International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

Vol. 42 No. 1 December 2023, pp. 265-276

Prediction De Precipitation Mensuelle Par Reseau De Neurone Recurrent

Rija Santaniaina RAKOTOARIMANANA[1], Tiana Razefania RAMAHEFY[2], Solofo RANDRIANJA[3]

[1]Ecole Doctorale GENESIS
Université d'Antsiranana, Madagascar
rijasant@yahoo.fr

[2]Université d'Antananarivo, Madagascar
r.ramahefy@gmail.com

[3]Université d'Antsiranana, Madagascar
srandrianja@gmail.com

Résumé—Cet article a l'objet de faire une « Prédiction de précipitation mensuelle par réseau de neurone récurrent ». Face au changement climatique et les effets néfastes qu'il engendre actuellement, notamment le problème climatique, cette expérience pourrait aider tout le monde à prendre de décision ou de mesures pour certaines études plus développées ou actions à entreprendre pour faire face à cette situation. La méthode utilisée pour la prédiction est le Réseau de Neurone Récurrent dont plus précisément les modèles Long Short-Term Memory (LTSM). Cette méthode est adaptée au traitement des données temporelles. Après l'opération, nous avons obtenu les prédictions durant la phase d'entrainement du modèle et du test. Les erreurs du modèle lors de l'entrainement et du test par rapport aux données originales sont aussi présentées dans cette étude.

Mots clés—Intelligence Artificielle, Réseau De Neurone Artificiels, Réseau De Neurone Récurrente, Prédiction.

7

I. INTRODUCTION

Actuellement, nous sommes tous à la recherche des moyens pour prévoir ou prédire des phénomènes comme les catastrophes naturelles, l'inflation sur le produit de première nécessité, la hausse du prix du carburant. Nombreux sont les outils cruciaux qui sont déjà inventés par des experts ou des étudiants chercheurs pour la prédiction et la prévision pour aider les décideurs nationaux ou internationaux pour la planification et l'élaboration de stratégie adaptées au futur.

L'intelligence Artificielle est à nos jours prend des places très importantes dans des sociétés sur l'organisation du travail : améliorer de la gestion des informations, la planification des activités et la coordination des acteurs.

Nombreux sont les algorithmes mais, c'est qui nous intéresse c'est le réseau de neurone qui a un large domaine d'applications dans différentes industries avec différents cas d'utilisation, du traitement du langage naturel (*NLP : Natural Language Processing*) à la vision par ordinateur.

Dans cet article, nous essayons de faire la prédiction de précipitation, de la ville de Moramanga, dans la Région d'Alaotra-Alaotra, à Madagascar, qui est un indicateur du changement climatique. Nous utilisons comme méthode pour cette prédiction le Réseau de Neurone Récurrent (RNN) : Long Short-Term Memory (LSTM) qui est plus pratique pour les données séquentielles ou chronologiques.

Corresponding Author: Rija Santaniaina RAKOTOARIMANANA

II. MATERIELS ET METHODES

A. Réseau de neurones artificiels

Un réseau de neurones artificiels, est un système inspiré du fonctionnement des neurones biologiques. Tout d'abord, dans les synapses ou dendrites, on fait la pondération de chaque élément en entrée x_i par w_i. Ensuite, dans le corps cellulaire, on fait la somme des entrées pondérées et on l'applique à une fonction d'activation f. Et enfin, dans l'axone qu'on trouve la sortie du modèle.

Le but d'un réseau de neurones est de représenter une fonction $\hat{y}(x, w)$ pour la régression ou la classification et d'apprendre les valeurs du vecteur de paramètres w et b permettant une bonne prédiction sur une base d'apprentissage donnée.

B. Architecture du réseau et Forward Propagation

Un réseau de neurone de type feedforward enchaîne différentes couches de neurones, chaque couche étant fonction de la précédente. Si le vecteur d'entrée est $x \in \mathbb{R}^p$, la première couche effectue d'abord une opération linéaire ou affine sur x, puis elle calcule une fonction non linéaire du résultat :

$$z^{[1]} = W^{[1]}x + b^{[1]} \text{ puis } a^{[1]} = g^{[1]}\left(z^{[1]}\right) \tag{1}$$

Avec $W^{[i]}$ une première matrice de poids de taille $n^{[2]} \times p$, $b^{[1]}$un vecteur d'offset de taille $n^{[2]} \times 1$ et $g^{[1]}$une fonction d'activation, généralement non linéaire (lorsqu'on écrit $g^{[1]}(z)$ avec z un vecteur, cela signifie que $g^{[1]}$est appliquée à toutes les coordonnées de z). La k^{ème} couche s'écrit alors en fonction de la précédente comme :

$$a^{[k]} = g^{[k]}\left(W^{[k]}a^{[k-1]} + b^{k}\right) \tag{2}$$

$$z^{[k]} = W^{[k]}a^{[k-1]} + b^{k} \tag{3}$$

Avec $W^{[k]}$ une matrice de poids de taille $n^{[k+1]} \times n^{[k]}$, et b^k un vecteur d'offset de taille $n^{[k+1]} \times 1$, ainsi de suite jusqu'à la couche de sortie l

$$\hat{y}(x, W, b) = g^{[l]}\left(W^{[l]}a^{[l-1]} + b^{l}\right) \tag{4}$$

C. Fonctions d'activation

Les fonctions $g^{[k]}$ sont des fonctions d'activations, appliquées sur les nœuds intermédiaires du réseau. Elles permettent au modèle de capturer les non linéarités dans les relations entre les données.

