République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de larecherche scientifique
Université Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Systèmes et Multimédias

Thème

Capacité de«Deeplearning»A prédire les chaos dans un système chaotique

Présenté Par :

Bendib Islem

Devant le jury :

Mr.Bendjanna Hakim	*Prof*	*Univérsité Larbi Tébessa*	*Président*
Mr. Haouam Med-Yacine	*MCB*	*Univérsité Larbi Tébessa*	*Examinateur*
Mr. Derdour Makhlouf	*Prof*	*Univérsité Larbi Tébessa*	*Encadreur*
Mr. Betouil Ali-Abdelatif	*MCB*	*Univérsité Larbi Tébessa*	*Co-Encadreur*

Date de soutenance : 15/09/2020

Remerciments

Je tiens tout d'abord à remercier ***Docteur Makhlouf Derdour*** *pour m'avoir proposé ce sujet qui m'a permis de m'initier à la recherche Scientifique. Son suivi régulier de l'évolution de mon travail, ses conseils et ses encouragements m'ont permis de réaliser ce mémoire dans d'excellentes conditions de travail.*

Mes remerciements iront également aux membres du jury ***Pr.BenDjanna Hakim*** *&* ***Haouam Med Yacine*** *pour avoir accepté d'évaluer mon travail de recherche.*

Merci a mon ami Mohamed Mellah. Sans oublier tous nos enseignants qui nous ont assurés des études de haut niveau et qui nous permis d'acquérir des connaissances

Dedicace

Je dédie ce travail à :

Mes parents.

Mon directeur de mémoire

Toutes mes enseignants

Toute ma famille

Mes amis.

Islem

Abstract

A century ago, the pioneers of chaos theory discovered that the "butterfly effect" makes long-term prediction impossible. Even the smallest disturbance in a complex system (like the weather, the economy or just about everything else) can trigger a concatenation of events leading to a dramatically divergent future. Unable to establish the state of these systems with sufficient accuracy to predict how they will unfold; we live in a veil of uncertainty.

Deep learning is a computer technique that is the basis of recent successes in artificial intelligence - to predict the future evolution of chaotic systems at distant horizons. External experts have described this approach as innovative and likely to find wide application.

The work of this master's project aims to involve deep learning in the process of predicting chaos in different types of dynamic systems in order to be able to avoid disasters that may come.

ملخص

قبل قرن من الزمن اكتشف رواد نظرية الفوضى أن "تأثير الفراشة" جعل التنبؤ طويل المدى مستحيلاً. حتى أصغر اضطراب في نظام معقد (مثل الطقس أو الاقتصاد أو كل شيء آخر تقريبًا) يمكن أن يؤدي إلى سلسلة من الأحداث التي تؤدي إلى مستقبل متباين بشكل كبير. غير قادر على تحديد حالة هذه الأنظمة بدقة كافية للتنبؤ بكيفية ظهورها ، نحن نعيش في حجاب من عدم اليقين.

التعلم العميق هو تقنية حاسوبية تقف وراء النجاحات الأخيرة في الذكاء الاصطناعي - للتنبؤ بالتطور المستقبلي للأنظمة الفوضوية في آفاق بعيدة. وصف الخبراء الخارجيون هذا النهج بأنه مبتكر ومن المرجح أن يجد تطبيقًا واسعًا.

يهدف عمل مشروع هذا الماستر إلى إشراك التعلم العميق في عملية التنبؤ بالفوضى في أنواع مختلفة من الأنظمة الديناميكية من أجل تجنب الكوارث التي قد تحدث.

RESUME

Il y a un siècle déjà, les pionniers de la théorie du chaos ont découvert que « l'effet papillon » rend la prédiction à long terme impossible. Même la plus petite perturbation dans un système complexe (comme la météo, l'économie ou à peu près tout le reste) peut déclencher une concaténation d'événements conduisant à un avenir dramatiquement divergent. Incapables d'établir l'état de ces systèmes avec suffisamment de précision pour prédire comment ils se déroulent, nous vivons dans un voile d'incertitude.

Le deeplearning ou apprentissage profond est une technique informatique à la base des récents succès en intelligence artificielle - pour prédire l'évolution future de systèmes chaotiques à des horizons lointains. Des experts externes ont qualifié cette approche d'innovante et susceptible de trouver une large application.

Le travail de ce projet de master vise à impliquer l'apprentissage profond dans le processus de prédiction de chaos dans les différents types de systèmes dynamiques afin de pouvoir éviter des catastrophes qui peuvent venir.

Mot clés : Chaos, Système dynamique, Equations différentielles, Long Short-TermMemory, RNN, Attracteur de Lorenz, CNN, jeux d'échecs, non equilibruim.

Table des matières

IntroductionGénérale 2

Chapitre 1/Les Systèmes Dynamiques Chaotiques

1.1.Introduction 5

1.2.Systèmes dynamiques 5

1.3.Types systèmes dynamiques 5

1.4.Caractéristiques d'un système chaotique 6

1.5.L'espacedesphases 7

1.6. Bifurcation et routes versle chaos 8

1.7.Exemple des systèmes dynamique chaotique 10

1.7.1. Notiond'attracteur 10

1.7.2.L'attracteurde Lorenz 11

1.7.3.Exposantsde Lyapunov 12

1.8.Conclusion 14

Chapitre2/ Prédiction de chaos &Deep Learning

2.1Introduction 16

2.2Réseau deneurone Artificielle 16

- **2.2.1 Réseaux de neurones recurrents**……16
- **2.2.2 Réseaux de neurones convolutifs**……21

2.3 Prédiction de chaos en météo utilisant deeplearning 26

- **2.3.1 limitations et contrôle de chaos**……..26
- **2.2.2 approche theoriques**……26
- **2.2.3 approche techniques**……**27**

2.4 Prédiction de position d'échecs 35

- **2.4.1 approche theoriques**…….....……40.
- **2.4.2 approche thechniques**…….....……...40

2.5.Conclusion 44

Chapitre 3/Les résultats

3.1Introduction 46

3.2. Résultat pour cas d'étude 1 46

3.3. Résultat pour cas d'étude2 47

3.4. Les outils utilisés 48
3.5. Conclusion 49

Conclusion Générale 51

Liste des figures

Figure 1.1 – Digramme de suite logistique	**6**
Figure 1.2 – Espace de phase prédominant	**8**
Figure 1.3 – Bifurcation de fourche	**9**
Figure 1.4 – Bifurcation col-nœud	**9**
Figure 1.5 – Les bifurcation de hopf	**9**
Figure 1.6 – Les bifurcation de dédoublement de période	**10**
Figure 1.7 – L'attracteur de Lorenz	**12**
Figure 1.8 – Divergence de deux trajectoires dans le plan de phase	**12**
Figure 2.1 – Génération de texte caractère par caractère	**17**
Figure 2.2 – Traitement d'une séquence de taille T=10 par une couche récurrente	**19**
Figure 2.3 – Prédiction de la distribution de probabilité du caractère suivant	**19**
Figure 2.4 – Une couche RNN , prenant en entrée des séquences de 10 caractères	**20**
Figure 2.5 – Le réseau CNN	**22**
Figure 2.6 – Allure de la fonction ReLu	**24**
W<c 006^{E}gl :b	**28**

Figure 2.8 – Fonction de LSTM	**30**
Figure 2.9 – Arber de minmax_1	**38**
Figure 2.10 – Algorithme minmax_1	**38**
Figure 2.11 – Algorithme minmax_2	**40**
Figure 2.12 – Algorithme minmax_2	**40**
Figure 2.13 – l'échiquier au model	**42**
Figure 3.1 – Résultat pour cas d'étude 1	**46**
Figure 3.2 – Résultat pour cas d'étude 2	**47**
Figure 3.3 – Logo Pycharm	**48**
Figure 3.4 – Logo Visual Studio	**49**

Liste des tableaux

Tableau 1.1 – Exposants de Lyapunov et Dimensions	**13**
Tableau 3.1 – Résultat pour cas d'étude	**47**

Introduction Générale

La majorité des références stipulent que ce sont les travaux d'Henri Poincaré sur la mécanique céleste et la mécanique statistique vers1900,qui sont à l'origine de La découverte de la dynamique chaotique des systèmes non-linéaires, depuis alors ils ont suscité peu d'intérêt et sont tombés dans l'oubli.

En 1963, le météorologue Edward Lorenz, du « Massachusetts Institute of Technolog », mette en évidence le caractère chaotique des conditions météorologiques et par conséquent des mouvements turbulents d'un fluide comme l'atmosphère[1].

Alors qu'il cherchait à déterminer des conditions météorologiques futures à partir de données initiales sur son ordinateur, il constata qu'une modification minime des données initiales (del'ordre d'unpourmille)entrainait des résultats radicalement différents.Après avoir modélisé le mouvement des masses d'air par des relations(trèssimplifiées)dethermodynamique et de mécaniquedes fluides, il a programmé son ordinateur de façon à obtenir une simulation numérique.Al'époque,cela prenait beaucoup de temps.

Un jour, pour ne pas recommencer les calculs depuis le début, il décida de reprendre son listing et de rentrer en tant que conditions initiales des valeurs prises au cours de la simulation de la veille. L'ordinateur lui donnait une précision à cinq chiffres, cependant trois chiffres significatifs lui semblaient largement suffisants pour ce genre de mesures physiques.Il tronqua donc ces nombres et repris le calcul. Les résultats qui suivirent furent le déclic. D'abord la simulation semblait redonner les mêmes valeurs, mais au bout d'un moment rien ne concordait, tout se passait comme si le mouvement représenté par ces valeurs changeait complètement de trajectoire et ce, à cause d'une approximation de l'ordre de1^{-4}[22].

Cette anecdote est à la base de ce que l'on appelle maintenant le chaos : une infime variation des conditions initiales d'un système bouleverse complètement son évolution. Lorenz venait de mettre en évidence la sensibilité aux conditions initiales.

Il expliqua d'ailleurs très joliment cette notion à l'aide de l'image métaphorique suivante :

- Le battement d'ailes de quelques papillons dans les forêts de l'Afrique centrale peut provoquer des tempêtes auxAmériques.
- La découverte de Lorenz intrigua un certain nombre de physiciens et demathématiciens.
- Les travaux de Poincaré sortirent alors du placard et furent compris comme ils auraient dûl'être depuislongtemps.
- Ilsfournirentl'ossaturemathématiquequiallaitpermettrel'étudedesphénomènesnon-linéaires sous un nouveau jour.

Tous ces travaux lancèrent sur de nouvelles bases les réflexions concernant le déterminisme et la prévisibilité

La prédiction de chaos est un problème de recherche clé dans le domaine de système dynamique précisément les systèmes non-linéaire pour résoudre de nombreux problèmes de recherche comme la prédiction à long terme et les calculs de taux d'erreur.

Dans ce travail, nous proposons une nouvelle approche basée deep Learning pour la prédiction de chaos dans deux différents types de systèmes dynamiques. le premier système proposé est évalué pour prédire le climat en utilisant le LSTM (Long Short Term Memory), les résultats expérimentaux ont montrés une prédiction a long term avec une grande précision.

