coeff. de Variation Cv
Cheliff	15.60	0	687.83	39.6377	2.5404
Isser	4.92	0	352.80	13.0064	2.6407
Soummam	18.51	0	1422.9	47.5504	2.5678

Tableau 04 : Caractéristiques statistiques des débits journaliers des bassins : Cheliff, Isser et Soummam.

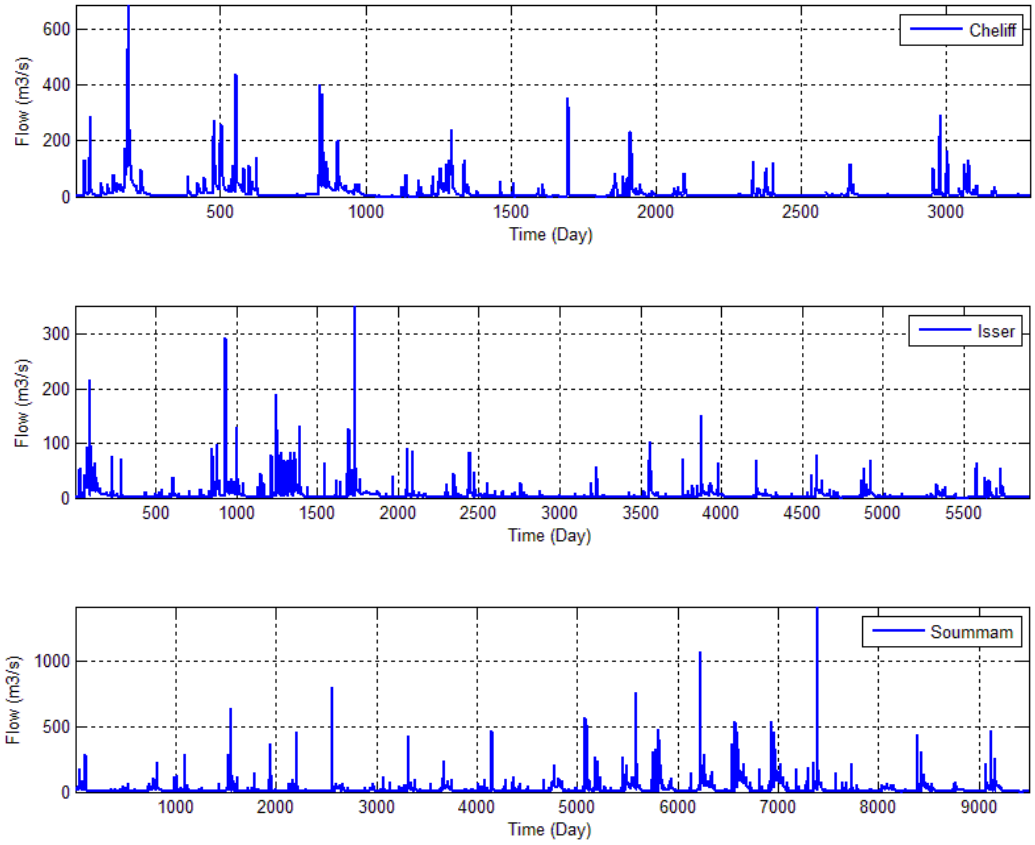


Fig. 57 : - Débits journaliers des bassins (Cheliff, Isser et Soummam).

III.4. Prévision Hydrologique

Dans ce paragraphe, nous présenterons les résultats obtenus à l'aide des deux codes de calcul des pluies mensuelles et des débits journaliers ainsi que les indices d'oscillation climatiques. Le choix de ces variables a été dicté par des travaux antérieurs. Pour les pluies mensuelles, elles ont une certaine périodicité due à la présence de saisonnalité en plus d'un phénomène aléatoire. Pour les débits journaliers, leur choix est à cause de leur déterminisme et à cause de leur non-stationnarité. Pour les indices d'oscillation climatiques, leur choix est surtout pour leur complexité.

III.4.1. Précipitations Mensuelles

- **Annaba**

La prévision des pluies d'Annaba a été réalisée par le premier code de calcul à une seule couche LSTM. Les meilleurs résultats sont obtenus pour un numHiddenUnits égal à 100. Les principaux résultats sont illustrés par les figures 58, 59,60.

- Le processus d'apprentissage (Fig. 58) montre un gradient de décente faible mais significatif pour l'RMSE et le LOSS, il présente quelques instabilités entre 200 et 250 itérations.
- La comparaison des valeurs observées et des valeurs simulées (Fig. 59), montre que le modèle restitue la périodicité du processus des pluies mensuelles, mais n'arrive pas restituer convenablement les pluies maximales . L'erreur sur ces valeurs maximales est la plus importante.

- Le coefficient de corrélation r entre les valeurs observées et les valeurs simulées est de 0.6217, une valeur significative mais relativement faible (Fig. 60).

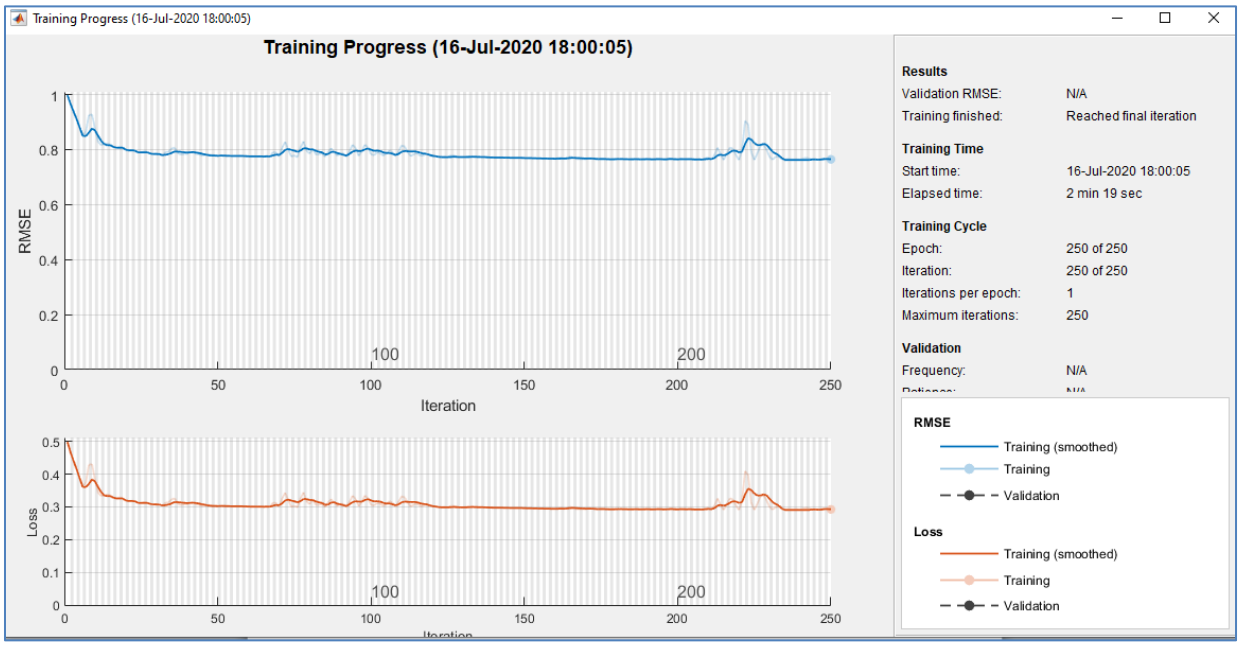


Fig. 58 : - Processus d'apprentissage des pluies mensuelles d'Annaba.

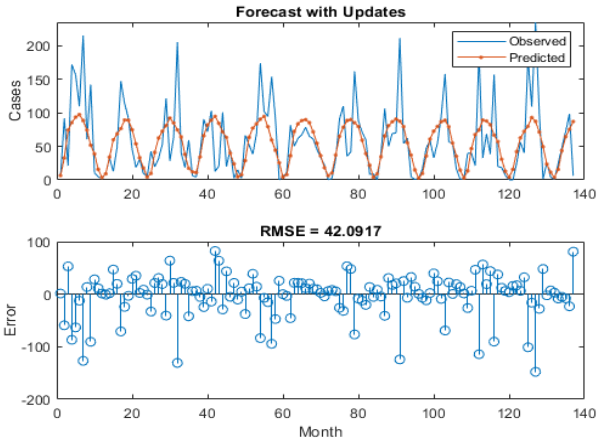


Fig. 59 : - Prévision et comparaison des valeurs observées et simulées (Annaba).

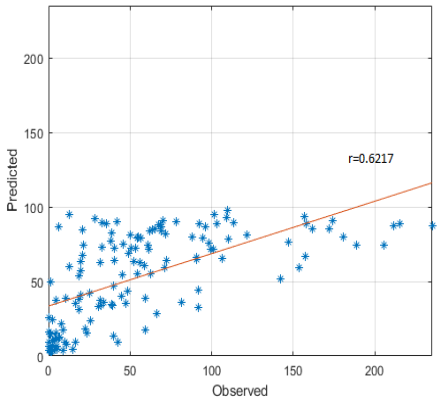


Fig. 60 : - Coefficient de corrélation entre les valeurs observées et simulées (Annaba).

- **Dar El Beida**

L'analyse des pluies de Dar El Beida à l'aide du premier code pour un numHiddenUnits égal à 1000 a pris énormément de temps pour les calculs.

- Le processus d'apprentissage (Fig. 61) montre une stabilité pour les 15 premières itérations puis un gradient faible peu significatif.
- Le modèle arrive à restituer moyennement les pluies mais pas les valeurs maximales comme dans le cas d'Annaba.
- Le coefficient de corrélation entre les valeurs observées et simulées est faible mais significatif.