D. Forward Propagation

Le processus consistant à évaluer la fonction $\hat{y}(x, W, b)$ dans un réseau feedforward est appelée Forward Propagation, puisqu'il consiste à propager l'information de gauche à droite dans le réseau. Ce processus permet à un réseau de neurones à faire la prédiction à partir de données.

E. Optimisation du réseau

On cherche les poids qui minimisent une fonction de perte *(loss function en anglais*) agrégeant les erreurs de prédiction sur la base. Parmi ces fonctions, on peut citer entre autres :

1) *Erreur quadratique moyenne:*

$$\mathcal{L}(W) = \frac{1}{2}\sum_{m=1}^{M}\left\|\hat{y}\left(x^{(m)}, W\right) - y^{(m)}\right\|^2 \tag{5}$$

2) *Maximum de vraisemblance est calculé dans ce cas comme :*

$$argmax_W \prod_{m=1}^{M} e^{-\frac{\left\|y^{(m)} - \hat{y}\left(x^{(m)}, W\right)\right\|^2}{2\sigma^2}} \tag{6}$$

3) Fonction d'entropie croisée:

$$\mathcal{L}(W) = -\sum_{m=1}^{M}\left(y^{(m)} ln\hat{y}\left(x^{(m)}, W\right) + \left(1 - y^{(m)}\right) ln\left(1 - \hat{y}\left(x^{(m)}, W\right)\right)\right) \quad (7)$$

4) Softmax:

$$\mathcal{L}(\hat{y}, y) = \sum_{m=1}^{M} -log\left(\frac{e^{\hat{y}} y^{(m)}}{e^{y_1 + \cdots + e^{y_3}}}\right) \quad (8)$$

F. Descente de gradient

L'optimisation du réseau se fait par descente de gradient, avec les étapes suivantes :

1) On commence par initialiser les poids W aléatoirement ;
2) On calcule le gradient de la fonction de perte $\mathcal{L}$ par rapport à tous les poids et aux biais du réseau en utilisant la dérivation en chaine(backpropagation) :

$$\left\{\frac{\partial \mathcal{L}}{\partial W_{ji}^{[k]}}\right\}_{i,j,k} \text{ et } \left\{\frac{\partial \mathcal{L}}{\partial b_j^{[k]}}\right\}_{j,k} \quad (9)$$

3) Pour un pas de descente $\alpha > 0$ donné, on met à jour les poids W :

$$W_{ji}^{[k]} \leftarrow W_{ji}^{[k]} - \alpha \frac{\partial \mathcal{L}}{\partial W_{ji}^{[k]}} \quad (10)$$

$$b_j^{[k]} \leftarrow b_j^{[k]} - \alpha \frac{\partial \mathcal{L}}{\partial b_j^{[k]}} \quad (11)$$

4) Et on retourne à l'étape 2, jusqu'à convergence.

G. Réseau de neurones récurrent

Un réseau de neurones récurrents (*RNN : Recurrent Neural Network*) est une variante du réseau de neurones artificiels dont la connexion entre les neurones sont récurrente. Dans le RNN, les neurones sont interconnectés et interagissent non-linéairement. La liaison se fait par des arcs(synapses) présentant des poids et la sortie est combinaison non-linéaire de ses entrées.

Dans un RNN, l'information passe par une boucle. Lorsqu'il prend une décision, il tient compte des données actuelles et de ce qu'il a appris des données reçues précédemment.

On peut appliquer le RNN pour le traitement des données séquentielle de tailles variables. Il a la même technique d'entrainement que des réseaux classiques, néanmoins les réseaux de neurones récurrents simples appelés aussi Vanilla Recurrent Neural Network ne sont pas capables de mémoriser le passé de longs écarts temporels et c'est pourquoi des architectures particulières comme les réseaux LSTM (Long Short-Term Memory) est mis en place.

Les données séquentielles sont des données dont chaque élément possède un ordre, une position et une inscription dans le temps.

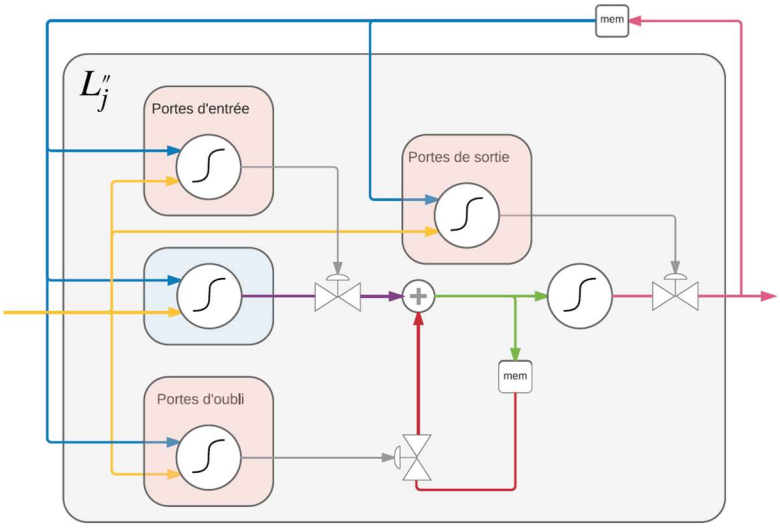

Fig. 1. Visualisation synthétique de la propagation de l'information lors de la passe avant dans une couche LSTM [2]