Pour le second système proposé nous avons proposé un moteur d'échec pour prédire chaque position et ses bifurcations basé sur un réseau de neurones convolutif, les résultats montre une grande occurrence pour les pièces de poids lourds.

Dans ce contexte et dans le cadre de notre projet de fin d'étude, après l'introduction générale, nous avons organisé ce manuscritde la façon suivante :

Le premier chapitre est consacré au contexte d'étude, Nous allons présenter les systèmes dynamiques et le concept du chao, ainsi des exemples des système chaotiques Applications

Dans le deuxième chapitre, nous allons présenter les systèmes de prédiction basant sur l'intelligence artificiel.

Le troisième chapitre présente les résultats deux cas d'étude,un système de prédiction météorologique et un système de jeu d'échecs.

Enfin, Nous conclurons ce manuscrit en présentant quelques perspectives ouvertes par notre travail.

CHAPITRE 1

Les Systèmes Dynamiques Chaotiques

1.1 Introduction

Un système dynamique est dit chaotique si une portion « significative » de son espace desphases présente simultanément le phénomène de sensibilité aux conditions initiales et une forterécurrence.

Dans ce chapitre, nous présentons les différèrent concepts des systèmes dynamiques chaotiques[2].

1.2 Systèmes dynamiques

Un système dynamique est une structure qui évolue au cours du temps de façon à la fois :

- ✓ Causale, où son avenir ne dépend que de phénomènes du passé ou duprésent
- ✓ Déterministe, c'est-à-dire réagit toujours de la même façon à unévénement.

1.3 Types des systèmes dynamiques

L'évolution déterministe du système dynamique peut alors se modéliser de deux façons :

- ✓ Une évolution continue dans le temps, représentée par une équation différentielle.
  - **Exemple :** rédaction chimique.

- ✓ Une évolution discrète dans le temps, représentée par les fonctionsitératives.

L'étude théorique de ces modèles discrets est fondamentale, car elle permet de mettre en évidence des résultats importants, qui se généralisent souvent aux évolutions dynamiques continues. Elle est représentée par le modèle général des équations aux différences finie [3] .

- **Exemple :** logisticmap

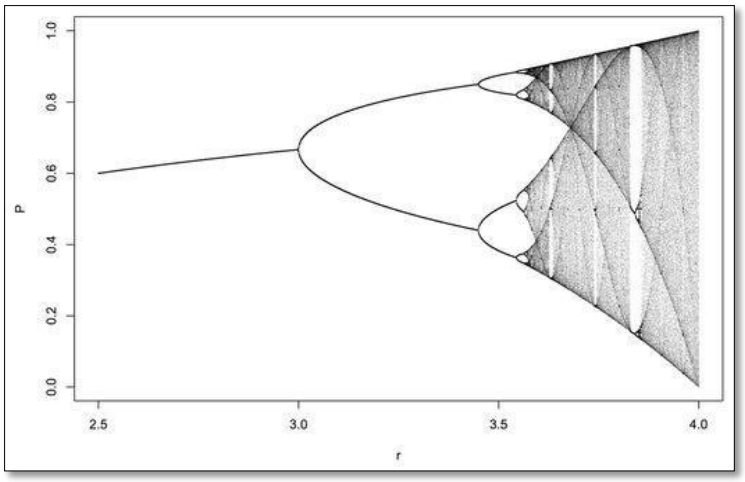

Figure 1.1 – Digramme de suite logistique[2]

1.4 Caractéristiques d'un système chaotique

Le chaos tel que le scientifique le comprend ne signifie pas l'absence d'ordre ; il se rattache plutôt à une notion d'imprévisibilité, d'impossibilité de prévoir une évolution à long terme du fait que l'état final dépend de manière si sensible de l'état initial. On appelle donc un système dynamique chaotique, un système qui dépend de plusieurs paramètres et qui est caractérisé par une extrême sensibilité aux conditions initiales. Il n'est pas déterminé ou modélisé par des systèmes d'équations linéaires ni par les lois de la mécanique classique [3].

a) Déterministe:C'est-à-dire a des règles fondamentales déterministes et non probabilistes, pas de hasard (no randonnes).

b) Sensibilité aux conditions initiales: Une légère différence dans les conditions initiales entraîne des résultats totalement différents.

c) Non périodique:C'est-à-dire ne répète pas ses valeurs à intervalles ou périodes réguliers, F(x+p) ! =f(x).

d) Délimité.

1.5 L'espace des phases

L'espace des phases un espace dans lequel tous les états possibles d'un système sont représentés.

Il permet de traduire des séries de nombre en une représentation spatiale, de dégager l'essentiel de l'information d'un système en mouvement et de dresser la carte routière de toutes ses possibilités.

Est un espace mathématique souvent multidimensionnel. Chaque axe de coordonnées de cet espace correspond une variable d'état du système dynamique étudié et chaque variable d'état caractérise le système à un instant donné. Pour chaque instant donné, le système est donc caractérisé par un point de cet espace. A l'instant suivant, il sera Caractérisé par un autre point et ainsi de suite[4].

Si l'espace des phases est représenté système dans le temps. L'ensemble des trajectoires possibles constitue le portrait de phases. Celui-ci peut aider à percevoir l'attracteur du système. Considérons la relation suivante entre deux points x et y de M :

x _ y , x; y appartiennent à la même orbite

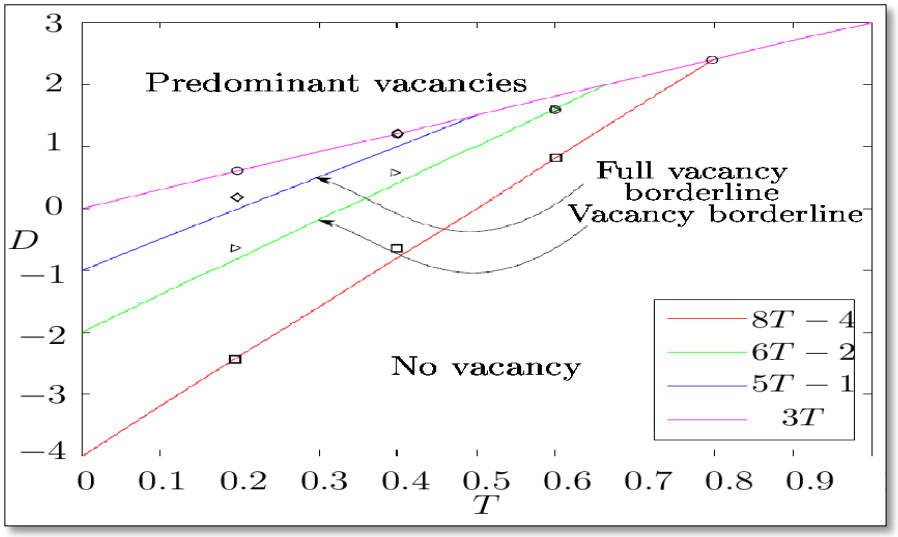

Figure 1.2 – Espace de phase prédominant[6]

1.6. Bifurcation et routes vers le chaos

La théorie de bifurcation est l'étude mathématique des changements qualitatifs ou topologiques de la structure d'un système dynamique. Une bifurcation survient lorsqu'une variation quantitative d'un paramètre du système engendre un changement qualitatif des propriétés d'un système telles que la stabilité, le nombre de points d'équilibre ou la nature des régimes permanents. Les valeurs des paramètres au moment du changement sont appelées valeurs de bifurcation.

Dans les systèmes dynamiques, un diagramme de bifurcation montre les comportements possibles d'un système, à long terme, en fonction des paramètres de bifurcation.

La théorie des bifurcations consiste à classer les différents types de bifurcation en classes. Chaque classe correspond à une certaine symétrie dans le problème.

Parmi les différents types de bifurcations, on trouve :

- **Les bifurcations « de fourche » :** Un équilibre stable se déstabilise en un équilibre instable, et deux équilibres stables sont créés. Cette transition peut se faire de façon supercritique (de façon continue et prévisible) ou sous-critique (discontinue, avec des phénomènesd'hystérèse)
- **Les bifurcations col-nœud :** Deux points d'équilibres existent (un stable et un instable) avant la bifurcation. Après la bifurcation, plus aucun équilibren'existe.
- **Les bifurcations de Hopf:**Ce sont des bifurcations oscillantes, comme l'attracteur de Lorenz.
- **Les bifurcations de dédoublement de période :** Ce sont des bifurcations qui mènent à des comportements chaotiques. Elles peuvent par exemple s'obtenir en faisant rebondir une balle de ping-pong sur une surface oscillante, et en augmentant la fréquence d'oscillation.

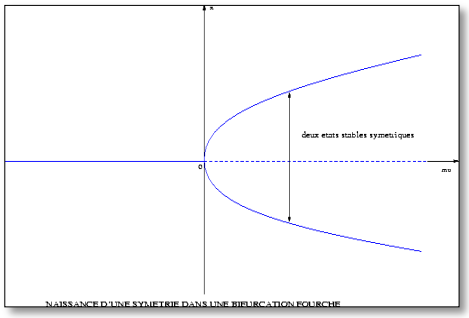

Figure 1.3 – Bifurcation de fourche[2]

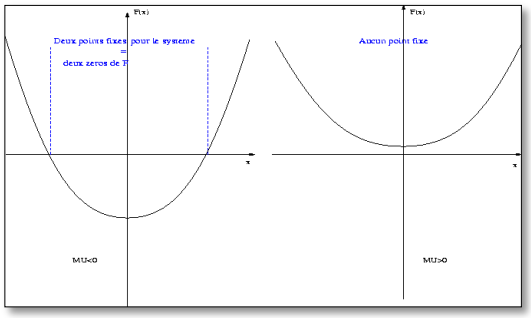

Figure 1.4 – Bifurcation col-nœud[2]

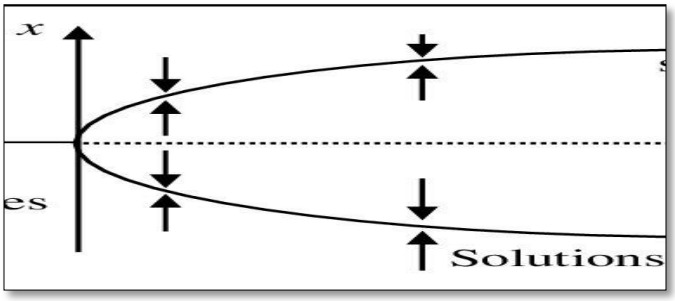

Figure 1.5 – Les bifurcation de hopf[2]

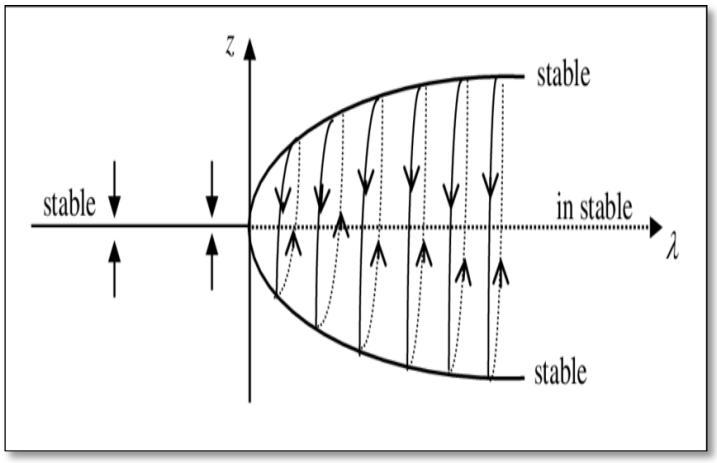

Figure 1.6 – Les bifurcation de dédoublement de période[2]

1.7 Exemple des systèmes dynamique chaotique

1.7.1 Notion d'attracteur

L'étude du comportement asymptotique d'un système dynamique régi par un flot d'équations différentielles non linéaires révèle très souvent la notion d'attracteur, défini comme l'ensemble compact de l'espace des phases invariant par ce flot et vers lequel convergent toutes les trajectoires du système. Il existe quatre cas defigures correspondants à des solutions différentes du flot, mettant en évidence des attracteurs différents:

- **Le point attracteur :** correspondant à une solution stationnaire constante, donc de fréquencenulle.
- **Le cycle limite attracteur** : caractérisant un régime périodique, la solution possède une seule fréquence debase.
- **Le tore supra Tr (r≥2) :** cet attracteur correspond à un régime quasi-periodique ayant r fréquences de base indépendantes (cas le plus simple r=2, dynamiquebiperiodique).