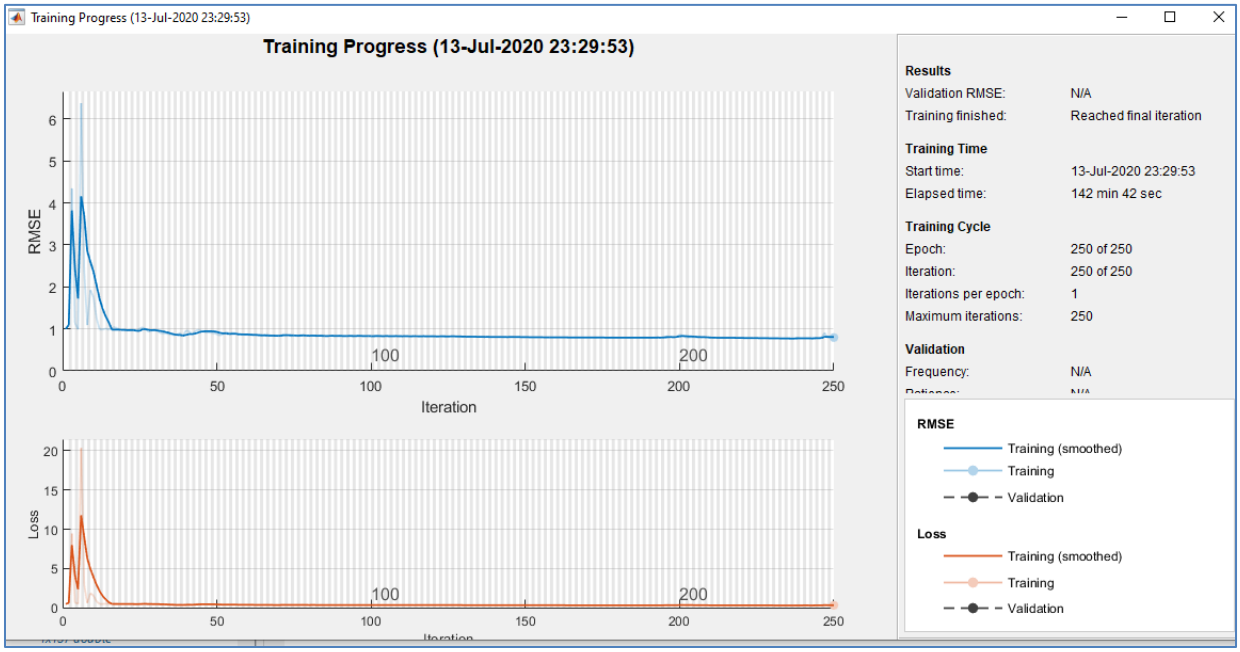


Fig. 61 : - Processus d'apprentissage des pluies mensuelles de Dar El Beida

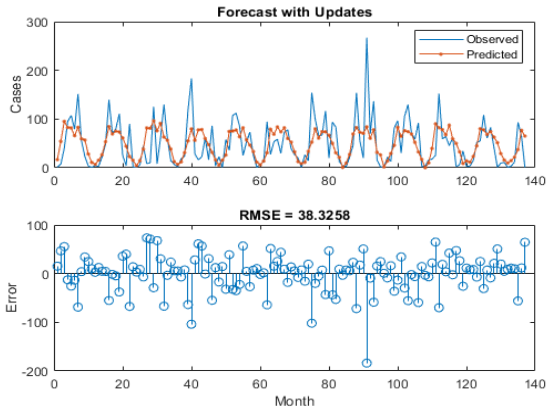


Fig. 62 : - Prévision et comparaison des valeurs observées et simulées (Dar El Beida).

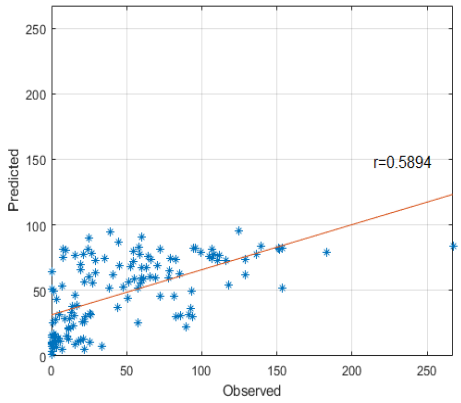


Fig. 63 : - Coefficient de corrélation entre les valeurs observées et simulées (Dar El Beida).

- **Oran**

Les simulations de prévision des pluies enregistrées à Oran à l'aide du premier code à une couche LSTM et pour un numHiddenUnits égal à 300 donnent les résultats des figures suivantes :

- Le processus d'apprentissage (Fig. 64) montre un gradient de décente faible mais significatif d'une légère stabilité et d'une convergence pour l'RMSE et le LOSS, les plus faibles valeurs de toutes les simulations effectuées.
- La prévision des pluies (Fig. 65) semble meilleure que les deux précédentes, la correspondance entre valeurs observées et valeurs simulées est assez acceptable même si les valeurs maximales sont toujours mal restituées.
- Dans ce cas, le coefficient de corrélation est de 0.6456 plus significatif que les deux précédents.

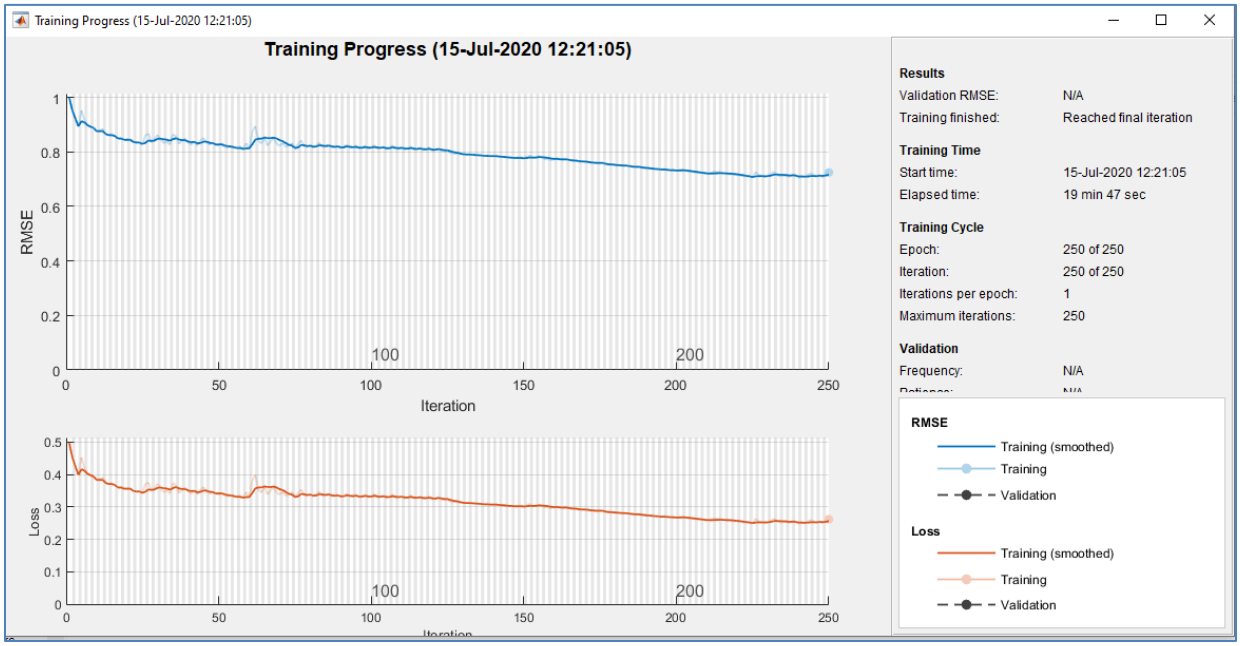


Fig. 64 : - Processus d'apprentissage des pluies mensuelles d'Oran.

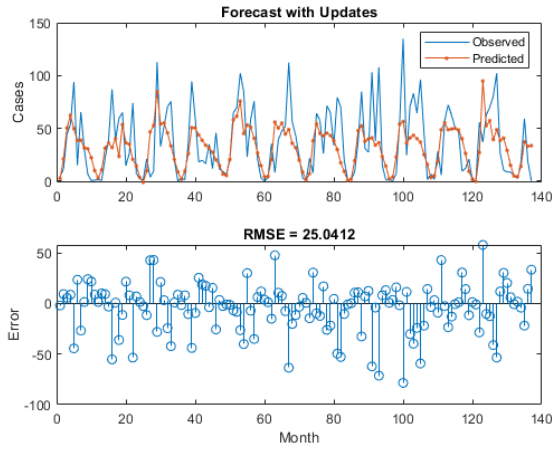


Fig. 65 : - Prévision et comparaison des valeurs observées et simulées (Oran).

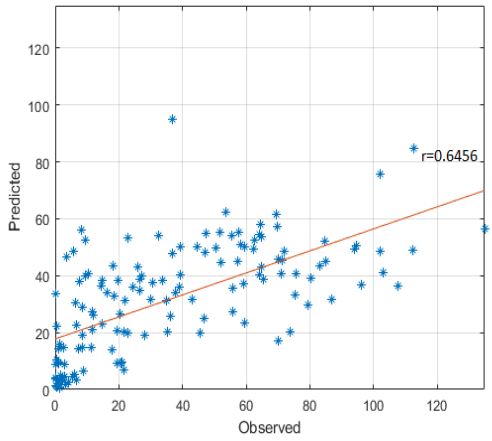


Fig. 66 : - Coefficient de corrélation entre les valeurs observées et simulées (Oran).

III.4.2. Indices Climatiques Mensuels

- **NAO**

La prévision de l'indice du NAO au pas mensuel a été réalisée par le deuxième code de calcul à deux couches LSTM. Les meilleurs résultats sont obtenus pour un numHiddenUnits égal à 70. Les principaux résultats sont présentés par les figures suivantes :

- Le processus d'apprentissage (Fig. 67) montre aucune décente de gradient c'est-à-dire aucun apprentissage qui pourrait réduire le RMSE et le LOSS.
- La prévision (Fig. 68) semble être un simple lissage de la série chronologique.
- Le coefficient de corrélation entre les valeurs observées et simulées est pratiquement insignifiant (r=0.342), les valeurs prévues simulées par le modèles sont presque nulles.
- Ces résultats que nous qualifiant de médiocres sont dus caractère aléatoire du NAO au pas mensuel. Le NAO à cette échelle se comporte comme la série aléatoire décrite

précédemment. Nous présenterons plus loin, une analyse du NAO au pas journalier pour voir la différence.