H. Description des modèles LSTM

Soit une couche LSTM constituée de 4 couches de réseau de neurone récurrent qui s'interagissent dont les 3 utilisent la fonction logistique comme fonction de transfert. Ses sorties se trouvent entre 0 et 1. Ces trois couches font de contrôle sur :

- La quantité des informations qui est dans l'entrée en couche j des cellules LSTM ;
- La quantité maintenue des informations par l'état interne des cellules LSTM vers l'étape suivante ;
- La quantité des informations sortie de la couche j des cellules LSTM.
- La quatrième couche est considérée comme une couche RNN standard qui alimente l'état interne des cellules LSTM à travers les portes d'entrée.

Pour la modélisation des retours entre l'état interne des cellules LSTM, on devrait définir les « peepholes », qui sont les 3 vecteurs de paramètres $u_i, u_f, u_0 \in \mathbb{R}^{n_j}$.

Soit $x(t)$, $t \in [\![1, t_f]\!]$, la séquence des vecteurs d'entrée et $z(t)$, $t \in [\![1, t_f]\!]$, la séquence des vecteurs de sortie appliquées récursivement à chaque pas sur l'ensemble des couches L_j'', $j \in [\![1, \mathbb{N}]\!]$ du réseau au vecteur de sortie de la couche précédente.

Déterminons d'abord l'état des portes d'entrée. Il se fait par la transformation affine a_i définie par W_i et b_i au vecteur de sortie de la couche précédente et par la transformation linéaire définie par V_i du vecteur de sortie de la couche courante j au pas de temps précédent en additionnant avec la contribution du vecteur d'états internes de la couche LSTM au pas de temps précédent par les « peepholes » [2] :

$$g_i(t) = W_i . h_{j-1}(t) + V_i . h_j(t-1) + u_i \odot s(t-1) + b_i \tag{11}$$

$$i(t) = \sigma_i\big(g_i(t)\big) \tag{12}$$

L'opérateur $\odot$ et le produit matriciel de Hadmard c'est-à-dire un produit terme à terme.

Ensuite, en faisant la même opération, l'état des portes d'oubli est donné par la formule suivante [2] :

$$g_f(t) = W_f . h_{j-1}(t) + V_f . h_j(t-1) + u_f \odot s(t-1) + b_f \tag{13}$$

$$f(t) = \sigma_f\left(g_f(t)\right) \tag{14}$$

Après, le nouvel état de la couche j est obtenu en faisant la pondération du nouveau vecteur par les valeurs des portes d'entrée et en faisant additionner avec le vecteur d'états internes du pas précèdent qui a été pondéré par les valeurs de portes d'oubli [2] :

$$g_s(t) = W_s.h_{j-1}(t) + V_s.h_j(t-1) + b_s \quad (15)$$

$$s(t) = f(t)\odot s(t-1) + i(t)\odot\sigma_s(g_s(t)) \quad (16)$$

L'état des portes de sortie s'obtient de la même manière que la porte d'entrée et d'oubli, néanmoins, il utilise l'état interne courant [2] :

$$g_o(t) = W_o.h_{j-1}(t) + V_o.h_j(t-1) + u_o\odot s(t) + b_o \quad (17)$$

$$o(t) = \sigma_0\big(g_O(t)\big) \quad (18)$$

Enfin, la sortie $h_j(t)$ de la couche j au pas de temps t en pondérant l'état interne courant par les valeurs des portes de sortie est donnée par [2] :

$$h_j(t) = o(t)\odot\sigma_h(s(t)) \quad (20)$$

Où $i(t)$, $f(t)$, $s(t)$ et $o(t)$ sont respectivement les vecteurs d'activation les portes d'entrée qui font le contrôle de la quantité d'information à l'entrée de la couche LSTM ; les portes d'oubli contrôlant la quantité d'information qui est en mémoire ; l'états internes de la couche LSTM ; et enfin les portes de sortie qui font le contrôle de la quantité d'information qui sort de la couche LSTM.

Les relations suivantes devraient être vérifiées $\forall t \in [\![1, t_f]\!]$ et $\forall j \in [\![1, \mathbb{N}]\!]$ [2] :

- $h_j(t), g_i(t), g_f(t), g_s(t), g_o(t), i(t), f(t), s(t), o(t) \in \mathbb{R}^{n_j}$
- $h_j(0) = s(0) = 0$
- $h_o(0) = x(0) = 0$
- $h_o(t) = x(t)$ et $z(t) = h_N(t)$

I. Rétropropagation des erreurs

Le calcul des dérivées partielles ou les gradients de la fonction de coût $C(z, c)$, qui sert à minimiser par les différents paramètres des couche LSTM, se fait en utilisant la technique de (BPTT : Backpropagation Through Time). La somme des contributions pour chaque pas de temps donne les gradients globaux. Ce calcul consiste à retro-propager le gradient dans l'ensemble de la couche L_j'' et il se fait en premier lieu au niveau des portes de sortie.