- **L'attracteur étrange :** cet attracteur est associé à un comportement quasi-aléatoire dit chaotique, caractérisé par un spectre de puissance continue et une fonction d'auto-corrélation s'annulant très rapidement. Contrairement aux signaux périodiques (quasi-périodiques) pour laquelle la similitude reste présente pour autant que la périodicité n'est altérée ; ce qui a pour conséquence immédiate la périodicité du comportement du système, le caractère fini de la portée de la fonction d'auto-corrélation temporelle pour le régime chaotique met en évidence la perte progressive de la similitude interne et donc l'imprédictibilité. Cette perte de mémoire du signal est due au phénomène de contraction des volumes dans l'espace des phases des systèmes dynamiques dissipatifs, mais aussi et surtout au phénomène de dilatation directionnelle de ces volumes.[24]

1.7.2 L'attracteur de Lorenz:

Est une structure fractale correspondant au comportement à long terme de l'oscillateur de Lorenz. L'attracteur montre comment les différentes variables du système dynamique évoluent dans le temps en une trajectoire non périodique.

Ce système différentiel s'écrit :

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

Où :

X *- proportionnel à l'intensité du mouvement de convection.*

Y *- proportionnel à la différence de température entre ascendant et courants descendants.*

Z *- proportionnel à la différence du profil de température vertical de linéarité.*

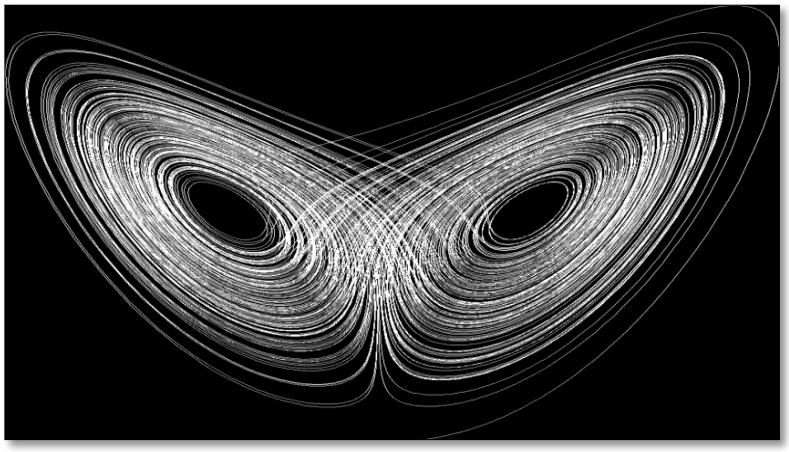

Figure 1.7 – L'attracteur de Lorenz[2]

1.7.3 Exposants de Lyapunov

L'évolution chaotique est difficile à appréhender car la divergence des trajectoires sur l'attracteur est rapide. Pour cette raison on essaye si c'est possible de mesurer sinon d'estimer la vitesse de divergence ou de convergence. Cette vitesse est donnée par l'exposant de Lyapunov qui caractérise le taux de séparation de deux trajectoires très proches. Donc deux trajectoires dans le plan de phase initialement séparées par un taux Z1 divergent après un temps $\Delta t = t2 - t1$ vers Z2 tel que:

$$|Z2| \approx \exp(\lambda.\Delta t)\,|Z1|$$

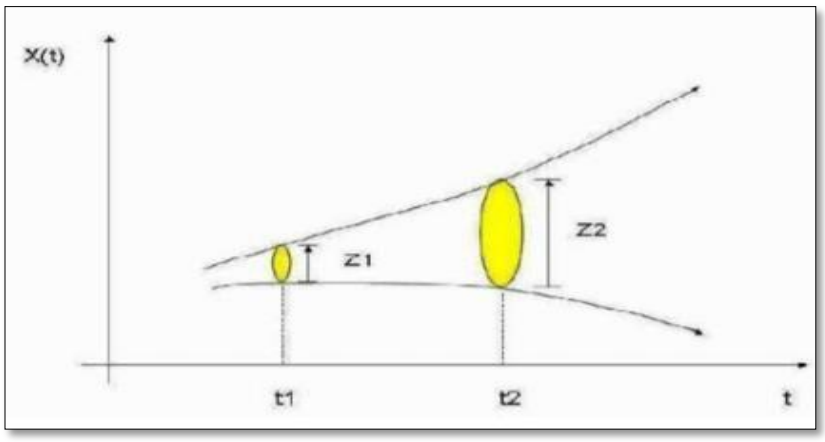

Figure 1.8 – Divergence de deux trajectoires dans le plan de phase[6]

Considérons un système dynamique dont l'espace des phases est de dimension n etprenons à t=0 une hyper sphère infiniment centré en X appartenant à l'attracteur$(X \in R^n)$ avec un rayon ε_0 .Au temps t >> 0, cette hyper sphère se transforme en une hyper-ellipsoïde de ndemi-axes [2]

$$\varepsilon_i(t) \approx \varepsilon_0 \exp(\lambda_i t) \qquad \text{i=1,2,..... n}$$

Les exposants de Lyapunov sont tels que :

$$\lambda_i = \lim_{t\to\infty} \lim_{\varepsilon_0 \to 0} \frac{1}{t} \log\left(\frac{\varepsilon_i}{\varepsilon_0}\right)$$

Ils caractérisent de façon assez précise la dynamique du système. Pour un attracteur non chaotique, les exposants de Lyapunov sont tous inférieurs ou égaux à zéro et leur somme est négative. Un attracteur étrange possèdera toujours au moins trois exposants de Lyapunov, dont un au moins doit être positif

Etat	Attracteur	Dimension	Exposants de Lyapunov
Point d'équilibre	Point	0	$\lambda_n \leq \cdots \leq \lambda_1 \leq 0$
Périodique	Cercle	1	$\lambda_1 = 0$ $\lambda_n \leq \cdots \leq \lambda_2 \leq 0$
Période d'ordre 2	Tore	2	$\lambda_1 = \lambda_2 = 0$ $\lambda_n \leq \cdots \leq \lambda_3 \leq 0$
Période d'ordre K	K-Tore	K	$\lambda_1 = \cdots = \lambda_k = 0$ $\lambda_n \leq .. \leq \lambda_{1k+1} \leq 0$
Chaotique		Non entier	$\lambda_1 > 0$ $\sum_{i=1}^{n} \lambda_i < 0$
Hyper chaotique		Non entier	$\lambda_1 > 0 \ \ \lambda_2 > 0$ $\sum_{i=1}^{n} \lambda_i < 0$

Tableau 1.1 – Exposants de Lyapunov et Dimensions

1.8 Conclusion

Dans ce chapitre nous avons défini les systèmes dynamiques chaotiques, leurs caractéristiques, leurs types. Le chapitre suivant présente la notion de deep Learning et leur utilisation efficace dans la prédiction des chaos.

CHAPITRE 2

Prédiction de chaos &Deep Learning

2.1 Introduction

L'apprentissage profond est compris comme une branche de machine learning, qui est basée sur un ensemble d'algorithmes qui cherchent à façonner des abstractions de haut niveau de données à l'aide d'un graphique profond avec plusieurs couches de traitement, Composé de plusieurs altérations linéaires et nonlinéaires.

Dans ce chapitre, nous présentons les différèrent concepts d'apprentissage ainsi leur utilisation dans la prédiction des chaos.

2.2 Réseau de neurone Artificielle

Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les neurones formels. Les principaux réseaux se distinguent par l'organisation du graphe (en couches, complets. . .), c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones (leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques...

2.2.1 Les réseaux récurrents

Les RNN (pour Récurrent Neural Networks)sont des réseaux de neurones dans lesquels l'information peut se propager dans les deux sens, y compris des couches profondes aux premières couches. En cela, ils sont plus proches du vrai fonctionnement du système nerveux, qui n'est pas à sens unique. Ces réseaux possèdent des connexions récurrentes au sens où elles conservent des informations en mémoire : ils peuvent prendre en compte à un instant t un certain nombre d'états passés. Pour cette raison, les RNNs sont particulièrement adaptés aux applications faisant intervenir le contexte, et plus particulièrement au traitement des séquences temporelles comme l'apprentissage et la génération de signaux, c'est à dire quand les données forment une suite.

Néanmoins, pour les applications faisant intervenir de longs écarts temporels (typiquement la classification de séquences vidéo), cette « mémoire à court-terme » n'est pas suffisante. En effet, les RNNs « classiques » (réseaux de neurones récurrents simples ou VanillaRNNs) ne sont capables de mémoriser que le passé dit proche, et commencent à « oublier » au bout d'une cinquantaine d'itérations environ. Ce transfert d'information à double sens rend leur entrainement beaucoup plus compliqué, et ce n'est que récemment que des méthodes efficaces ont été mises au point comme les LSTM (Long Short Term Memory). Ces réseaux à large « mémoire court-terme » ont notamment révolutionné la reconnaissance de la voix par les machines (Speech Recognition) ou la compréhension et la génération de texte (Natural Langage Processing). D'un point de vue théorique, les RNNs ont un potentiel bien plus grand que les réseaux de neurones classiques : des recherches ont montré qu'ils sont « Turing-complet » (ou Turing-complete), c'est à dire qu'ils permettent théoriquement* de simuler n'importe quel algorithme. Cela ne donne néanmoins aucune piste pour savoir comment les construire pour cela dans la pratique.

❖ Architecture globale:

Commençons par rappeler le problème de génération de texte. Afin de générer du texte, nous partons d'une séquence de caractères de taille fixe pour prédire le caractère suivant.