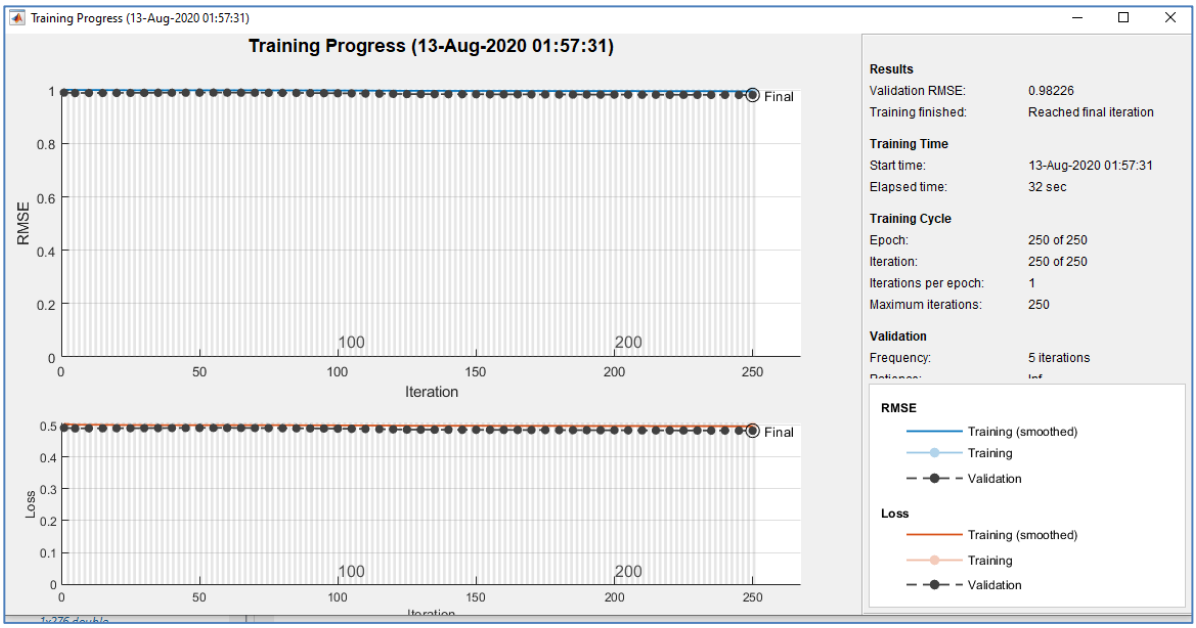


Fig. 67 : - Processus d'apprentissage du NAO au pas mensuel.

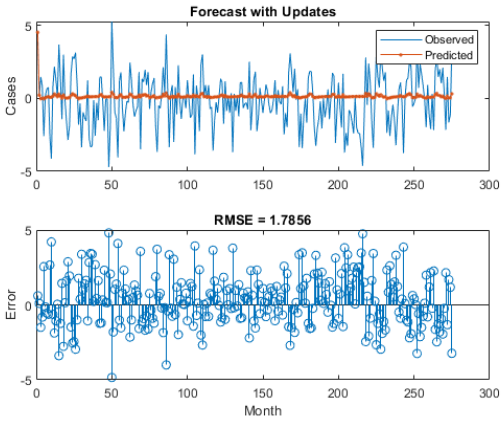


Fig. 68 : - Prévision et comparaison des valeurs observées et simulées (NAO - mensuel).

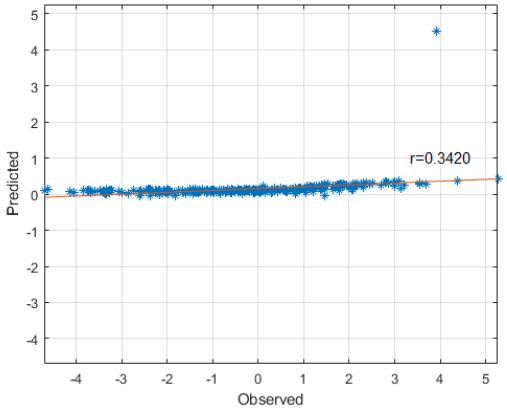


Fig. 69 : - Coefficient de corrélation entre les valeurs observées et simulées (NAO - mensuel).

- **WeMO**

La prévision de l'indice du WeMO également au pas mensuel a été réalisée toujours par le deuxième code de calcul à deux couches LSTM. Pour plusieurs valeurs du numHiddenUnits allant de 1 à 1000, les meilleurs résultats sont obtenus pour un numHiddenUnits égal à 5. Les principaux résultats sont présentés par les figures suivantes 70, 71 et 72.

- Le processus d'apprentissage (Fig. 70) n'est pas vraiment significatif jusqu'à 250 itérations, la réduction du RMSE et du LOSS ne semble pas être appréciable comme dans le cas du NAO à l'échelle mensuelle.
- La prévision (Fig. 71), malgré quelques petites variabilités reste stationnaire lissant la série chronologique et fluctue légèrement autour de zéro.

- Le coefficient de corrélation est faible pour une valeur de 0.5292 montrant la mauvaise corrélation entre les valeurs observées et les valeurs simulées.
- Comme dans le cas du NAO au pas mensuel, la chronique du WeMO ne comporte aucune structure ni régularité ni règle de prédiction identifiable comme dans le cas du processus aléatoire.

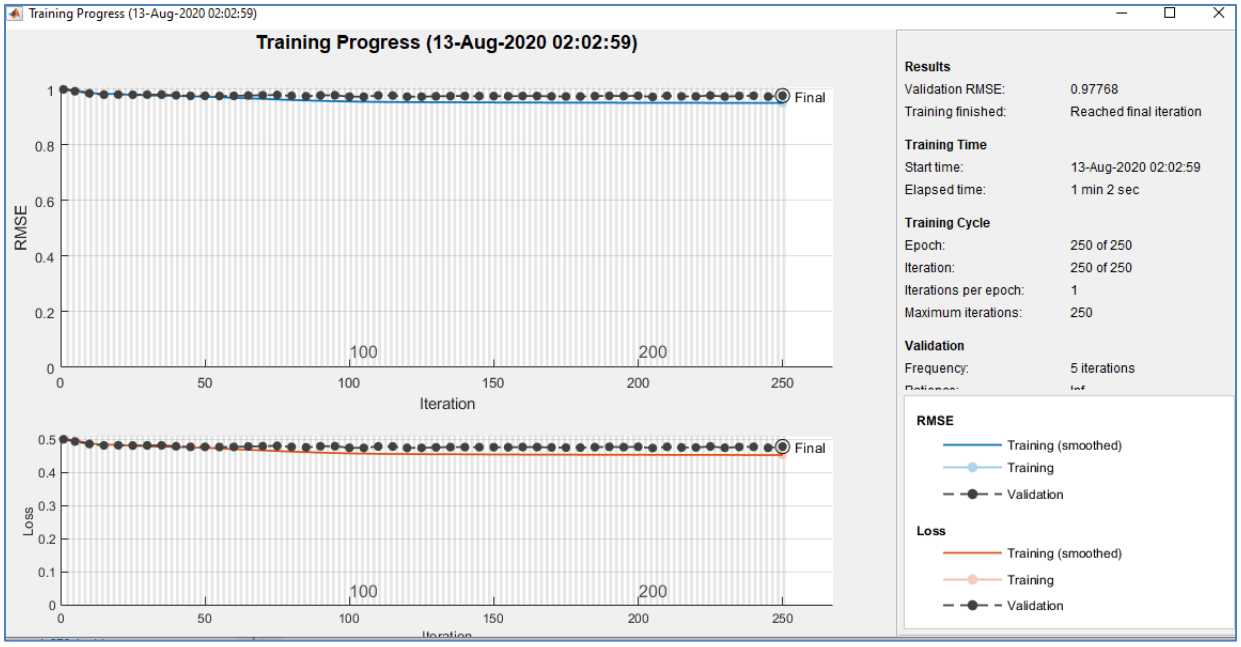


Fig. 70 : - Processus d'apprentissage du WeMO au pas mensuel.

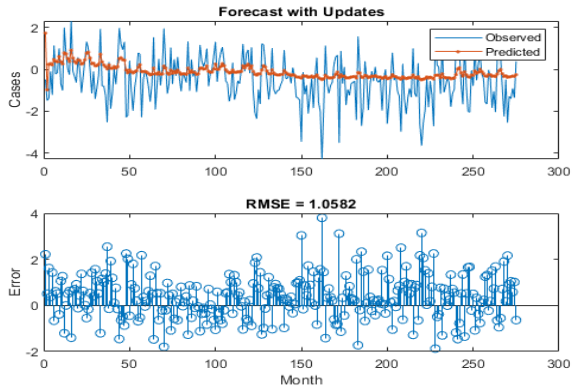


Fig. 71 : - Prévision et comparaison des valeurs observées et simulées (WeMO - mensuel).

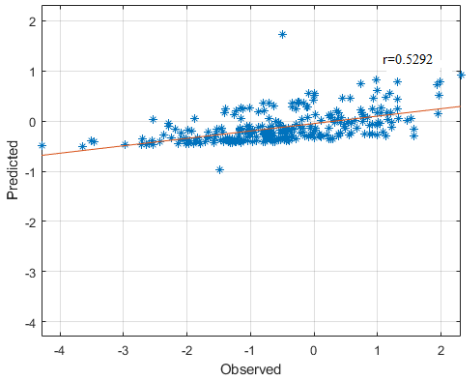


Fig. 72 : - Coefficient de corrélation entre les valeurs observées et simulées (WeMO - mensuel).

III.4.3. Indices Climatiques Journaliers

- **NAO**

La série chronologique de l'indice du NAO au pas mensuel ne comporte aucune structure ni régularité, sa prévision à l'aide de notre modèle n'a pas été satisfaisante. Nous avons préféré reprendre le travail mais au pas journalier. A ce titre, nous disposons de la série du NAO au pas journalier sur une période s'étalant du premier Janvier 1950 au 30 Septembre 2009. La prévision de l'indice du NAO au pas journalier a été réalisée cette fois par le deuxième code de calcul à deux couches LSTM et pour un numHiddenUnits égal à 10. Les résultats de la simulation de prévision

sont nettement meilleurs et d'une très haute performance relativement au cas du NAO au pas mensuel. Ils sont présentés par les figures suivantes :

- Le processus d'apprentissage montre une descente nette et une très bonne convergence matérialisée par une baisse de l'RMSE et de LOSS.
- La prévision est très bien illustrée par les figures 74 et 75 où nous constatons une très bonne restitution de l'indice.

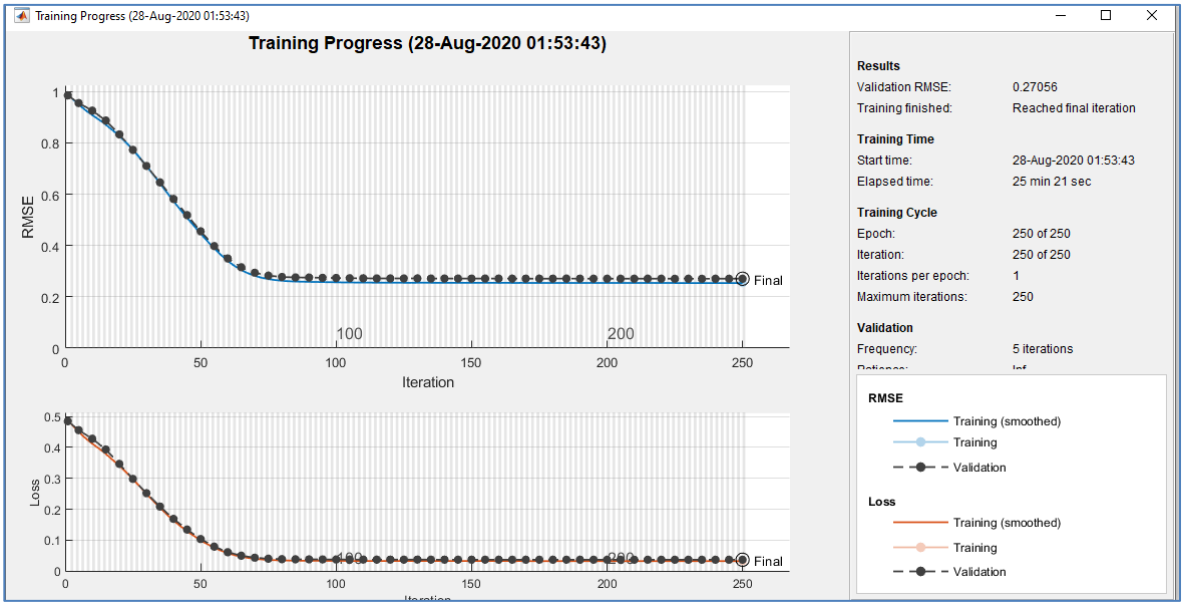


Fig. 73 : - Processus d'apprentissage du NAO au pas journalier.