La dérivation de l'équation (20) donne [2] :

$$\frac{\partial C}{\partial o}(t) = (\frac{\partial C}{\partial h_j}(t) + \epsilon(t+1)) \odot \sigma_h(s(t)) \quad (21)$$

Où $\epsilon(t+1))$ est obtenu à partir de l'équation [2] :

$$\epsilon(t) + \epsilon_o(t) + \epsilon_s(t) + \epsilon_f(t) + \epsilon_i(t) \quad (22)$$

Cette équation correspond à la contribution du lien récurrent de la couche L_j'' sur elle-même avec $\epsilon\big(t_f + 1\big) = 0$ parce qu'il n'y a pas de contribution qui parvient au gradient à partir de la fin de la séquence.

En continuant à rétro-propager le gradient par l'équation (18), nous avons [2] :

$$\frac{\partial C}{\partial g_o}(t) = J_{\sigma_o}\big(g_o(t)\big)\frac{\partial C}{\partial o}(t) \quad (23)$$

Avec $J_{\sigma_o}\big(g_o(t)\big)$ est la matrice jacobienne de la fonction d'activation σ_o au point $g_o(t)$.

De l'équation (17), les 3 équations de contributions des portes de sortie aux différents gradients du reste de la couche L_j'' s'écrivent [2] :

$$\delta_o(t) = W_o^T.\frac{\partial C}{\partial g_o}(t) \quad (24)$$

$$\epsilon_o(t) = V_o^T . \frac{\partial C}{\partial g_o}(t) \tag{25}$$

$$\gamma_o(t) = u_o \odot \frac{\partial C}{\partial g_o}(t) \tag{26}$$

Donc :

$$\frac{\partial C}{\partial s}(t) = J_{\sigma_h}(s(t)).\left(\left(\frac{\partial C}{\partial h_j}(t) + \epsilon(t+1)\right)\right) \odot o(t) + \gamma(t) \tag{27}$$

En faisant ensuite la retro-propagation du gradient par l'équation (16), en appliquant les règles de dérivation des fonctions composées, nous avons les formules suivantes [2] :

$$\frac{\partial C}{\partial g_s}(t) = J_{\sigma_s}(g_s(t)).\left(\frac{\partial C}{\partial s}(t) \odot i(t)\right) \tag{28}$$

$$\gamma_s(t) = \left(\frac{\partial C}{\partial s}(t) \odot f(t)\right) \tag{29}$$

En effet, les gradients en sortie des portes d'oubli et les portes d'entrée sont donnés par [2] :

$$\frac{\partial C}{\partial f}(t) = \frac{\partial C}{\partial s}(t) \odot s(t-1) \tag{30}$$

$$\frac{\partial C}{\partial i}(t) = \frac{\partial C}{\partial s}(t) \odot \sigma_s(g_s(t)) \tag{31}$$

Après la rétro-propagation des gradients au travers les équations (14) et (12) en faisant les mêmes processus que pour les portes de sortie, on obtient [2] :

$$\frac{\partial C}{\partial g_f}(t) = J_{\sigma_f}\left(g_f(t)\right).\frac{\partial C}{\partial f}(t) \tag{32}$$

$$\frac{\partial C}{\partial g_i}(t) = J_{\sigma_i}(g_i(t)).\frac{\partial C}{\partial i}(t) \tag{33}$$

Les notations suivantes sont données pour la description des contributions des portes d'oubli aux différents gradients du reste de la couche L_j'' à partir de l'équation (13) [2] :

$$\delta_f(t) = W_f^T . \frac{\partial C}{\partial g_f}(t) \tag{34}$$

$$\epsilon_f(t) = V_f^T . \frac{\partial C}{\partial g_f}(t) \tag{35}$$

$$\gamma_f(t) = u_f \odot \frac{\partial C}{\partial g_f}(t) \tag{36}$$

Et les notations pour les contributions des portes d'entrée aux différents gradients du reste de la couche L_j'' obtenues à partir de l'équation (11) s'écrivent [2] :

$$\delta_i(t) = W_i^T . \frac{\partial C}{\partial g_i}(t) \tag{37}$$

$$\epsilon_i(t) = V_i^T . \frac{\partial C}{\partial g_i}(t) \tag{38}$$

$$\gamma_i(t) = u_i \odot \frac{\partial C}{\partial g_i}(t) \tag{39}$$

Pour terminer, on fait la rétropropagation du gradient via de l'équation (15) :

$$\delta_s(t) = W_s^T . \frac{\partial C}{\partial g_s}(t) \tag{40}$$

$$\epsilon_s(t) = V_s^T . \frac{\partial C}{\partial g_s}(t) \tag{41}$$

Le calcul de la contribution par les équations (21), (27) du lien récurrent de la couche L_j'' sur elle-même pour le pas de temps suivant (t-1) dans le processus de rétropropagation donne [2] :

$$\epsilon(t) = \epsilon_o(t) + \epsilon_s(t) + \epsilon_f(t) + \epsilon_i(t) \tag{42}$$