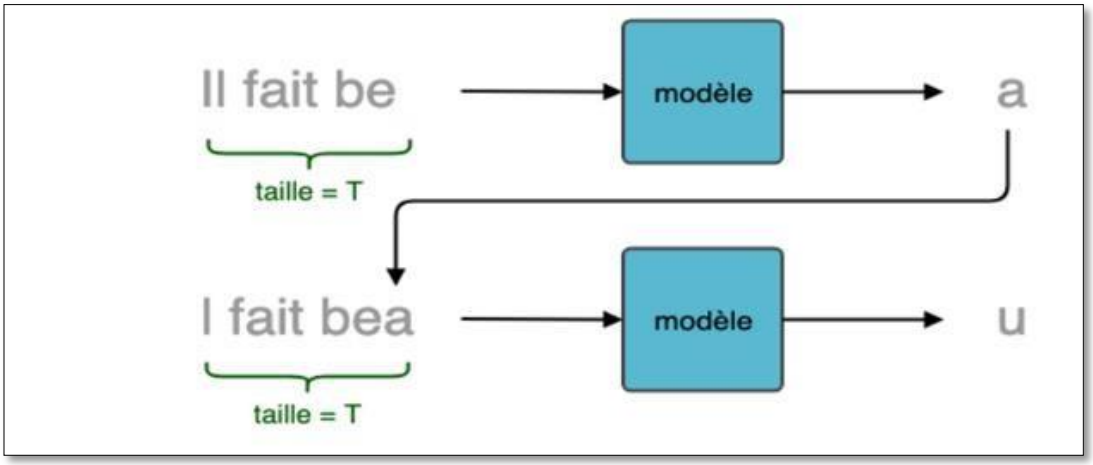

Figure 2.1 – Génération de texte caractère par caractère[19]

Le pré-traitement détaillé dans l'article précédent permet de représenter numériquement le texte brut, le transformant en une matrice dont les dimensions sont :

nb_échantillons (N) * taille_séquence (T) * nb_variables (M)

Où :

- **nb_échantillons (N)** = nombre de séquences d'entraînement (taille dudataset)
- **taille_séquence (T)** = nombre (fixe) de caractères par séquence (dimension temporelle). Dans l'article précédent, nous avions choisi d'entraîner le modèle sur des séquences de taille T=50. Le choix de T est empirique et il y a un compromis à trouver : si T est petit, le modèle aura la « mémoire courte ». Si T est grand, l'entraînement sera très long et pas toujoursefficace.
- **nb_variables (M)** = taille de la représentation vectorielle de chaque caractère. Nous avons choisi de limiter notre vocabulaire aux 26 lettres de l'alphabet + 4 caractères spéciaux. Chaque caractère est donc représenté par un vecteur de taille M=30 grâce auone-hot-encoding.

En résumé, le modèle prend en entrée un tableau de N séquences, chacune de longueur T=10 caractères, où chaque caractère est un vecteur numérique de taille M=30.

Une couche de type RNN prend en entrée une séquence de ce tableau (donc une matrice de taille (T x M)) et retourne en sortie un vecteur de taille R qui est une représentation compressée de la totalité de la séquence de caractères. Encore une fois, R est un hyper paramètre à choisir judicieusement. Pour l'exemple, on peut choisir R=16. C'est en quelque sorte l'équivalent du nombre de neurones dans une couche dense (ou « fully-connected«).

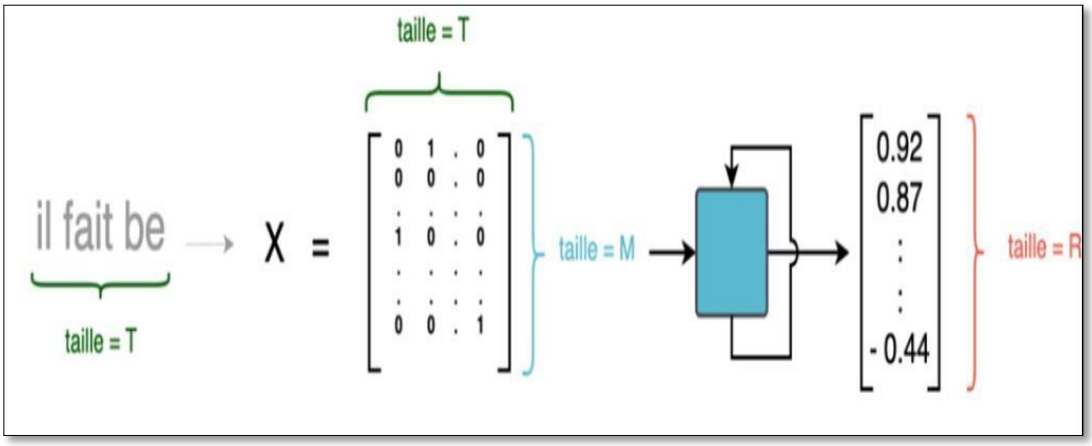

Figure 2.2 – Traitement d'une séquence de taille T=10 par une couche récurrente[19]

Générer le caractère suivant consiste à choisir le caractère le plus pertinent parmi les M=30 caractères possibles. Il faut donc que notre modèle produise une sortie de taille M=30. Ce vecteur de sortie associe alors à chaque caractère une probabilité d'être le suivant dans la séquence, étant donné les T caractères précédents.

Pour obtenir cette distribution de probabilité, on empile une couche de neurones

« Classique » (appelée également couche dense) de taille M=30 à la suite de la couche RNN et on utilise une fonction d'activation softmax. Considérons l'exemple ci-dessous : la séquence d'entrée étant « il fait be », la probabilité de sortie la plus élevée pourrait correspondre à la lettre « a », dans l'optique de générer la phrase « il fait beau »

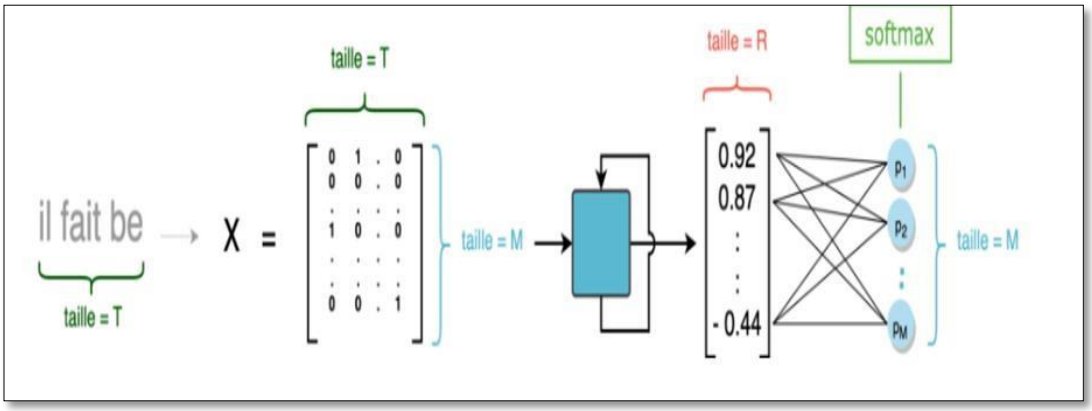

Figure 2.3 – Prédiction de la distribution de probabilité du caractère suivant[19]

Détaillons maintenant ce qui se passe à l'intérieur-même de la couche RNN. Une couche RNN est une succession de T cellules. Chaque cellule a deux entrées :

- ✓ L'élément de la séquence lui correspondant (en version one-hot) : la tème cellule est associée au tème caractère de laséquence
- ✓ Le vecteur en sortie de la cellule précédente. La première cellule, qui n'a pas d'antécédent, prend alors un vecteur initialiséaléatoirement

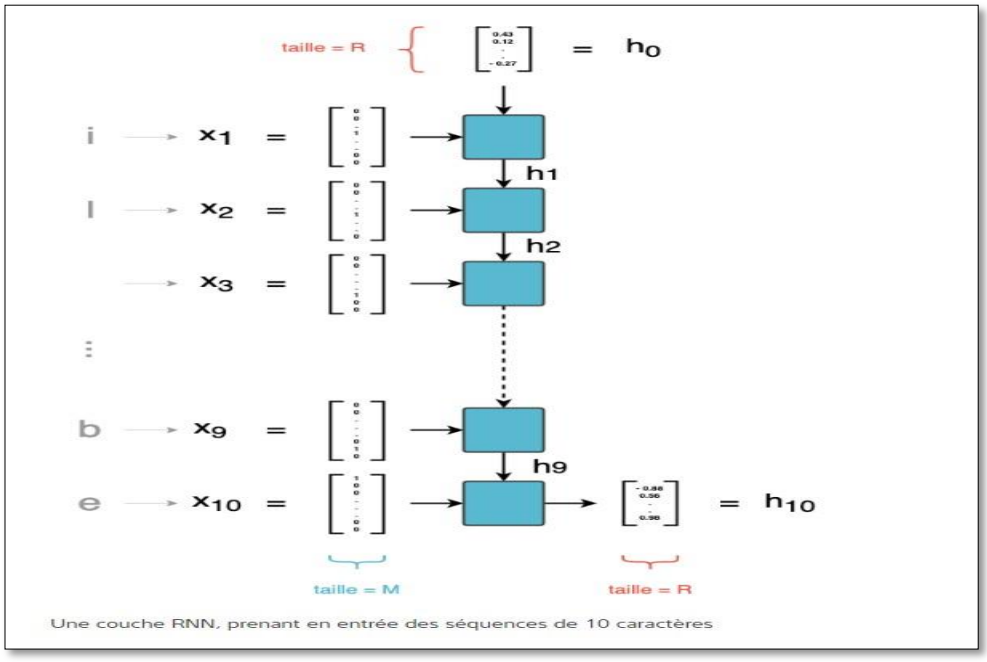

Figure 2.4 – Une couche RNN , prenant en entrée des séquences de 10 caractères[19]

Dans une couche RNN, on parcourt donc successivement les entrées x1 à xT. À l'instant t, la tème cellule combine l'entrée courante xt avec la prédiction au pas précédent ht-1 pour calculer une sortie ht de taille R.

Le dernier vecteur calculé hT (qui est de taille R) est la sortie finale de la couche RNN. Une couche RNN définit donc une relation de récurrence de la forme :

$$\mathbf{ht = f(xt, ht\text{-}1)}$$

2.2.2 Les réseaux convolutifs

Sont une forme particulière de réseau neuronal multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères.

Leur conception suit la découverte de mécanismes visuels dans les organismes vivants. Ces réseaux de neurones artificiels (aussi baptisés réseau de neurones à convolution, ou CNN) sont capables de catégoriser les informations des plus simples aux plus complexes. Ils consistent en un empilage multicouche de neurones, des fonctions mathématiques à plusieurs paramètres ajustables, qui pré-traitent de petites quantités d'informations.

Les réseaux convolutifs sont caractérisé par leurs premières couches convolutionelles (généralement une à trois). Une couche convolutive, est basée comme son nom l'indique sur le principe mathématique de convolution, et cherche à repérer la présence d'un motif (dans un signal ou dans une image par exemple).

Pour une image, la première couche convolutionelle peut détecter les contours des objets (par exemple un cercle), la seconde couche convolutionelle peut combiner les contours en objets (par exemple une roue), et les couches suivantes (non nécessairement convolutionelles) peuvent utiliser ces informations pour distinguer une voiture d'une moto. Une phase d'apprentissage sur des objets connus permet de trouver les meilleurs paramètres en montrant par exemple à la machine des milliers d'images d'un chien, d'une voiture ou d'un sport… L'un des enjeux est de trouver des méthodes pour ajuster ces paramètres le plus rapidement et le plus efficacement possible. Les réseaux neuronaux convolutifs ont de nombreuses applications dans la reconnaissance d'images, de vidéos ou le traitement du langage naturel [19].