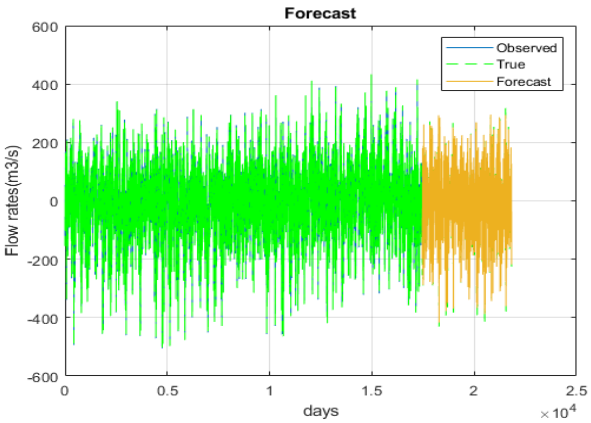


Fig. 74 : - Série chronologique : apprentissage et prévision (NAO - Journalier).

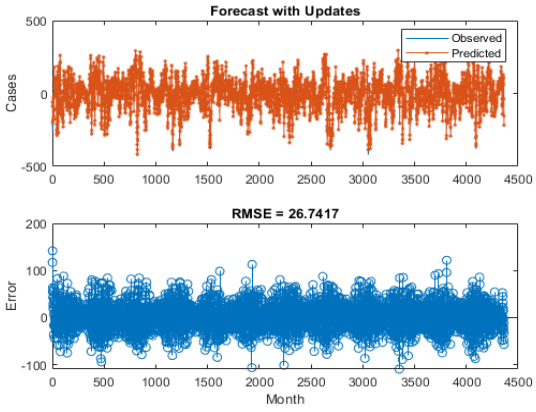


Fig. 75 : - Prévision et comparaison des valeurs observées et simulées (NAO - Journalier).

- La valeur forte du coefficient de corrélation (r=0.9706) indique vraisemblablement les résultats très satisfaisants et encourageants du modèle.

Afin de mieux visualiser les résultats, nous avons zoomé sur les 3000 dernières valeurs de la série du NAO au pas journalier. Les résultats de la simulation pour le même numHiddenUnits sont présentés ci-dessous (Fig. 77, 78, 79 et 80).

Nous constatons la haute performance du modèle à restituer les valeurs observées.

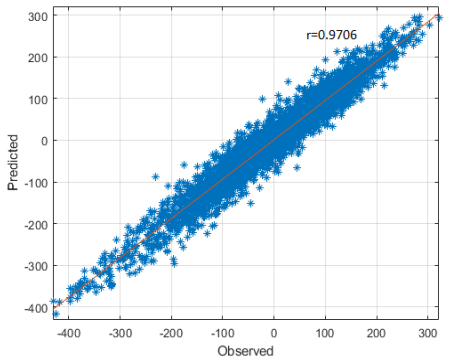


Fig. 76 : - Coefficient de corrélation entre les valeurs observées et simulées (NAO - Journalier).

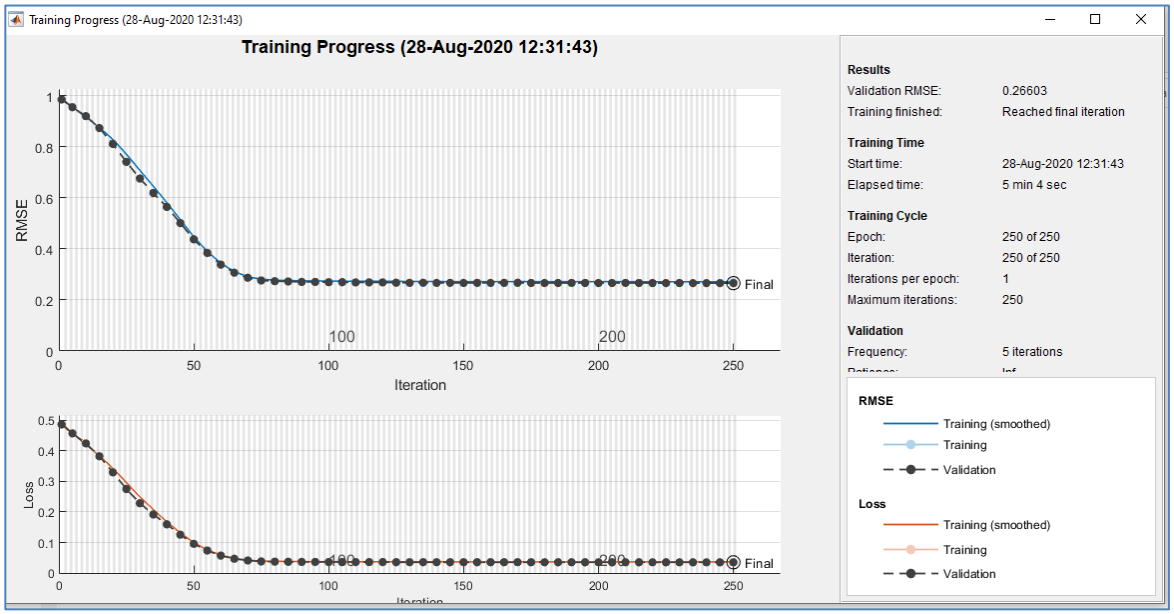


Fig. 77 : - Processus d'apprentissage du NAO au pas journalier (3000 valeurs).

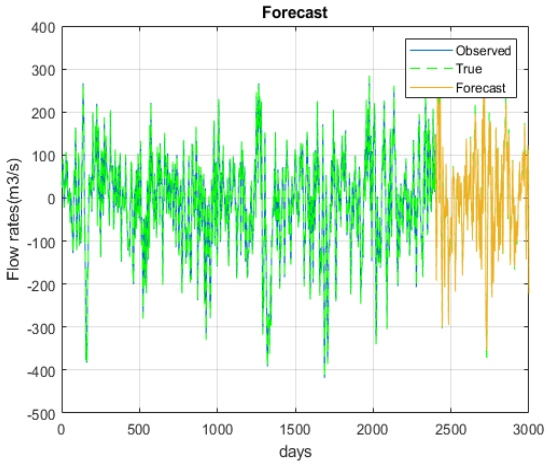


Fig. 78 : - Série chronologique : apprentissage et prévision (NAO – Journalier, 3000 valeurs).

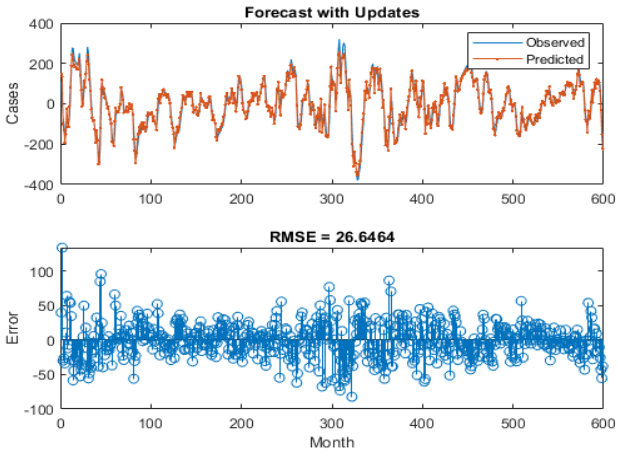


Fig. 79 : - Série chronologique : apprentissage et prévision (NAO – Journalier, 3000 valeurs).

Cette simulation a permis de montrer l'importance du choix du pas d'échantillonnage pour une bonne prévision ou même pour une bonne analyse de données. Au pas mensuel, l'indice du NAO perd beaucoup d'informations qui le composent et le structurent au pas journalier, au pas mensuel, il se comporte comme un processus aléatoire.

Cette simulation a permis entre autres, d'orienter le choix du modèle selon le pas d'échantillonnage des données, c'est-à-dire qu'un modèle stochastique sera mieux adapté pour les données de l'indice du NAO au pas mensuel.

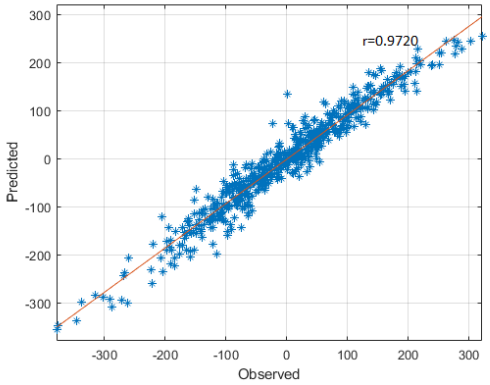


Fig. 80: - Coefficient de corrélation entre les valeurs observées et simulées (NAO – Journalier, 3000 valeurs).

- **MO**

La série chronologique de l'indice de l'Oscillation Méditerranéenne (MO) comporte près de 14000 valeurs. Une première simulation a été réalisée sur l'ensemble de la série, elle a permis d'avoir les résultats présentés par les figures 81, 82,84 et 85.