De ce fait, la rétropropagation du gradient dans la couche L_{j-1}'' est donnée par [2] :

$$\frac{\partial C}{\partial h_{j-1}}(t) = \delta_i(t) + \delta_f(t) + \delta_s(t) + \delta_o(t) \tag{43}$$

Les dérivées partielles du coût C par rapport aux paramètres de la couche L_j'' pourraient être calculées dès que la séquence soit parcourue complètement de la fin jusqu'au début. Donc pour $x \in \{i, f, s, o\}$, on obtient [2] :

$$\frac{\partial C}{\partial W_x} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_x}(t).\, h_{j-1}(t)^T \tag{44}$$

$$\frac{\partial C}{\partial V_x} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_x}(t).\, h_j(t-1)^T \tag{45}$$

$$\frac{\partial C}{\partial b_x} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_x}(t) \tag{46}$$

Et pour les portes d'entrée, d'oubli et de sortie, nous avons [2]:

$$\frac{\partial C}{\partial u_i} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_i}(t) \odot s(t-1)^T \tag{47}$$

$$\frac{\partial C}{\partial u_f} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_f}(t) \odot s(t-1)^T \tag{48}$$

$$\frac{\partial C}{\partial u_o} = \sum_{t=1}^{t_f} \frac{\partial C}{\partial g_o}(t) \odot s(t)^T \tag{49}$$

J. Les types de réseaux récurrents

Le choix du type de l'architecture dépend du résultat attendu. Cependant, les types les plus connus sont :

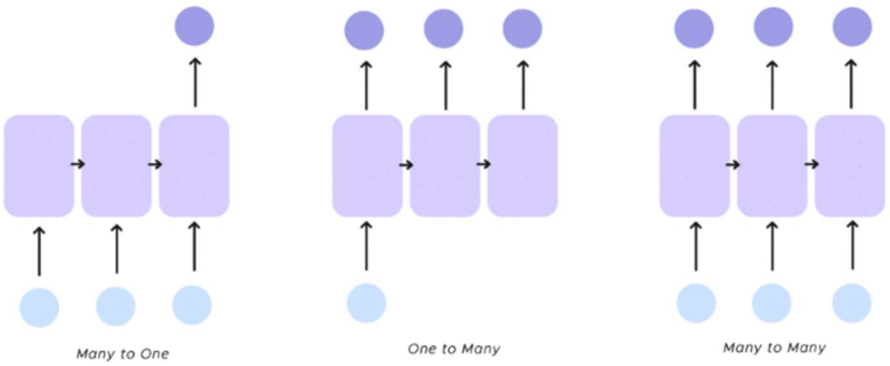

Fig. 2. Type de réseaxu de neurones récurrents

5) One To One

Le type le plus simple de RNN est One-to-One, il a une seule entrée et une seule sortie. Ses tailles d'entrée et de sortie sont fixes et agit comme un réseau neuronal traditionnel. L'application de One-to-One peut être utilisée dans la classification d'image.

6) One To Many

One-to-Many est un type de RNN qui donne plusieurs sorties lorsqu'il est donné une seule entrée. Il prend une taille d'entrée fixe et donne une séquence de sorties de données. Ses modèles peuvent d'appliquer à la génération de musique et sous-titrage d'image.

7) Many To One

Many-to-One est utilisé lorsqu'une seule sortie est requise à partir de plusieurs unités d'entrée ou d'une séquence d'entre elles. Il faut une séquence d'entrées pour afficher une sortie fixe. L'analyse des sentiments est un exemple courant de ce type de réseau neuronal récurrent.

8) Many To Many

Many-to-Many est utilisé pour générer une séquence de données de sortie à partir d'une séquence d'unités d'entrée. Ce type de RNN est divisé en deux sous-catégories :

a) *Taille égale des unités* : Dans ce cas, le nombre d'unités d'entrée et de sortie est le même.

b) *Taille inégale des unités* : Dans ce cas, les entrées et les sorties ont un nombre d'unités différent.

K. Limitations et variantes des RNNs

Les réseaux neuronaux récurrents souffrent d'un problème appelé disparition de gradient, qui est également un problème commun pour d'autres algorithmes de réseaux neuronaux. Le problème de la disparition du gradient est le résultat d'un algorithme appelé rétropropagation qui permet aux réseaux neuronaux d'optimiser le processus d'apprentissage.

En bref, le modèle de réseau neuronal compare la différence entre sa sortie et la sortie désirée et renvoie ces informations au réseau pour ajuster des paramètres tels que les poids à l'aide d'une valeur appelée gradient. Une plus grande valeur de gradient implique de plus grands ajustements aux paramètres, et vice versa. Ce processus se poursuit jusqu'à ce qu'un niveau satisfaisant de précision soit atteint.

Les RNNs exploitent l'algorithme de rétropropagation dans le temps (BPTT : Backpropagation Through Time) où les calculs dépendent des étapes précédentes. Cependant, si la valeur du gradient est trop petite dans une étape pendant la rétroconversion, la valeur sera encore plus petite dans l'étape suivante. Cela entraîne une diminution exponentielle des gradients à un point où le modèle cesse d'apprendre.