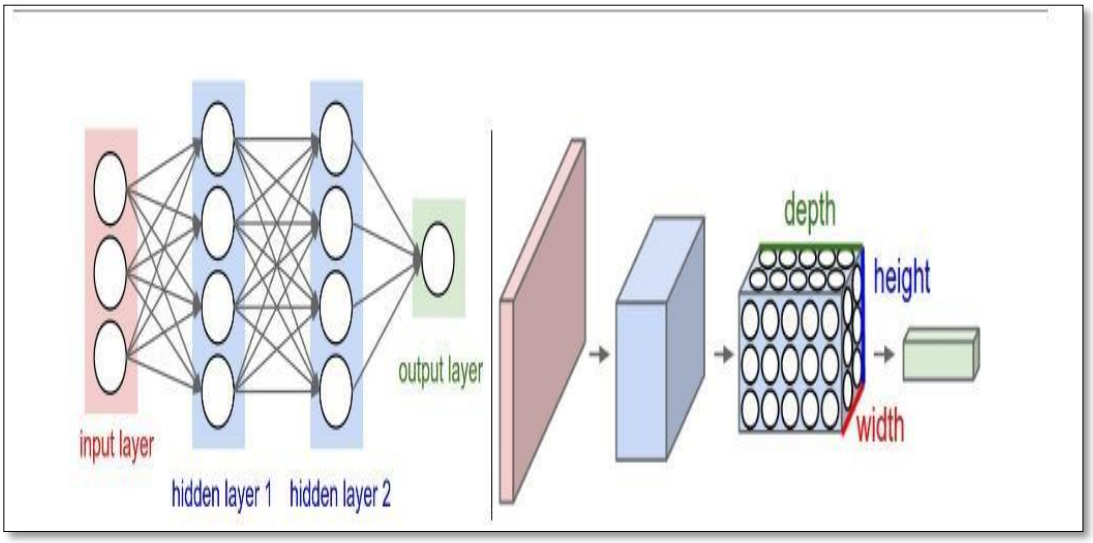

Figure 2.5 – Le réseau CNN [20]

❖ Couche pour construire ConvNet:

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de *pooling*, la couche de correction ReLUet la couche *fully- connected* [20]. Dans ce chapitre, je vais vous expliquer le fonctionnement de ces différentes couches.

a) La couche de convolution

Est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Son but est de repérer la présence d'un ensemble de features dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une fenêtre représentant la feature sur l'image, et de calculer le produit de convolution entre la feature et chaque portion de l'image balayée. Une feature est alors vue comme un filtre : les deux termes sont équivalents dans ce contexte.

La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement auxfeatures que l'on souhaite retrouver dans les images.

On obtient pour chaque paire (image, filtre) une carte d'activation, ou featuremap, qui nous indique où se situent les features dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la feature.

Contrairement aux méthodes traditionnelles, les features ne sont pas pré-définies selon un formalisme particulier (par exemple SIFT), mais apprises par le réseau lors la phase d'entraînement ! Les noyaux des filtres désignent les poids de la couche de convolution. Ils sont initialisés puis mis à jour par rétropropagation du gradient.

b) La couche de pooling:

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs *featuremaps*, et applique à chacune d'entre elles l'opération de *pooling*.

L'opération de *pooling*consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pourcela, on découpe l'image en cellulesrégulière, puis on garde au sein de chaque cellule la valeurmaximale.

En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 ×× 2 pixels qui ne se chevauchent pas, ou des cellules de taille 3 ×× 3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent donc).

On obtient en sortie le même nombre de *featuremaps*qu'en entrée, mais celles-ci sont bien plus petites. La couche de *pooling*permet de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage.

Les valeurs maximales sont repérées de manièremoins exacte dans les *featuremaps*obtenues après *pooling*que dans celles reçues en entrée – c'est en faitun grand avantage ! En effet, lorsqu'on veut reconnaître un chien par exemple, ses oreilles n'ontpasbesoin d'être localisées le plus précisément possible : savoir qu'elles se situent à peu près à côté de la tête suffit !

Ainsi, la couche de *pooling*rend le réseau moins sensible à la position des *features*: le fait qu'une *feature*se situe un peu plus en haut ou en bas, ou même qu'elle ait une orientation légèrement différente ne devrait pas provoquer un changement radical dans la classification de l'image.

c) La couche de correction ReLU:

ReLU (RectifiedLinearUnits) désigne la fonction réelle non-linéaire définie parReLU(x)=max(0,x)ReLU(x)=max(0,x).

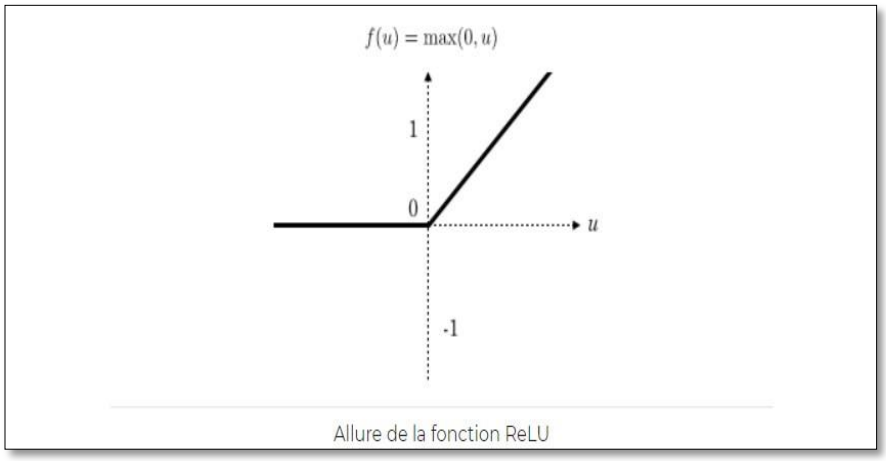

Figure 2.6 – Allure de la fonction ReLu[20]

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

d) La couchefully-connected

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non – elle n'est donc pas caractéristique d'un CNN.

Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille NN, où NN est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

Par exemple, si le problème consiste à distinguer les chats des chiens, le vecteur final sera de taille 2 : le premier élément (respectivement, le deuxième) donne la probabilité d'appartenir au classe "chat" (respectivement "chien"). Ainsi, le vecteur [0.90.1][0.90.1] signifie que l'image a 90% de chances de représenter unchat.

Chaque valeur du tableau en entrée "vote" en faveur d'une classe. Les votes n'ont pas tous la même importance : la couche leur accorde des poids qui dépendent de l'élément du tableau et de la classe.

Pour calculer les probabilités, la couche fully-connected multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (logistique si N=2N=2, softmax si N>2N>2) :

Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme fully-connected.

La couche fully-connected détermine le lien entre la position des features dans l'image etune classe. En effet, le tableau en entrée étant le résultat de la couche précédente, il

correspond à une carte d'activation pour une feature donnée : les valeurs élevées indiquent la localisation (plus ou moins précise selon lepooling) de cette featuredans l'image. Si la localisation d'une feature à un certain endroit de l'image est caractéristique d'une certaine classe, alors on accorde un poids important à la valeur correspondante dans le tableau.

2.3 Prédiction de chaos en météo en utilisant le deep-learning

2.3.1 Limitations et contrôles de chaos:

Bien que les systèmes chaotiques aient de bonnes caractéristiques qui conviennent à l'analyse de comportements complexes, il existe des facteurs importants qui empêchent de prédire avec précision le comportement d'un système complexe. Ils incluent une dépendance sensible aux conditions initiales qui sont dans la plupart des cas inconnues car la plupart des hypothèses émises conduisent souvent à des erreurs.

Les chaos peuvent être contrôlés afin de réduire les erreurs de calcul dues aux effets néfastes des limitations. La littérature suggère qu'il est très nécessaire de vérifier le chaos dans un ensemble de données donné avant de faire des prédictions. La raison de la vérification est qu'il pourrait y avoir présence de données aléatoires, qui sont souvent supposées être chaotiques, dans l'ensemble de données.

2.3.2 Approche Théorique:

« Les prévisions climatiques à long terme sont impossibles.» En 2007, ce fut la conclusion du mathématicien et météorologiste Edward Lorenz, qui a renversé l'idée que l'univers est aussi prévisible qu'une horloge avec sa théorie révolutionnaire du chaos. Aujourd'hui, Lorenz serait stupéfait par les progrès technologiques que nous avons réalisés, ce qui pourrait infirmer sa déclaration sur les prévisions à long terme.

Cette théorie ne concerne pas juste les prévisions climatiques, mais aussi tous les systèmes dynamiques, dans notre cas on va essayer d'utiliser le Lstm pour prédire les changementsclimatiques.

2.3.3 Approche Technique :

❖ Architecture deLSTM:

Les réseaux de neurones classiques reposent sur le fait que les données n'ont aucun ordre lors de leur entrée dans le réseau et la sortie dépend uniquement des caractéristiques d'entrée. Dans le cas où la sortie dépend des caractéristiques et des sorties précédentes, le réseau FFN classique ne peut pas aider. La solution à ce problème peut être réseau neuronal qui peut être récursivement fournit les sorties précédentes. Alors on parle de RNN.

La sortie courant du réseau récurrent est définie par l'entrée courant Xt, et aussi sur les états liés aux sorties réseau précédentes ht-1, ht-2,…

Le concept de réseau neuronal récurrent est simple et facile à implémenter, mais le problème se pose pendant l'apprentissage, un comportement de gradient imprévisible, le problème de gradient du réseau neuronal peut être résumé en deux catégories : the vanishing and the exploding gradient[16].

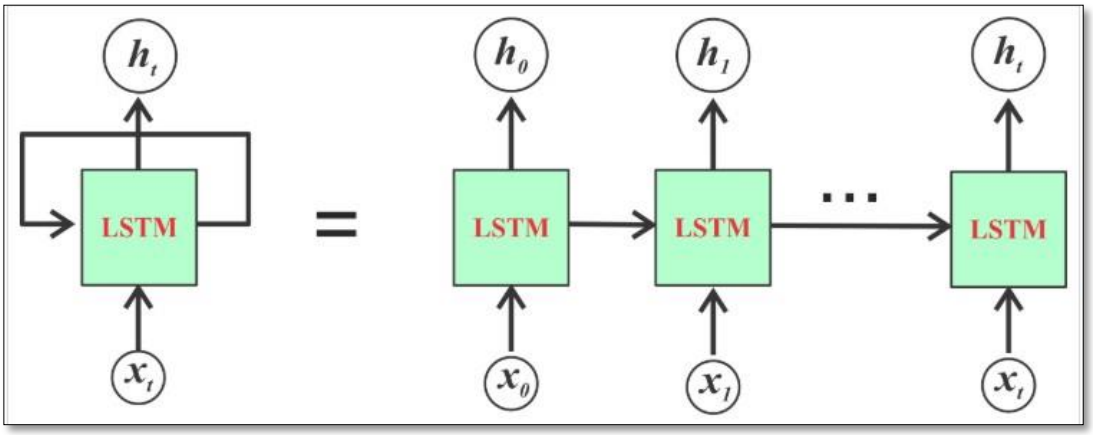

Figure 2.7 – Le réseau LSTM

La sortie courant du réseau récurrent est définie par l'entrée courant Xt, et aussi sur les états liés aux sorties réseau précédentes ht-1, ht-2, …

Le concept de réseau neuronal récurrent est simple et facile à implémenter, mais le problème se pose pendant l'apprentissage, un comportement de gradient imprévisible, le problème de gradient du réseau neuronal peut être résumé en deux catégories : the vanishing and the exploding gradient. Le réseau neuronal récurrent est basé sur l'algorithme de rétropropagation, spécialement développé pour l'ANN récurrente, qui est appelé

Rétropropagation envers le temps (BPTT), En cas vanishing gradient problem, les mises à jour des paramètres sont proportionnelles au gradient de l'erreur, qui dans la plupart des cas négligeable et les résultats que les poids correspondants sont des constants et empêcher le réseau de poursuivre son apprentissage.