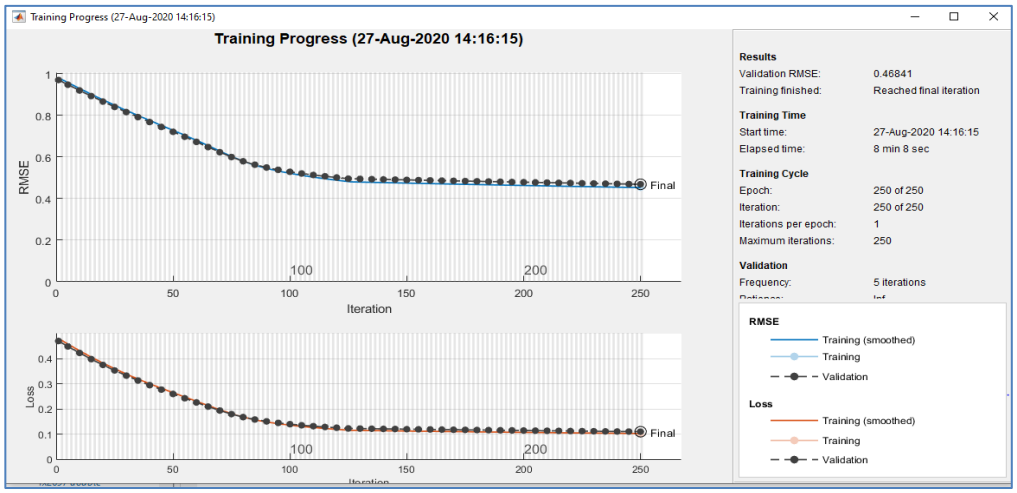


Fig. 81 : - Processus d'apprentissage du MO au pas journalier.

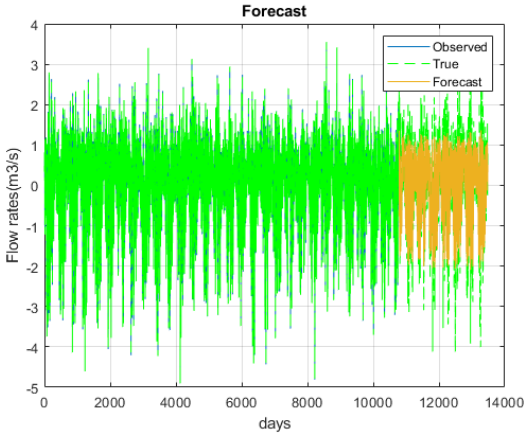


Fig. 82 : - Série chronologique : apprentissage et prévision (MO – Journalier).

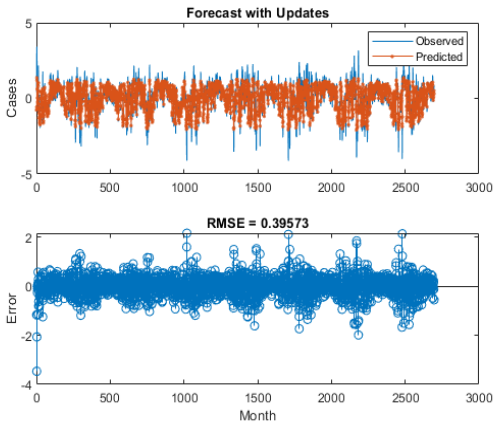


Fig. 83 : - Prévision et comparaison des valeurs observées et simulée (MO – Journalier).

- Comme dans le cas du NAO au pas journalier, le processus d'apprentissage montre une descente régulière et nettement visible de l'RMSE et du LOSS.

- La prévision est très bien restituée, seules les valeurs extrêmes sont peu sous-estimées.

- Le coefficient de corrélation fort (0.9141) indique bien le résultat satisfaisant.

Dans ce cas également, pour visualiser convenablement les valeurs prévues par le modèle, nous avons choisi les 3000 premières valeurs de la série et numHiddenUnits égal à 1. Les résultats sont résumés par les figures 85, 86, 87 et 88.

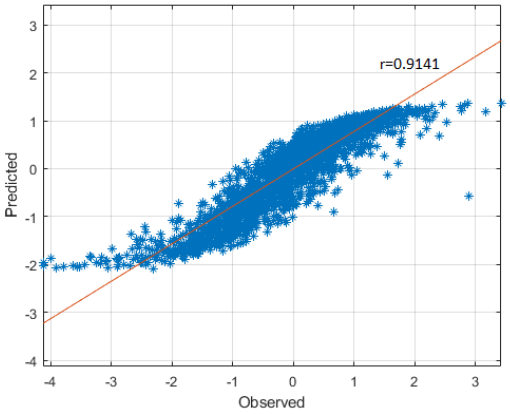


Fig. 84 : - Coefficient de corrélation entre les valeurs observées et simulées (MO – Journalier).

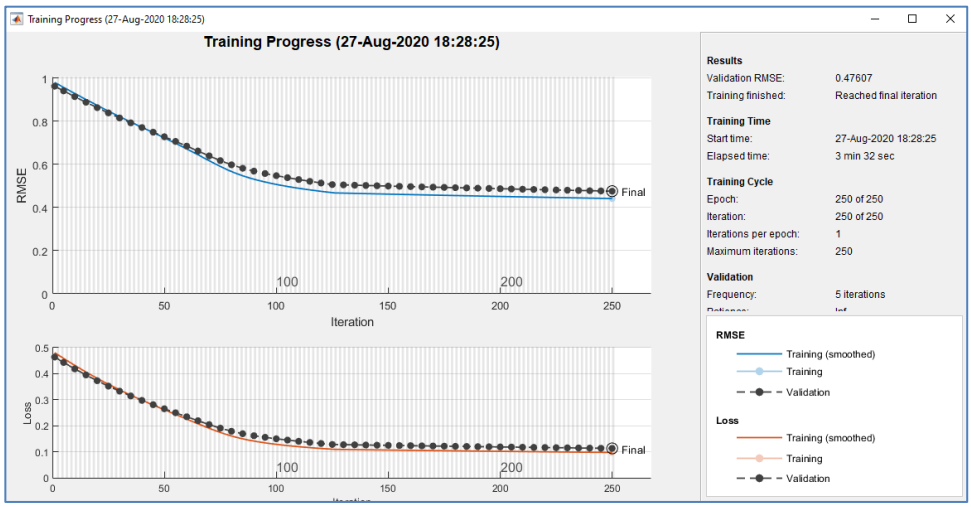


Fig. 85 : - Processus d'apprentissage du MO au pas journalier (3000 valeurs).

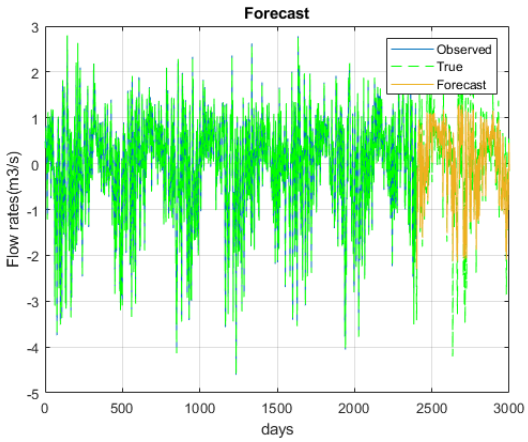


Fig. 86 : - Série chronologique : apprentissage et prévision (MO – Journalier, 3000 valeurs).

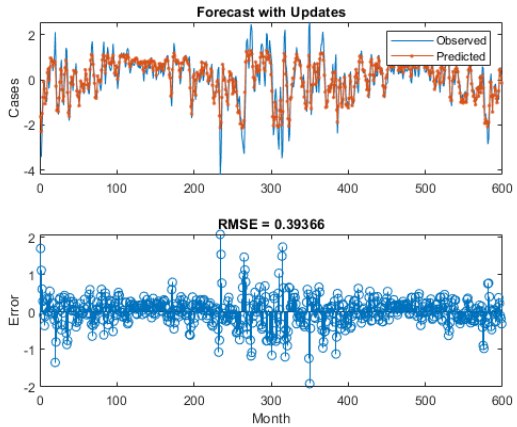


Fig. 87 : - Prévision et comparaison des valeurs observées et simulées (MO – Journalier, 3000 valeurs).

Les résultats montrent la bonne correspondance des valeurs observées et simulées et le bon coefficient de corrélation (0.9134) qui témoigne de la performance du modèle pour la prévision de la série. En définitive, le bon choix du pas d'échantillonnage et le bon choix des paramètres du modèle ont tous contribué à donner des résultats aussi performants.

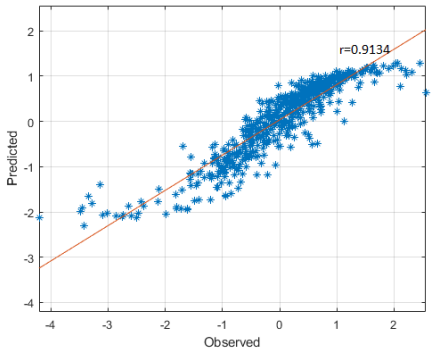


Fig. 88 : - Coefficient de corrélation entre les valeurs observées et simulées (MO – Journalier, 3000 valeurs).

III.4.4. Débits Journaliers

- **Cheliff**

Les débits journaliers du bassin du Cheliff sont mesurés au niveau de la station de Sidi Belattar dans le bas Cheliff codée 13602 par l'ANRH. La prévision de ces débits a été réalisée par le deuxième code de calcul avec deux couches LSTM et pour un numHiddenUnits égal à 15. Les principales étapes de calcul des différentes couches de la ***sequenceinput*** jusqu'à la ***regressionoutput*** sont présentées dans la figure ci-dessous (Fig. 89).

- La figure 90 montre le processus d'apprentissage et celui de la validation. Nous constatons un gradient de descente jusqu'à l'itération 50. Après cette phase de fort gradient, une phase plus stable témoigne de la bonne convergence.
- La figure 91 montre les deux parties de la série des débits, la première partie utilisée pour l'apprentissage et la deuxième partie utilisée pour la prévision.

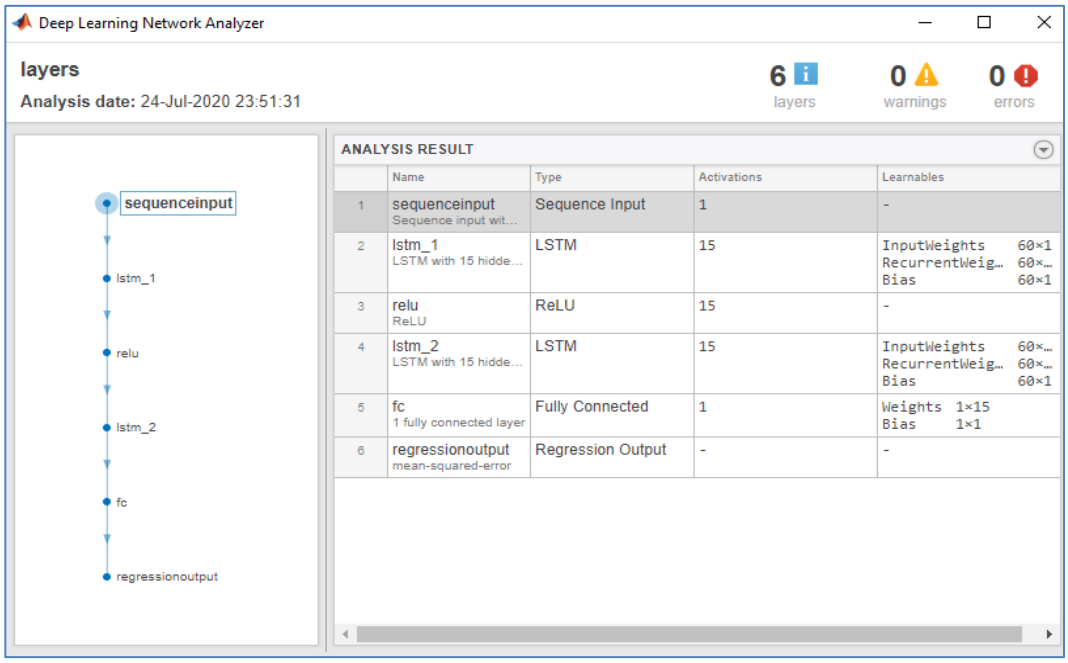


Fig. 89 : - La succession des couches de calcul et leurs caractéristiques.