Ceci est appelé le problème de disparition de gradient et provoque les RNNs à avoir une mémoire à court terme : les sorties antérieures ont de plus en plus peu ou pas d'effet sur la sortie actuelle. Deux variantes peuvent résoudre ce problème : le LSTM (Long Short-Term Memory) et GRU (Gated Recurrent Units)

Dans notre cas, nous utilisons le LSTM (Long Short-Term Memory). Il est capable de se souvenir des entrées sur une longue période de temps. C'est parce qu'ils contiennent des informations dans une mémoire, un peu comme la mémoire d'un ordinateur. Le LSTM peut lire, écrire et supprimer des informations de sa mémoire. Dans cette mémoire, on décide à chaque fois de stocker ou de supprimer des informations en fonction de l'importance qu'elle accorde à l'information. L'attribution de l'importance se fait par des poids, qui sont également appris par l'algorithme.

L. Mesure des résultats du LSTM avec Root Mean Square Error(RMSE)

Après la phase d'entrainement, une phase de test doit être réalisée afin de mesurer la précision du modelé c'est à dire l'écart entre les observations et la prévision obtenue du modèle. Il existe beaucoup de méthodes pour ce calcul, mais dans notre cas nous avons utilisé le RMSE dont la formule est donnée par la relation suivante :

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Avec $\hat{y}_i$ est la valeur de prévision du modèle et y_i est la valeur d'observation et n le nombre d'observation

M. Données utilisées

Les données utilisées sont des données de précipitation journalière dans la ville de Moramanga, Région Alaotra-Alaotra, Madagascar, datées de 2007 au 2023, du 1 avril 2017 jusqu'au 30 avril 2023. Ça fait 2221 observations au total.

Dans cette étude, la prédiction se fait par rapport à la valeur de précipitation moyenne mensuelle, qui nous donne 73 observations. Nous avons divisé les données en Train set et en Test set : 48 pour le Train set et les reste pour le Test set.

III. RÉSULTATS

5 types de LSTM sont utilisés et les suivants sont les résultats respectifs :