D'autre coté, exploding gradient problem fait référence au comportement opposé, où les mises à jour des poids sont devenues importantes à chaque étape de rétro-propagation.

Ce problème est causé par l'explosion des composants à long terme dans le réseau neuronal récurrent. La solution aux problèmes ci-dessus est la conception spécifique du réseau actuel appelé Lstm. L'un des principaux avantages du LSTM est qu'il peut fournirun flux d'erreur constant. La cellule LSTM contient un ensemble de blocs de mémoire, qui ont la capacité de stocker l'état temporel du réseau. Le LSTM possède également des unités multiplicatives spéciales appelés portes qui contrôlent le flux d'informations.

La cellule LSTM consiste à :

- ✓ Porte d'entrée - qui contrôle le flux des activations d'entrée dans la cellule demémoire.
- ✓ Porte de sortie qui contrôle le flux de sortie de l'activation de la cellule.
- ✓ Oublier la porte, qui filtre les informations de l'entrée et de la sortie précédente et décide lequel doit être retenu ou oublié etabandonné.
- ✓ Outre trois portes, la cellule LSTM contient unemise à jour de cellule qui est généralement une couche pour faire partie de l'état de lacellule.
- ✓ Dans chaque cellule LSTM, les trois variables entrent dans la cellule:
    - L'entrée courant ext,
    - Sortie précédente ht-1et
    - État précédent des cellules ct-1.
- ✓ D'autre part, de chaque cellule LSTM deux variables sortent:
    - La sortie de courant htet
    - L'état actuel de la cellulect.

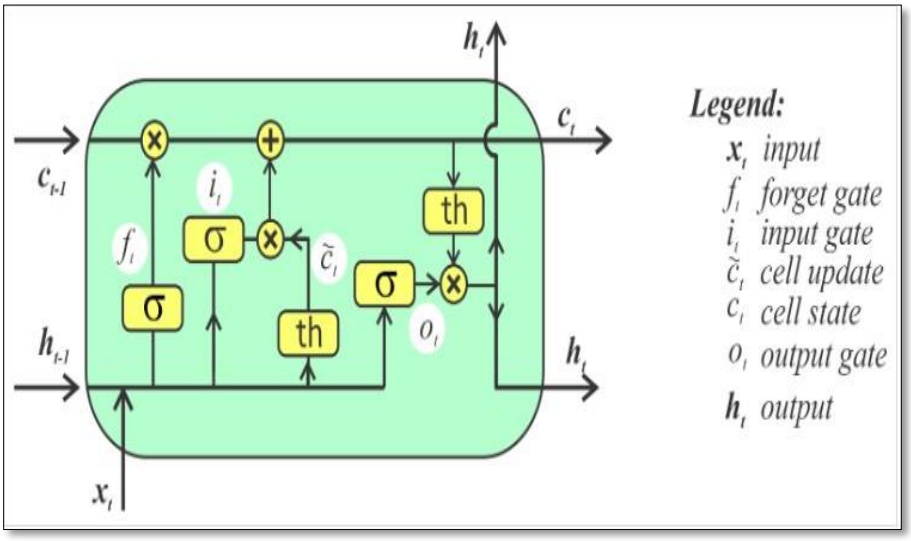

Figure 2.8 – Fonction de LSTM

Afin d'implémenter le réseau récurrent LSTM, d'abord la cellule LSTM doit être implémentée. La cellule LSTM a trois portes, et deux états internes qui doit être déterminé afin de calculer la sortie actuelle et l'état actuel de la cellule.

La cellule LSTM peut être définie comme un réseau neuronal où le vecteur d'entrée $x = (x_1, x_2, x_3, \ldots x_t)$au temps t, correspond au vecteur de sortie $y = (y_1,\ y_2,\ \ldots, y_m)$, grâce au calcul des couches suivantes :

- La couche sigmoïde de la porte d'oubli pour le temps t, ftest calculée par la sortie précédente ht-1 du vecteur d'entrée xt, et la matrice de poids de la couche oublier Wf avec addition du biais correspondantbi:$f_t = \sigma\left(W_f \bullet [h_{t-1}, x_t] + b_f\right).$

- La couche d'entrée sigmoïde de porte pour le temps t, il est calculé par le précédent sortie ht-1 le vecteur d'entrée xt et la matrice de poids de la couche d'entrée Wi avec ajout du biais correspondantbi: $i_t = \sigma\left(W_i \bullet [h_{t-1}, x_t] + b_i\right).$

- l'état de la cellule au temps t, Ct est calculé à partir de la ported'oubli ftet l'état de

cellule précédent Ct-1. Le résultat est additionné avec la porte d'entrée et l'état de mise à jour de la cellule $\tilde{c}_t$,, c'est-à-dire la couche tanh calculée par la sortie précédente ht-1 du vecteur d'entrée xt,etla matrice de poids pour la cellule avec addition du biais correspondant bi:
$C_t = f_t \otimes C_{t-1} + i_t \otimes \tanh\left(W_C \bullet [h_{t-1}, x_t] + b_C\right).$

- lacouche de sortie sigmoïde de porte pour le temps t, otest calculé par le précédent sortie ht-1, le vecteur d'entrée xtet la matrice de poids de la couche de sortie Woavec addition du biais correspondantbi:
$o_t = \sigma\left(W_0 \bullet [h_{t-1}, x_t] + b_0\right).$

- La dernière étape de la cellule LSTM est le calcul du courant de sortie ht. lasortie courante est calculée avec l'opération de multiplication $\otimes$entre la couche de grille de sortie et la couche tanh de l'état actuel des cellulesCt. $h_t = o_t \otimes \tanh(C_t)$.

❖ LSTM bidirectionnels :

Les LSTM bidirectionnels sont une extension des LSTM traditionnels qui peuvent améliorer les performances du modèle sur les problèmes de classification de séquence.
Dans les problèmes où tous les pas de temps de la séquence d'entrée sont disponibles, ils forment deux LSTM au lieu d'un sur la séquence d'entrée. Le premier sur la séquence d'entrée en l'état et le second sur une copie inversée de la séquence d'entrée.

❖ Implémentation de LSTM:

Evolution superbe de python dans les domaines de intelligence artificielle nous donne des grandes avantages aident à créer des projets avec des bibliothèques, APIs et interfaces déjà existent comme:

- **Keras**: est une API conçue pour les êtres humains, pas pour les machines. Keras suit les meilleures pratiques pour réduire la charge cognitive: il propose des API simples et cohérentes, il minimise le nombre d'actions utilisateur requises pour les cas d'utilisation courants et il fournit des messages d'erreur clairs et exploitables. Il dispose également d'une documentation complète et de guides de développement[13].

- **Tensorflow :** est une bibliothèque logicielle gratuite et open-source pour le flux de données et la programmation différenciable à travers une gamme de tâches. C'est une bibliothèque mathématique symbolique, et est également utilisée pour des applications d'apprentissage automatique telles que les réseaux de neurones [17].

- **Matplotolib :** est une bibliothèque complète pour créer des visualisations statiques, animées et interactives enPython

```
#  Keras libraries
#
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Bidirectional

#
# For data conditioning
#
from scipy.ndimage import gaussian_filter1d
from scipy.signal import medfilt

#
# Make results reproducible
#
from numpy.random import seed
seed(1)
import tensorflow
tensorflow.random.set_seed(1)
```

1) Ensuite, nous voulons simplement appeler les autres bibliothèques nécessaires au programme. En particulier, nous voulons inclure une fonction de normalisation pour préparer nos données pour la formation. La normalisation est importante pour rendre notre modèle plus efficace.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from numpy import array
```

2) Ici, vous pouvez définir le n_timestamp, le nombre timestamps (dans ce cas, jours) à utiliser comme entrée pour la prédiction. C'est important car nous voulons considérer la quantité de données que notre modèle devrait vérifier avant de faire une prédiction. Pour l'apprentissage, nous avons les n_periode, le nombre de période nécessaires à l'apprentissage. Comme nous avons pu tester ce programme au préalable, environ 25 époques suffisaient déjà pour entraîner le modèle sans overtraining.

```
n_timestamp = 10
train_days = 1500  # nombre de jours pour l'apprentissage
testing_days = 500 # nombre de jours pour predictions
n_epochs = 25
filter_on = 1
```

3) On va importer le data set pour faire µl'apprentissage

```
url = "data/weather.csv"
dataset = pd.read_csv(url)
```

4) On effectue un filtre médian et un filtre gaussien sur l'ensemble de données.

```
77  if filter_on == 1:
78      dataset['Temperature'] = medfilt(dataset['Temperature'], 3)
79      dataset['Temperature'] = gaussian_filter1d(dataset['Temperature'], 1.2)
80
```

5) On doit définir notre ensemble de données d'apprentissage et de test en fonction de nos variables définies

```
train_set = dataset[0:train_days].reset_index(drop=True)
test_set = dataset[train_days: train_days+testing_days].reset_index(drop=True)
training_set = train_set.iloc[:, 1:2].values
testing_set = test_set.iloc[:, 1:2].values
```

6) Normaliser lesdonnées

```
100  sc = MinMaxScaler(feature_range = (0, 1))
101  training_set_scaled = sc.fit_transform(training_set)
102  testing_set_scaled = sc.fit_transform(testing_set)
103
104
```

7) Nos données sont réparties en fonction du nombre detimestamps

```
109  def data_split(sequence, n_timestamp):
110      X = []
111      y = []
112      for i in range(len(sequence)):
113          end_ix = i + n_timestamp
114          if end_ix > len(sequence)-1:
115              break
116          # i to end_ix as input
117          # end_ix as target output
118          seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
119          X.append(seq_x)
120          y.append(seq_y)
121      return array(X), array(y)
122
123
124  X_train, y_train = data_split(training_set_scaled, n_timestamp)
125  X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
126  X_test, y_test = data_split(testing_set_scaled, n_timestamp)
127  X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

8) Construire le model LSTM

```
if model_type == 1:

    model = Sequential()
    model.add(LSTM(units = 50, activation='relu',input_shape = (X_train.shape[1], 1)))
    model.add(Dense(units = 1))
if model_type == 2:

    model = Sequential()
    model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(X_train.shape[1], 1)))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(1))
if model_type == 3:

    model = Sequential()
    model.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(X_train.shape[1], 1)))
    model.add(Dense(1))

#
# Start training
#
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
history = model.fit(X_train, y_train, epochs = n_epochs, batch_size = 32)
loss = history.history['loss']
epochs = range(len(loss))
```

2.4. Prédiction de position d'échecs

Représentation classique de l'échiquier:

On a plusieurs façons pour représenter l'échiquier, mais la plus connue c'est la présentation bitboard qui consiste à représenter chaque pièce avec 12 symboles binaires,la case de l'échiquier va représenter par -1 s'il contient une pièce noire , par 1 si la pièce est blanche ,par 0 si la case est vide .Cette représentation est une séquence binaire de bits de longueur 768 capable pour représenter la position d'échecs complète, Il existe en fait 12 types de pièces différents et 64 au total cases, ce qui donne 768 entrées.