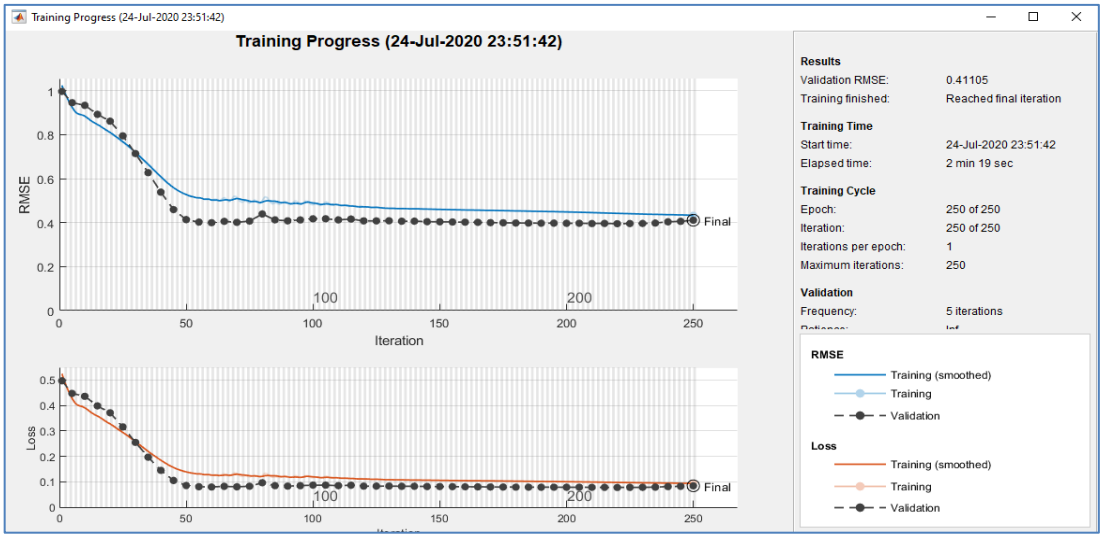


Fig. 90 : - Processus d'apprentissage des débits journaliers du bassin du Cheliff au niveau de la station de Sidi Belattar.

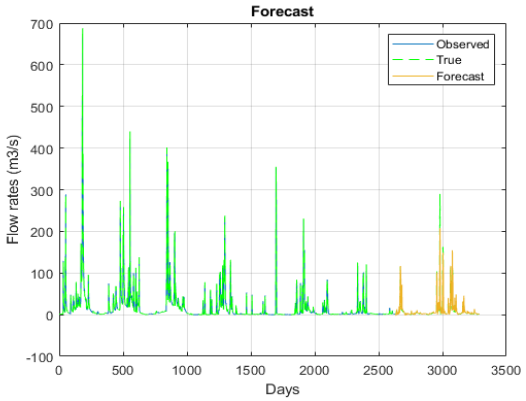


Fig. 91 : - Série chronologique : apprentissage et prévision (débits journaliers - Cheliff).

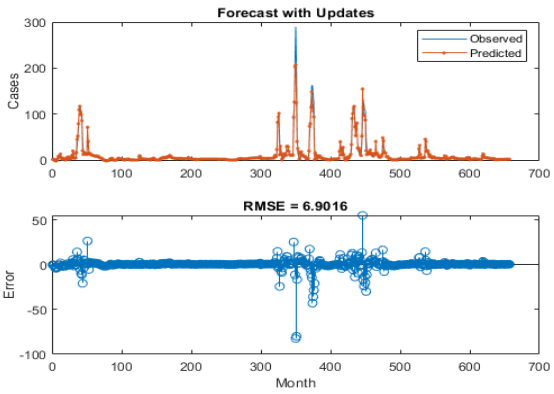


Fig. 92 : - Prévision et comparaison des valeurs observées et simulées (débits journaliers - Cheliff).

- La figure 92 donne un aperçu zoomé sur la deuxième partie de la série et sa prévision à l'aide du modèle et l'erreur entre valeurs observées et simulées. Nous constatons presqu'une parfaite correspondance, seules quelques valeurs maximales sont légèrement sous-estimées.
- Le coefficient de corrélation (Fig. 93) entre les valeurs observées et simulées est assez fort (0.9703) pour justifier le bon résultat obtenu et montre ainsi la bonne performance du Deep Learning dans la prévision des débits journaliers.

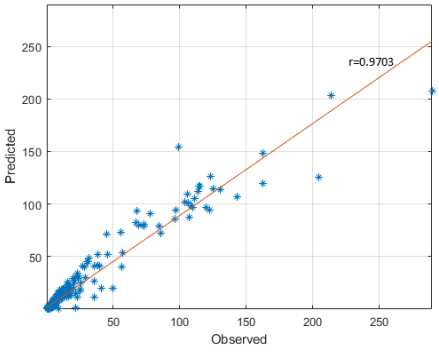


Fig. 93 : - Coefficient de corrélation entre les valeurs observées et simulées (Débits journaliers - Cheliff).

- **Isser**

Les débits journaliers du bassin de l'Oued Isser sont mesurés au niveau de la station de Lakhdaria codée 090501 par l'ANRH. La prévision des débits journaliers a été réalisée par la version du code à deux couches LSTM et numHiddenUnits égal à 50.

Les résultats obtenus sont présentés par les figures 94, 95, 96 et 97.

- Le processus d'apprentissage (Fig. 94) montre une descente de gradient assez forte pour les premières 25 itérations, le processus est suivie d'une descente de gradient plus faible mais très significative. Cependant, un phénomène de surapprentissage (***overfitting***) apparait après les 100 premières itérations, ceci arrive souvent quand l'erreur de validation augmente alors que l'erreur d'apprentissage continue à diminuer.
- La prévision des débits est très bien illustrée par les figures 95 et 96. Le modèle restitue d'une façon satisfaisante les débits observés. Cependant, là aussi, les valeurs maximales sont sous-estimées par le modèle.
- Le coefficient de corrélation entre les débits observés et simulés est fort (0.9391), il reflète le bon résultat de prévision.

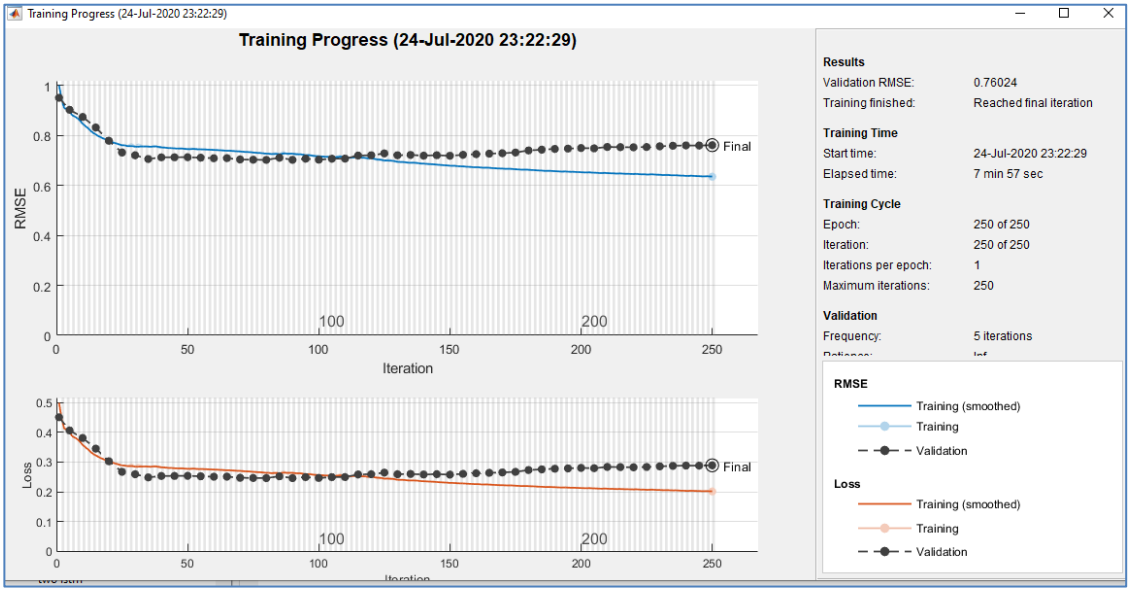


Fig. 94 : - Processus d'apprentissage des débits journaliers du bassin d'Isser.

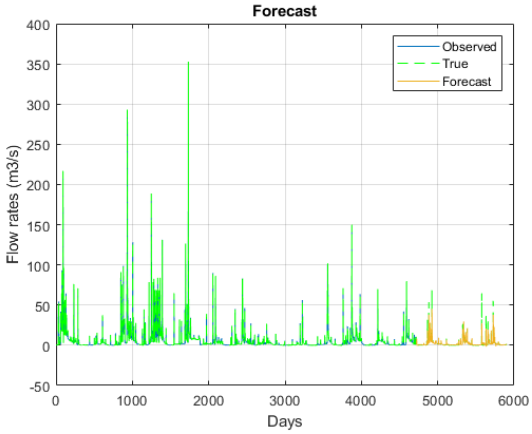


Fig. 95 : - Série chronologique : apprentissage et prévision (débits journaliers - Isser).

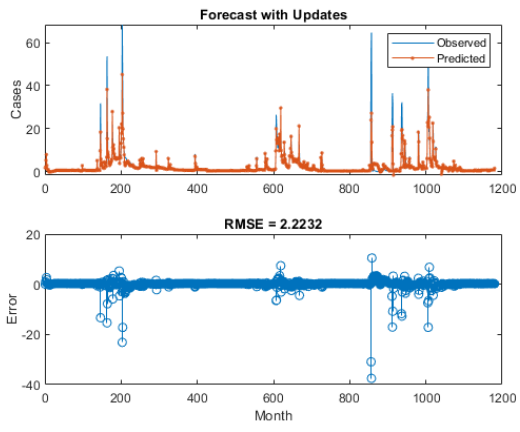


Fig. 96 : Prévision et comparaison des valeurs observées et simulées (débits journaliers - Isser).