A. LSTM Network for Regression

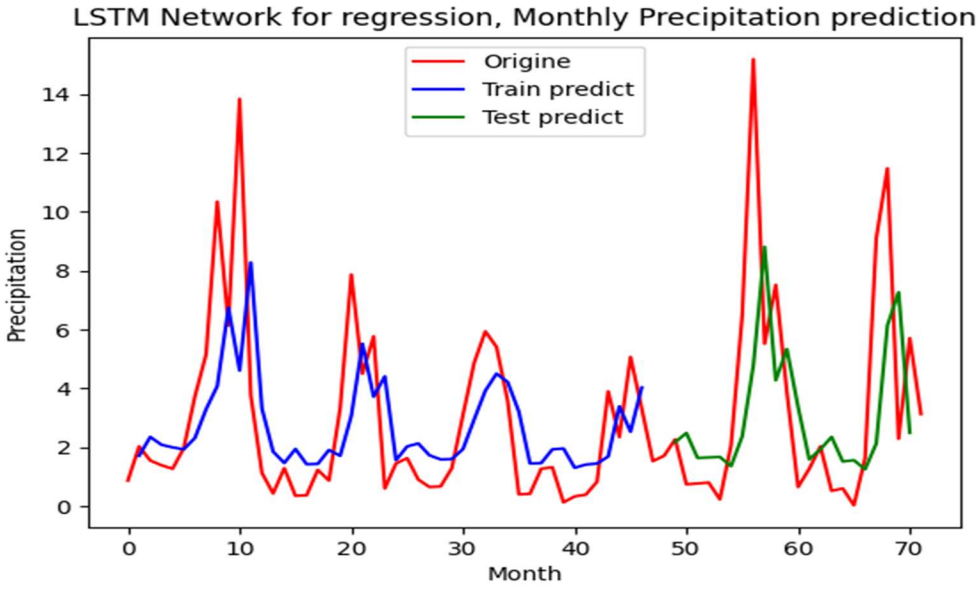

Fig. 3.Résultat obtenu par LSTM Netxork for regression

B. LSTM for Regression Using the Window Method

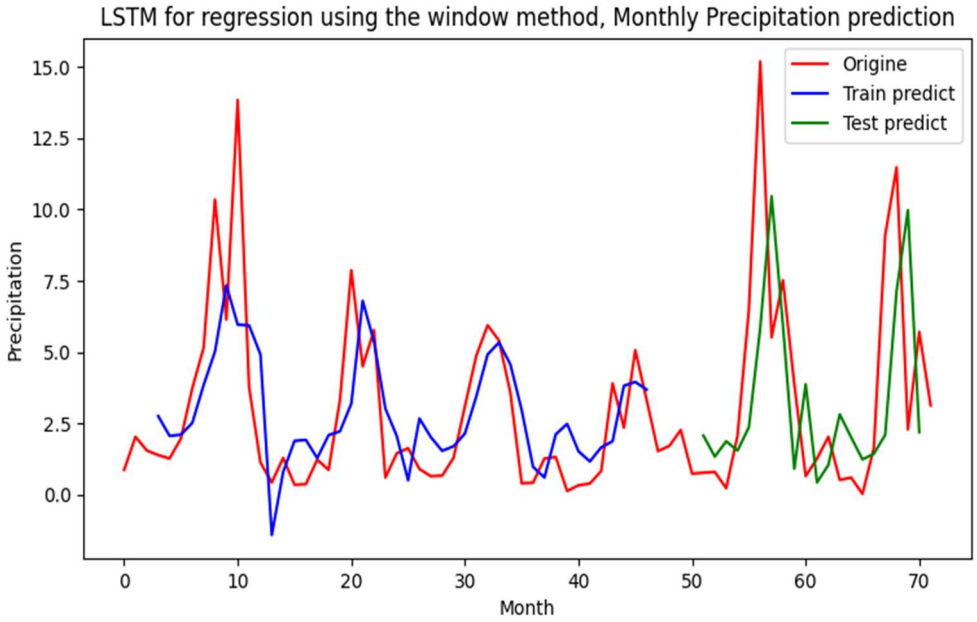

Fig. 4. Résultat obtenu par LSTM for regression using the window method

C. LSTM for Regression with Time Steps

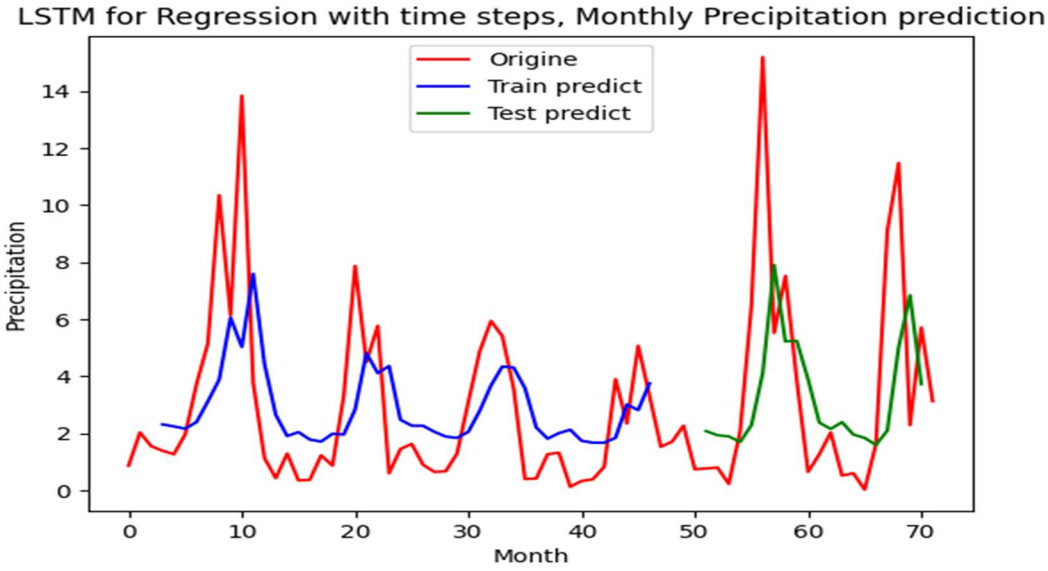

Fig. 5. Résultat obtenu par LSTM for regression with time step

D. LSTM with Memory Between Batches

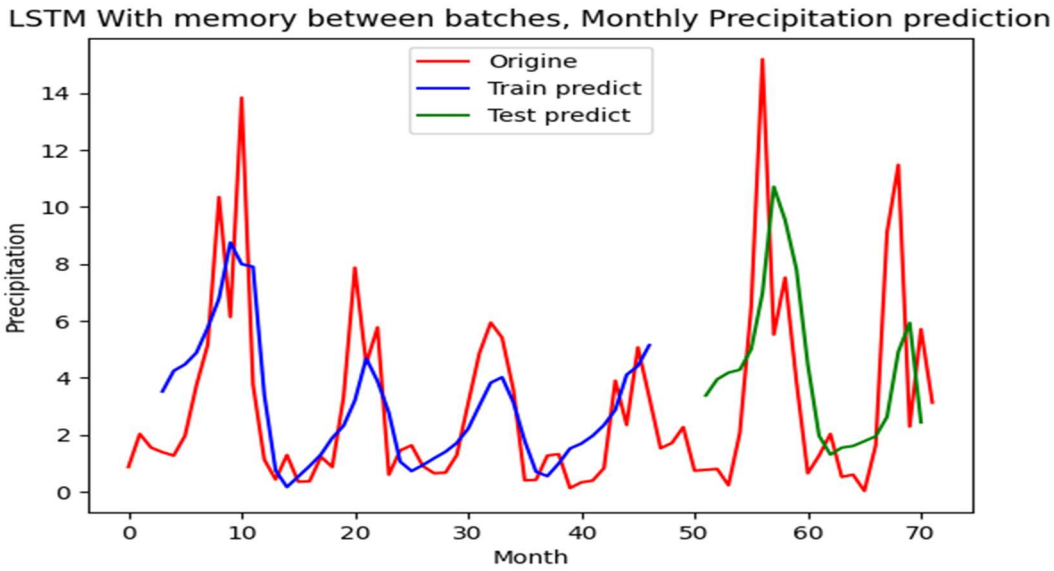

Fig. 6. Résultat obtenu par LSTM with memory between batches

E. Stacked LSTMs with Memory Between Batches

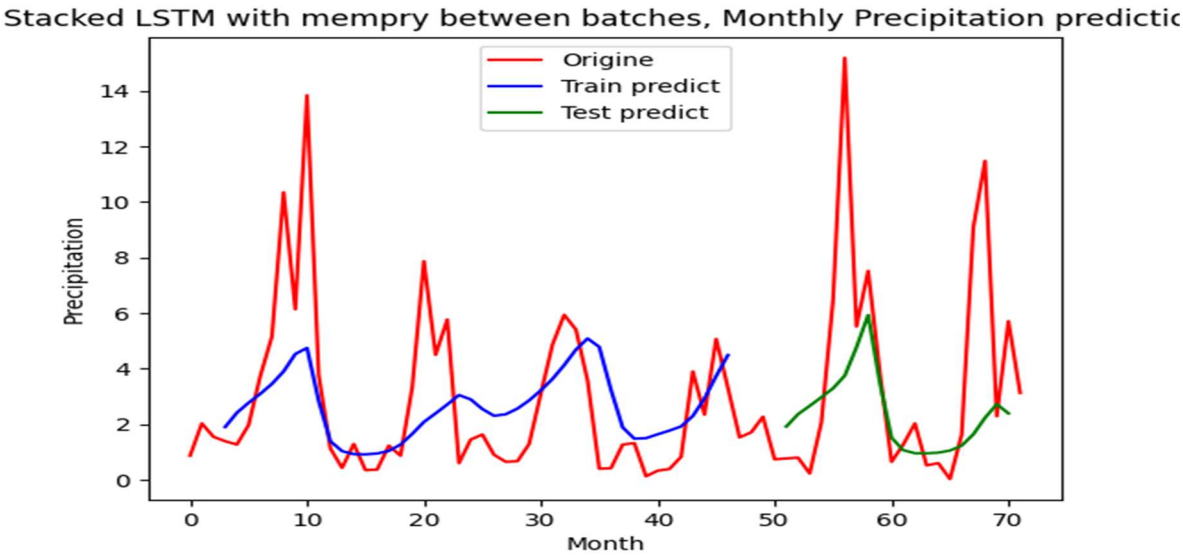

Fig. 7. Résultat obtenu par Stacked LSTM with memory between batches

L'erreur (*RMSE : Root-Mean-Square Error*) du modèle se résume dans le tableau suivant :

TABLE I. ERREURS DES MODELES UTILISES

Type LSTM	Train score RMSE	Test score RMSE
LSTM Network for Regression	2.34	3.58
LSTM for Regression Using the Window Method	2.12	3.91
LSTM for Regression with Time Steps	2.43	3.84
LSTM with Memory Between Batches	1.92	3.76
Stacked LSTMs with Memory Between Batches	2.40	3.94

IV. DISCUSSION

La phase d'entrainement montre que la valeur de la RMSE varie de 1,92 pour le LSTM with Memory Between Batches à 2,43 pour le LSTM for Regression with Time Steps.

Même pour la phase de test, la valeur varie entre 3,58 pour le LSTM Network for Regression et 3,94 pour le Stacked LSTMs with Memory Between Batches.

V. CONCLUSION