Évaluation

A. Structure depions

- Pénaliser les pions doublés, reculés et bloqués.
- Encouragez l'avancement du pion là où il est correctement défendu.

- Encouragez le contrôle du centre du tableau. de position : l'évaluation prend plusieurs facteur en considération .

B. Placement de pièces

- Encouragez les chevaliers à occuper le centre du plateau.
- Encouragez les reines et les freux à se défendre et à attaquer.
- Encouragez les attaques de 7e rang pour les tours.

C. Pions passés

- Ceux-ci méritent un traitement spécial car ils sont si importants.
- Vérifiez la sécurité du roi adverse et des pièces ennemies.
- Ajoutez des incitations énormes pour les pions passés près de la promotion.

D. Sécurité de roi

- Encouragez le roi à rester au coin du jeu.
- Essayez de conserver un bouclier de pion efficace.
- Essayez d'empêcher les pièces ennemies de s'approcher du roi.

L'algorithme de minimax

Minimax est un algorithme décisionnel, généralement utilisédans un jeu au tour par tour, à deux joueurs. Le but de l'algorithme est de trouver le prochain coup optimal.

Dans l'algorithme, un joueur est appelé le maximiseur et l'autre joueur est un minimiseur. Si nous attribuons un score d'évaluation au tableau de jeu, un joueur tente de choisir un état de jeu avec le score maximum, tandis que l'autre choisit un état avec le score minimum.

En d'autres termes, le maximiseur permet d'obtenir le score le plus élevé, tandis que le minimiseur tente d'obtenir le score le plus bas en essayant de contrer les mouvements.Il est basé sur le concept zero-sum. Dans un jeu à somme nulle, le score total d'utilité est divisé

entre les joueurs. Une augmentation du score d'un joueur entraîne une diminution du score d'un autre joueur. Le score total est donc toujours égal à zéro.

Pour qu'un joueur gagne, l'autre doit perdre. Des exemples de tels jeux sont les échecs, le poker, les dames, le tic-tac-toe.

Un fait intéressant : en 1997, l'ordinateur de jeu d'échecs Deep Blue d'IBM (construit avec Minimax) a battu Garry Kasparov (champion du monde des échecs).

Notre objectif est de trouver le meilleur coup pour le joueur. Pour ce faire, nous pouvons simplement choisir le nœud avec le meilleur score d'évaluation. Pour rendre le processus plus intelligent, nous pouvons également regarder de l'avant et évaluer les mouvements de nos adversaires potentiels.

Pour chaque mouvement, nous pouvons prévoir autant de mouvements que notre puissance de calcul le permet. L'algorithme suppose que l'adversaire joue de manière optimale [9].

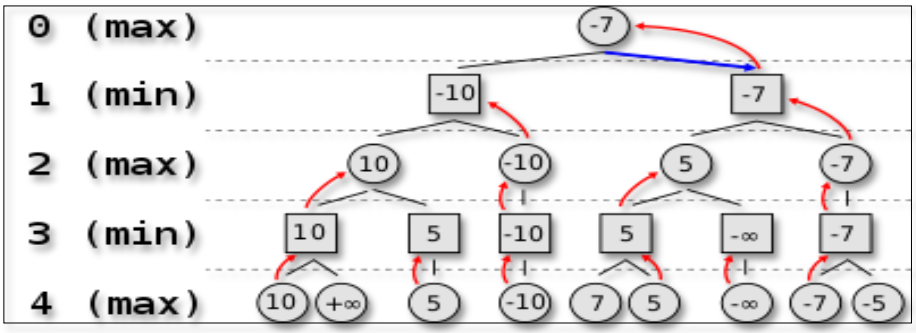

Figure 2.9 – algorithme de minimax_1[25]

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Figure 2.10 – Algorithme minmax_1

L`élagage Alpha-Beta

On observe que l`algorithme MINIMAX effectue l`évaluation pour tous les nœuds de l`arbre de jeu d`un horizon donné. Mais il existe des situations dans lesquelles, pour déterminer la valeur Minimax associée à la racine, il n'est pas nécessaire de calculer les valeurs associées à tous les nœuds de l'arbre. On peut remarquer une telle situation dans la figure 3.4.3, où la racine a un fils avec la valeur maximale possible (+100), donc les valeurs des autres fils n'importent pas : elles ne peuvent pas changer la valeur MiniMax

L`algorithme d`élagage Alpha-Beta conduit à une économie de temps de calcul et mémoire, en renonçant à l`évaluation des sous arbres dès que leur valeur devient inintéressante pour le calcul de la valeur associée aux nœuds père de ces sous arbres.

Pour réaliser l`algorithme Alpha-Beta on va associer à chaque nœud n de type MAX une valeur auxiliaire appelée Alpha(n) égale à la valeur de f(n) si n est un nœud terminal, ou à la meilleur valeur de ses fils trouvée jusqu`ici;

une valeur Beta(m) pour les nœuds MIN, valeur égale à f(m), si m est un nœud terminal, ou à la valeur minimum de ses successeurs trouvés jusqu`ici

Parce que dans un arbre de jeu les niveaux des nœuds de type MAX et MIN s`alternent, on a les relations suivantes

Alpha(n) = max {Beta(fl), …, Beta(fk) }

Beta(m) = min {Alpha(j1), .. Alpha(jk) } Beta(fi) <= Alpha (Père(fi))

Alpha(fi) >= Beta (Père(fi))

Où fi et ji sont respectivement les fils de n et de m

Dans l`exécution de l`algorithme Alpha-Beta coupe les sous arbres selon les règles suivantes :

on interrompt la recherche d`un nœud n de type MAX si Alpha(n) ³ Beta(Père(n)), parce que c`est une valeur inintéressante qui ne peut pas changer la valeur Beta du nœud père de n (qui est un nœud de typeMIN);

On interrompt la recherche d`un nœud m de type MIN si Beta(m) £ Alpha (Père(m)) parce que cette valeur ne peut pas influencer le processus de détermination du maximum qui établit la valeur Alpha de sonpère.

De cette façon, l`algorithme Alpha-Beta ne génère plus les successeurs d`un nœud dès qu`il est évident que, suite aux propriétés des fonctions minimum et maximum et aux valeurs associées aux nœuds déjà générés, ce nœud ne sera pas pris en compte pour le choix du coup.

L`algorithme Alpha-Beta, présenté par le pseudo code qui suit, réalise la construction de l`arbre de jeu en partant d`une position p, à l`aide de l`ensemble d`opérateurs, mais sansdévelopper les sous arbres inintéressants [8].