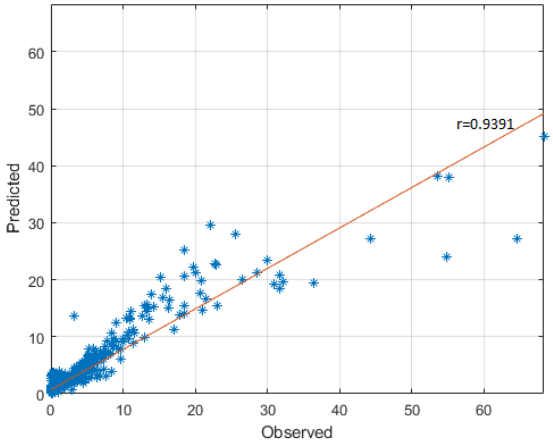


Fig. 97 : - Coefficient de corrélation entre les valeurs observées et simulées (Débits journaliers - Isser).

- **Soummam**

Les débits journaliers du bassin de la Soummam sont mesurés au niveau de la station d'El Asnam codée 150204 par l'ANRH. La prévision des débits journaliers a été réalisée par la version du code à deux couches LSTM et en faisant varier le numHiddenUnits de 1 à 100. Le meilleur résultat est obtenu pour un numHiddenUnits égal à 5.

- Le processus d'apprentissage présenté par la figure 98 montre une descente régulière du gradient jusqu'à l'itération 70, puis un gradient stable jusqu'à la fin du processus.
- La prévision illustrée par les figures 99 et 100 semble être très bien estimée et concorde très bien avec les données expérimentales avec quelques erreurs ponctuelles pour les valeurs maximales.
- Le coefficient de corrélation assez fort confirme une fois de plus l'efficacité du modèle à la prévision des débits journaliers.

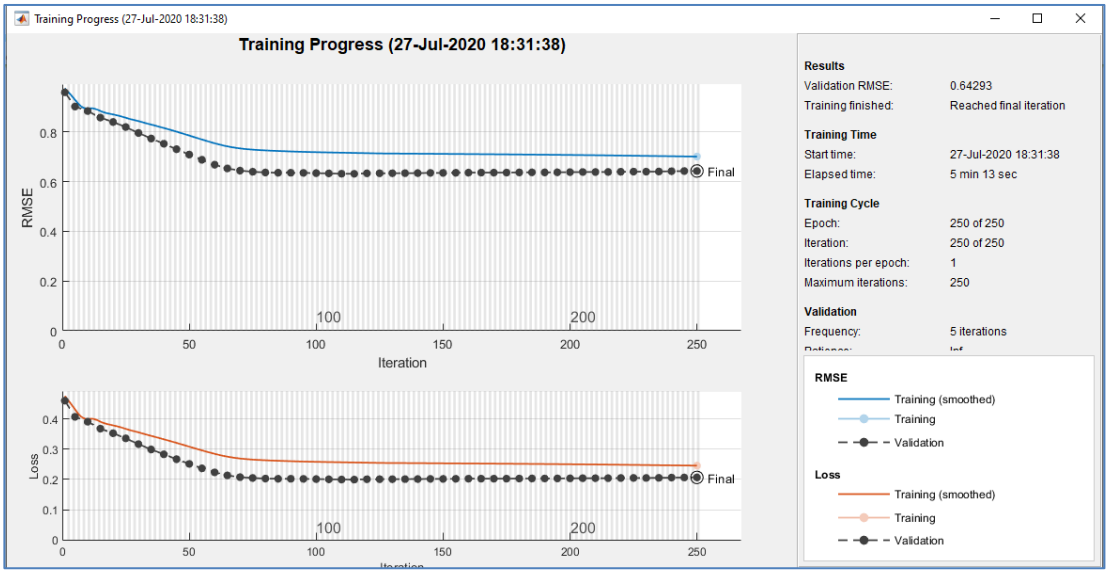


Fig. 98 : - Processus d'apprentissage des débits journaliers du bassin de la Soummam.

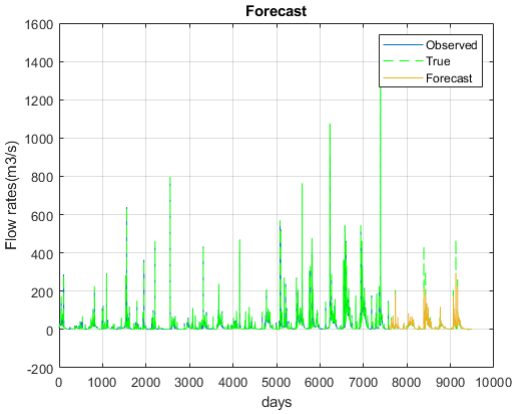


Fig. 99: - Série chronologique : apprentissage et prévision (débits journaliers - Soummam).

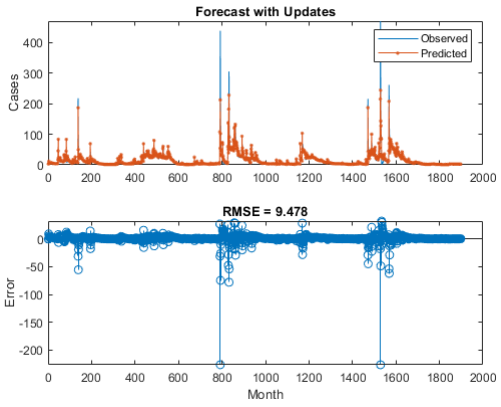


Fig. 100 : - Prévision et comparaison des valeurs observées et simulées (débits journaliers - Soummam).

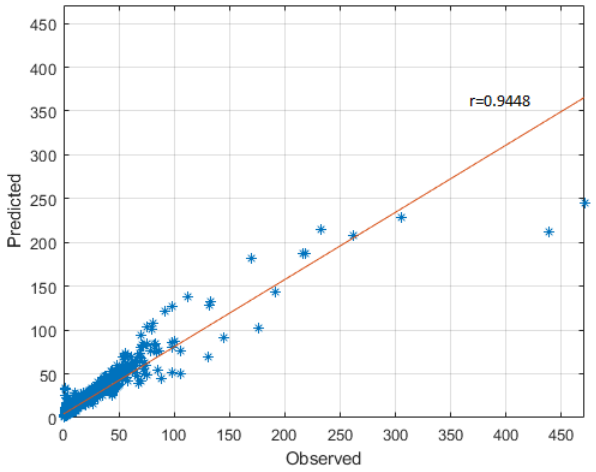


Fig. 101 : - Coefficient de corrélation entre les valeurs observées et simulées (Débits journaliers - Soummam).

III.4.5. Evaluation de la performance du modèle LSTM

Pour évaluer la performance du modèle LSTM avec ses trois options : une couche LSTM, deux couches LSTM et une couche BiLSTM, une comparaison a été réalisée avec un modèle classique à réseaux de neurones artificiels (Artificial Neural Network : ANN) et ces trois modèles LSTM.

La structure du réseau utilisé pour le modèle ANN est celui d'un Perceptron Multicouche. Le réseau comporte une couche pour capter les entrées, une couche cachée et une couche pour émettre les sorties. Nous avons choisi l'Algorithme de rétro-propagation pour l'apprentissage et utilisé la méthode de Levenberg-Marquardt. Néanmoins, le choix de l'algorithme de la rétro-propagation pour l'apprentissage du réseau impose des fonctions dérivables. A ce titre, la fonction sigmoïde a été choisie pour fonction de transfert pour les neurones des couches cachées. Pour ce modèle, nous avons utilisé la même base de données que pour le modèle LSTM.

La performance du modèle ANN est validée par des paramètres statistiques des phases d'apprentissage et de validation. Les paramètres statistiques utilisés pour ce modèle sont ceux utilisés pour le modèle LSTM pour ses trois options : l'erreur quadratique moyenne RMSE et le coefficient de corrélation *r*.

Le modèle le plus performant est obtenu à l'aide d'un nombre de neurones égal à 11 dans la couche cachée. Le tableau 05 résume les principaux résultats pour la phase de test des différents modèles.

Bassin versant	Phase de Test		
	Modèles	RMSE	*R*
Cheliff	ANN	20.512	0.7330
	LSTM	13.783	0.9017
	BiLSTM	12.173	0.9066
	Tow LSTM	**6.9016**	**0.9703**

Tableau 05 : Résultats obtenus par les Modèles : une couche LSTM, deux couches LSTM, BiLSTM et ANN pour la phase de Test.

Ces résultats montrent la bonne performance des modèles LSTM qui dépasse celle du modèle ANN. Cette performance est bien marquée pour le modèle à deux couches LSTM. Cette performance s'explique par la robustesse du modèle LSTM et la précision de ses sorties.

III.5. Conclusion

En conclusion de ce chapitre, on peut dire que la performance du Deep Learning avec les réseaux LSTM pour la prévision des séries temporelles a été mise en évidence à travers les différents exemples réalisés.

La prévision à l'aide des réseaux LSTM des débits journaliers et des indices climatiques au pas journalier a été d'une très haute performance, mais la prévision des précipitations mensuelles reste moyennement acceptable. Toutefois, les réseaux LSTM restent incapables de prévoir les séries temporelles qui ne comportent aucune structure ni régularité ni règle de prédiction identifiable. Ainsi, nous avons constaté que l'échantillonnage des séries joue un rôle important dans la prévision.

Les programmes de calcul ont prouvé la robustesse et la souplesse de la méthodologie des réseaux LSTM, mais n'arrivent à prévoir avec précision les valeurs maximales et prennent énormément de temps quand les séries sont très longues et où le nombre de blocs cachés est grand.

Conclusion Générale

La modélisation pour la prévision hydrologique constitue un outil rentable et rapide permettant de réaliser des prédictions sur le comportement du système hydrologique de manière à guider les prises de décision et faciliter la compréhension des risques et de bien planifier et convenablement gérer la ressource. Malgré le nombre important d'approches proposées pour les prévisions des séries temporelles, l'intelligence artificielle reste le domaine privilégié des hydrologues pour la prévision hydrologique.

Bien que le formalisme des réseaux d'automates a été dès les années 1940 à la base d'applications technologiques de l'intelligence artificielle, il a également permis une description semi-quantitative des systèmes complexes. L'avènement de l'informatique et l'évolution technologique de la puissance de calcul ont beaucoup facilité la formulation d'approches complexes et permis d'avoir d'autres alternatives plus efficaces.

Récemment, grâce à l'avancée des performances de calcul des ordinateurs que s'est développé le concept de Deep Learning qui est une forme d'intelligence artificielle, dérivée du Machine Learning et qui est très en vogue ces derniers temps.