La recherche des outils pour la prédiction de précipitation est l'un de problème continu et nécessite une étude très approfondie. Dans cette expérience, les modèles LSTM with Memory Between Batches et LSTM Network for Regression présentent respectivement de bon résultat de prédiction pendant la phase d'entrainement et de test.

Ce résultat est encore indicatif car seulement pour prédiction obtenue par 73 observations. Mais on peut encore approfondir les recherches en appliquant avec de données de précipitation en plus grand nombre d'observations et en comparant encore avec les autres outils de prédiction.

REFERENCES

[1] Valentin NOËL, Séries temporelles et réseaux de neurones récurrents, Culture Science de l'ingénieur, Édité le 05/09/2022

[2] Gregory GellyRéseaux de neurones récurrents pour le traitement automatique de la parole, Submitted on 12 Oct 2017, HAL Id : tel-01615475.

[3] Ali Djaidja, Étude de la classification supervisée des données environnementales à l'aide de réseaux de neurones de fonctions à base radiales, juin 2016, 84 pages.

[4] Lefebvre Brossard Antoine, Sur l'utilisation de réseaux de neurones dans un système de recommandations réciproques, août 2018, 65 pages.

[5] David Roche, Deep Learning et apprentissage par renforcement pour la conception d'une intelligence artificielle pour le jeu yokai no mori, 46 pages.

[6] Bendaoud Youcef, Prédiction des résistances mécaniques des bétons à base des ciments composés en utilisant les réseaux neurones artificiels, 07 juillet 2014, 126 pages.

[7] Aurélien Pottiez et Marc Duquesnoy, Intelligence artificielle et apprentissage de réseaux connexionnistes, 11 mai 2018, page 67.