L'objectif de notre travail est d'élaborer des modèles basés sur les réseaux de neurones récurrents du Deep Learning permettant la prévision des séries hydrologiques.

Ainsi, trois approches basées sur les réseaux de longue mémoire à court terme (LSTM) ont été utilisées pour les prévisions des débits journaliers, des pluies mensuelles et des indices d'oscillation océaniques et atmosphériques à grande échelle aux pas mensuels et journaliers.

Le choix des réseaux LSTM est justifié pour les séries hydrologiques en particulier les débits qui ont souvent une très longue mémoire. Ces réseaux neuronaux récurrents sont capables d'apprendre les dépendances à long terme et conservent les informations sur des périodes plus longues.

Les algorithmes décrits dans cette étude et leur implémentation sous Matlab pour les prévisions des séries temporelles ont montré une très haute performance sur des cas académiques.

La prévision des débits journaliers a montré une très haute performance où le coefficient de corrélation maximal entre les valeurs observées et simulées est de 0.97.

Pour les pluies mensuelles, la prévision semble de moyenne qualité, car le coefficient de corrélation maximal est de 0.64, il est significatif mais relativement faible. Ce résultat est vraisemblablement lié à la structure des séries de pluies malgré la présence de saisonnalité, elles renferment une part considérable de stochasticité.

Les indices d'oscillations climatiques au pas mensuel se comportent comme un processus aléatoire qui n'a aucune structure ni régularité ni règle de prédiction identifiable, sa prévision s'avère impossible. Cependant, au pas journalier, la prévision des indices semble très prometteuse et donne de très bons résultats.

En définitive, les résultats obtenus dans ce travail, s'avèrent très encourageants et très significatifs et d'un apport substantiel où la performance du Deep Learning avec des réseaux LSTM a été nettement mise en évidence relativement aux résultats obtenus par un modèle à réseaux de neurones artificiels classique.

La technique du Deep Learning avec les réseaux LSTM être utilisée comme outil alternatif pour une bonne gestion des ressources en eau et la prévision des pluies et des débits afin d'évaluer et réduire les risques d'inondations et les longues sécheresses dans les bassins hydrologiques en Algérie.

Les perspectives de ce travail sembles inépuisables, si toutefois on augmenterait la base de données par des données complémentaires ou/et intégrer aux réseaux LSTM des techniques métaheuristiques afin d'optimisation les paramètres des réseaux.

Bibliographies

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Ghemawat, S. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

Abrahart R.J., P. E. Kneale, L. M. See, Neural networks for hydrological modelling. Taylor & Francis, London, UK, 2004, p. 304.

Alan, M. (1950). Turing. *Computing machinery and intelligence. Mind*, *59*(236), 433-460.

Barber, D. P., Becker, U., Benda, H., Boehm, A., Branson, J. G., Bron, J., ... & Chen, M. (1979). Discovery of three-jet events and a test of quantum chromodynamics at petra. *Physical Review Letters*, *43*(12), 830.

Baron, G. L. (2014). Elèves, apprentissages et «numérique»: regard rétrospectif et perspectives. *Recherches en éducation*, *18*(2), 91-103.

Basdevant ,& Partners, L. (2016),*Pourquoi une étude pluridisciplinaire sur l'IA ?, Intelligence Artificielle, un nouvel horizon, societe d'edition electronique sur l'economie, la geopolitique et le droit. 119 p.*

Beale M. H., Hagan M. T. et Demuth H.B. (2019) - Deep Learning Toolbox User's Guide. The MathWorks, Inc. 1 Apple Hill Drive Natick, MA 01760-2098

Beghdad, A., & Ouserir, A. (2018). Une approche Deep Learning pour l'analyse des Sentiments Sur Twitter.

Blanche-Benveniste, C. (1994). Quelques caractéristiques grammaticales des sujets employés dans le français parlé des conversations. *M. Yaguello (éd.) Subjecthood and Subjectivity. The Status of the subject in linguistic theory. Paris/Londres: Ophrys*, 77-108.

Blog de Colah
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Broomhead, D. S., & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks* (No. RSRE-MEMO-4148). Royal Signals and Radar Establishment Malvern (United Kingdom).

Chettih, M., Mesbah, M. (2006) - Utilisation des analyses corrélatoire et spectrale pour inférer sur la structure et le comportement hydrodynamique des aquifères de l'Atlas Saharien. Vol. 17, n°2, pp 145-159.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Climat scisnack
https://www.scisnack.com/2013/11/26/to-index-or-not-to-index/

crudata
https://crudata.uea.ac.uk/cru/data/moi/moi1.output.dat

Demmak A., (1982)- Contribution à l'étude de l'érosion et des transports solides en Algérie septentrionale. Thèse de Docteur Ingénieur, Université Pierre-et-Marie-Curie, Paris, 323 p.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2015. arXiv:1412.6980v9

Dreyfus, G. (2002). Les réseaux de neurones pour la modélisation des procédés industriels: du ruban adhésif au soudage par point. *Signal processing and Machine learning laboratory (SIGMA lab), Ecole Superieure de Physique et de Chimie Industrielle (ESPCI), Paris, http://www. neurones. espci. fr*.

Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y., & Vapnik, V. (1994). Boosting and other machine learning algorithms. In *Machine Learning Proceedings 1994* (pp. 53-61). Morgan Kaufmann.

Foster, D. (2019). *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

Gülen, L. (2018). *A study of neural networks for natural language processing* (Doctoral dissertation, Haute école de gestion de Genève).

Hinton, G. (1985). Learning in parallel networks. *Byte*, *10*(4), 265-273.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554-2558.

Hu Caihong, Qiang Wu, Hui Li, Shengqi Jian, Nan Li and Zhengzheng Lou (2018) - Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation. Water 2018, 10, 1543; doi:10.3390/w10111543.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*(3), 574.

import.io
https://www.import.io/post/history-of-deep-learning/

Irsoy, O., & Cardie, C. (2014). Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems* (pp. 2096-2104).

Ismail, R., Linder, L. A., MacPherson, C. F., & Fugate Woods, N. (2016). Feasibility of an iPad application for studying menopause-related symptom clusters and women's heuristics. *Informatics for Health and Social Care*, *41*(3), 247-266.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, (4), 364-378.

Joan A. Lopez-Bustins J.A., Martin-Vide J., Arbiol-Roca L. et and Prohom M. (2017) Intraanual variability of the Western Mediterranean Oscillation (WeMO) and occurrence of extreme torrential rainfall in Catalonia (NE Iberia). *European Geosciences Union General Assembly, 2017*. Vienna, Austria 23-28 April 2017.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, *30*(10), 947-954.

Kolboom, I., & Weisenfeld, E. (Eds.). (1993). *Frankreich in Europa: ein deutsch-französischer Rundblick*. Europa Union Verlag.

Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork rnn. *arXiv preprint arXiv:1402.3511*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Labat D.(2000). *Non-linéarité et Non-stationnarité en hydrologie Karstique*.thése,université de toulouse,N°1740,220p.

Lana X., Burgueno A., Martinez M.D. and Serra C. (2016)- Complexity and predictability of the monthly Western Mediterranean Oscillation index*. Int. J. Climatol*. 36: 2435-2450.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989) . Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998) . Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lockett, A. J., & Miikkulainen, R. (2009). Temporal convolution machines for sequence learning. *Dept. Comput. Sci., Univ. Texas, Austin, Tech. Rep. AI-09-04*.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, *11*(7), 674-693.

Mathbout, S., Lopez-Bustins, J. A., Royé, D., Martin-Vide, J., Bech, J., & Rodrigo, F. S. (2018). Observed changes in daily precipitation extremes at annual timescale over the eastern Mediterranean during 1961–2012. *Pure and Applied Geophysics*, *175*(11), 3875-3890.

Mathworks
https://www.mathworks.com/discovery/lstm.html

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

medium(a)
https://medium.com/analytics-vidhya/an-introduction-to-generative-deep-learning-792e93d1c6d4

medium(b)
https://medium.com/@kartikshangle123/convolutional-recursive-deep-learning-for-3d-object-classification-b53ef2ebf3bd

Minsky, M., & Papert, S. (1969). An introduction to computational geometry. *Cambridge tiass., HIT*.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, *1*(2), 281-294.

Nayak PC, K .P. Sudheer and S. K. Jain, “Rainfall-Runoff modeling through hybrid intelligent system,” *Water Resources Research,* vol. 43 w07415, 2007, pp. 1-17.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., Dreyfus, G., & Marcos, S. (1993). Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms. *Neural computation*, *5*(2), 165-199.

Oussar, Y., & Dreyfus, G. (2000). Initialization by selection for wavelet network training. *Neurocomputing*, *34*(1-4), 131-143.

Paluszek, M., & Thomas, S.(a). (2020). Algorithmic Deep Learning. In *Practical MATLAB Deep Learning* (pp. 77-90). Apress, Berkeley, CA.

Paluszek, M., & Thomas, S.(b). (2020). *Practical MATLAB Deep Learning: A Project-Based Approach*. Apress.

Powell, M. J. (1987). Radial basis functions for multivariable interpolation: a review. *Algorithms for approximation*.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Samuel, A. L. (1952). *U.S. Patent No. 2,605,323*. Washington, DC: U.S. Patent and Trademark Office.

Scisnack
https://www.scisnack.com/2013/11/26/to-index-or-not-to-index/

Siècle Digital
https://siecledigital.fr/2019/01/30/differences-intelligence-artificielle-machine-learningdeep-learning/

Sivakumar B., R. Berndtsson, J. Olsoon, K. Jinno, “Evidence of chaos in the rainfall-runoff process,” *Hydrological Sciences Journal*, 46(1), 2001, p. 131-145.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Taouti M.B. (2012) - Analyses fractales et multifractales des précipitations et des débits en Algérie septentrionale. Mémoire de Magister en Hydraulique. Université de Laghouat, 105 p.

Tayfur G., “Soft computing in water resources engineering : Artificial Neural Networks, Fuzzy Logic and Genetic Algorithms,” Eds. WIT, 2012, p. 288.

Turing, A. M (1950.Mind.Mind,59(236), 433-460.

Turing, A. (1947). The automatic computing engine. *Lecture to the London Mathematical Society*.

Velluz, L., Le Grand, M., & Grosjean, M. (1965). *Optical circular dichroism: principles, measurements, and applications*. Verlag Chemie.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

WIKIPEDIA
https://en.wikipedia.org/wiki/Extreme_learning_machine

WILDML
http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

WORLD BANK
http://sdwebx.worldbank.org/climateportal/

Yao, Z., Wu, B., Shen, X., Cao, X., Jiang, X., Ye, Y., & He, K. (2015). On-road emission characteristics of VOCs from rural vehicles and their ozone formation potential in Beijing, China. *Atmospheric Environment*, *105*, 91-96.