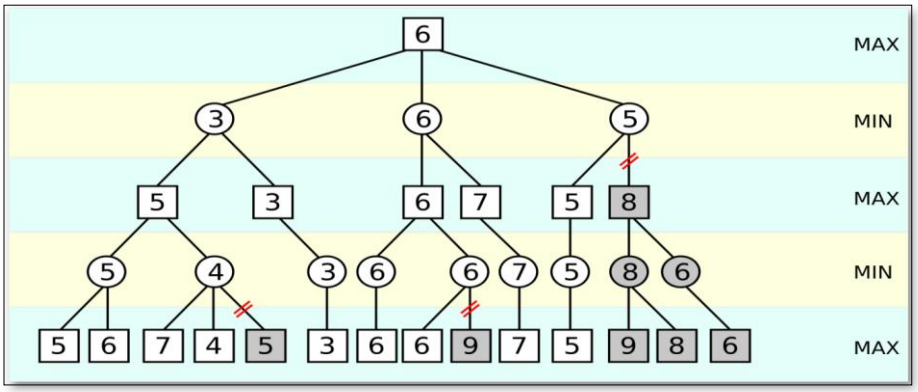

Figure 2.11 – Algorithmeminmax_2

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if α ≥ β then
                break (* α cut-off *)
        return value
```

Figure 2.12 – Algorithme minmax_2

Générateur de coup (fonction itérative):

C'est une fonction itérative qui aide a générer un liste de coup légal et pseudo coup légal.

Dataset : On utilise un grand nombre de partie entre des grands joueurs, et ça nous aide à faire training optimal.

2.4.1 Approche théorique : On va voir un autre type de système dynamique le jeu d'échec. Les échecs est un système dynamique dans lequel le jeu lui-même peut changer grâce a influence des comportements et des états des joueurs.

le jeux d'échec comme un système dynamique passe de la position de départ à un échec et mat et ça se qu'on appelle système dynamique non-equilibruim, la position va changer après chaque coup, et chaque deux coup snous donnes différents on a deux positions différentes complètement [10], [15]. Alors on parle de la sensibilité au conditions initiales.

Notre objectif consiste à exploiter l'avancement de l'intelligence artificielle pour maitriser un moteur d'échec a prédire les coups et déterminer les variantes possibles

2.4.2 Approche téchnique

1) Représentation classique de l'échiquier:

On a six différents types de pièces et 64 cases donc la représentation dans notre model ça sera (64*6), chaque pièce a un spécifique poids (valeur), Nous avons également choisi d'utiliser une couche pour les deux couleurs, +1 pour indiquer pièce amie et -1 pour désigner les pièces adverses, en utilisant 12 couches pour représenter chaque morceau des deux couleurs feraient les données clarté.

On utilise plusieurs dataset pour l'apprentissage de réseaux, une pour le sélecteur de coup et les autres pour sélecteur de pièce.

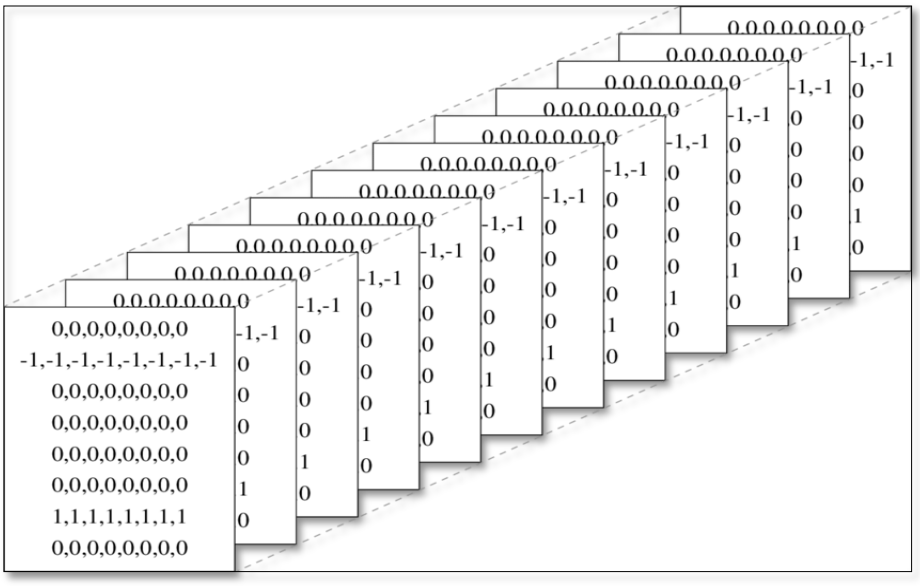

Figure 2.13 – l'échiquier au model

2) L'architecture de réseau de neurones convolutifs

Les sept réseaux prennent comme entrée la représentation des échecs décrit ci-dessus et produit une distribution de probabilité 8*8.

Nous utilisons un réseau neurones convolutif à cinq couches de la forme [conv-relu] x2-pool-conv-relu-affine-relu-affinesoftmax.

Nous soulignons la nécessité de deux couches affines afin que les caractéristiques de bas niveau puissent être accompagnées de évaluations logiques globales.

3) Prétraitement

L'ensemble de données FICS comprend des parties d'échecs au format PGN, qui enrôle chaque mouvement joué par les deux joueurs dans notation algébrique d'échecs (ACN). Pour que cet ensemble de données être pertinent pour notre projet, la notation ACN a dû être convertie à deux coordonnées représentant la position d'une pièce a été déplacé et laposition vers laquelle une pièce a été déplacée. Ces coups ont ensuite été joués sur un échiquier pour obtenir la représentation du conseil d'administration à chaque position. Les deux jeux

d'étiquettes utilisés sont alors les coordonnées d'un morceau a été retiré de et la coordonnée dans laquelle la pièce a été déplacée.

4) Apprentissage

En utilisant des représentations d'échiquier de taille 8*8*6, nous formons les sCNN et eCNN à chaque coup dans le jeu. À chaque coup, nous ne formons qu'un des eCNN. L'eCNN sera déterminé par la pièce déplacée dans la précédente position. Nous avons initialisé les poids de façon aléatoire en utilisant l'approche fan-in et fan-out. Cela garantirait que la symétrie est rompue et que lales poids sont effectivement mis à jour pour refléter la formationdu

Modèle. On n'utilise pas la régularisation comme concept de Lisser l'activation des entrées n'est pas applicable pour l'image d'échecs et pourrait briser une grande partie de la logique d'échecs.

5) Test

Lors des tests, on exécute l'entrée sur les sCNN et eCNN pour produire les distributions de probabilité pour chaque réseau de neurone. Pour chaque pièce de l'sCNN, nous faisons ensuite référence à distribution de probabilité de l'eCNN et mettre à 0 tous les cases qui représente coup illégal. Nous multiplions la valeur maximale de cela par la valeur de l'sCNNet l'enregistrer et associé dans et hors coordonnées, on choisit les coordonnés internes et les externes avec maximum probabilité et la sortie de coup.

2.4.3 Conclusion

Dans ce chapitre, nous avons présenté deux differents systèmes dynamiques et nous avons montré l'efficacité de l'apprentissage profond pour améliorer le comportement des systèmes dynamiques et pour éviter le chao.

CHAPITRE 3

Résultats & Discussion

3.1 Introduction

Dans ce dernier chapitre, nous présentons notre résultat obtenu de deux cas d'étude ainsi qu'une discussion.

3.2 Résultat pour cas d'étude 1:

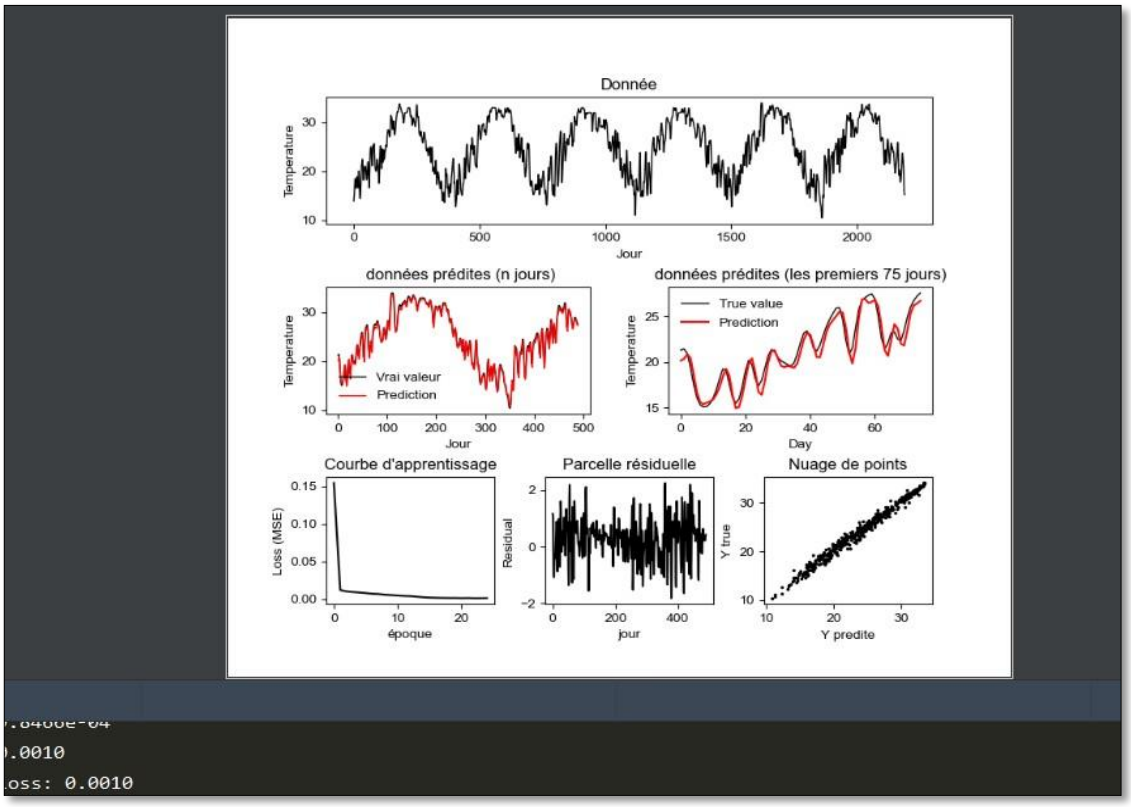

Figure 3.1 – Résultat pour cas d'étude 1

D'après les résultats, nous pouvons voir que notre prédiction du modèle a réussi.Mais, il peut être observé à partir des prévisions (n jours), Les erreurs proviennent généralement de l'augmentation ou du déclin inattendu des données, par exemple 350-360 jours. Mais, sur la base des 75 premiers jours, le modèle peut suivre correctement le modèle des données.

	R^2			MEC		
Jours	Seule cellule	stacked	Bidirectionnel	Seule cellule	stacked	Bidirectionnel
5	0.97	0.96	0.97	1.1	1.2	0.86
10	0.98	0.98	0.97	0.81	0.57	0.84
25	0.97	0.98	0.97	1.06	0.78	0.9
50	0.97	0.97	0.96	1.01	1.08	1.33
100	0.98	0.95	0.9	0.53	1.79	3.66

Tableau 3.1 – Résultat pour cas d'étude

On remarque du tableau que le LSTM stacked a obtenu les meilleurs résultats par rapport aux autres types, Nous pouvons également voir que la cellule unique fonctionnait très bien en utilisant 100 jours d'entrée, mais nous avons trouvé. Le LSTM bidirectionnel a également obtenu de moins bons résultats avec plus de jours d'entrée.

3.3 Résultat pour cas d'étude 2:

Figure 3.2 – Résultat pour cas d'étude 2

Nous avons réussi à prétraiter les échecs jeux de données de jeu au format PGN dans une représentation qui convient nos exigences de modèle et ça n'était pas facile car le format de fichier d'un partie jeux d'échec (pgn) vise les humains pas les machines alors pour résoudre ce problème nous avons utilisé le module pgn-parser. On peut aussi remarquer que les pieces avec lourd poids a un grand niveaud'occurrence.

position	Coup prédit	Meilleur coup
1	Qb3	E6
2	E6	E5
3	Kf3	Kf3
4	B4	Qc2
5	B5	B5
6	Kxc6	Qd2

3.4 Les outils utilisés:

1) **Pycharm** est un environnement de développement intégré utilisé pour programmer enPython.

   Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains,

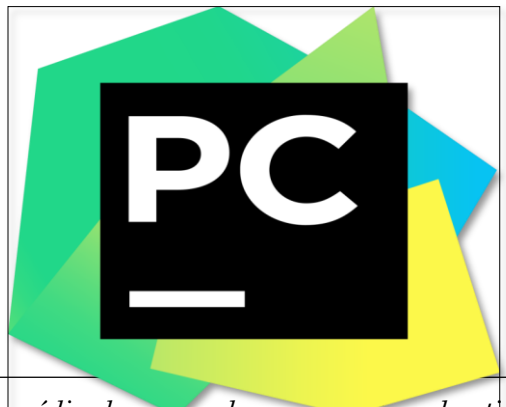

Figure 3.3 – Logo Pycharm

2) **Vscode** est unéditeur de codeextensible développé par Microsoft pour Windows, Linux etmacOS

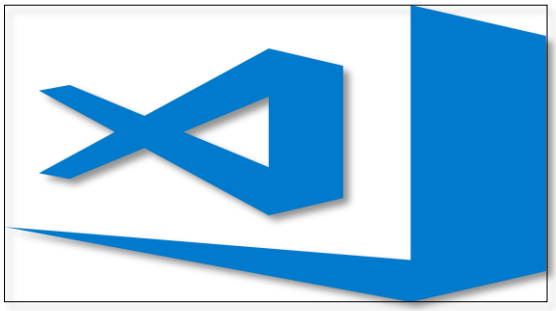

Figure 3.4 – Logo Visual Studio

3) **PlayWithArena :** desktop et plateforme on ligne permet de tester les moteurs d'échec(http://www.playwitharena.de/).

3.5 Conclusion :

Les tests effectués montrent une grande supériorité des approches proposées, par rapport à l'utilisation d'autres techniques. De plus, les résultats obtenus en appliquant ces techniques sur les cas sont excellents : une amélioration significative est apportée, comparativement aux meilleurs résultats trouvés par les autres méthodes.

Conclusion Générale

Conclusion Générale

Dans les dernières années, Le Deep Learning démontre que les méthodes et les techniques d'apprentissage pointent dans plusieurs domaines, tel que les systèmes dynamiques non linéaire, il confirme son évolution énorme par rapport à d'autre domaine comme la recherche opérationnel par exemple.

Et çamèneauxchangementsradicauxqu'onvavoirdans lesprochainesannées.Danscetravail, nous avons appliqué certaines techniques de Deep Learning habituelles dans ce domaine, mais avec des nouvelles approches. Et avec les résultats qu'on a obtenus démontre que Le LSTM est un outil très efficace a prédiction à côté de plusieurs autres techniques de DeepLearning.

L'approche de trouver le chaos dans le moteur d'échecs à besoins beaucoup de développements aux futures par exemple relier les coups des fous et cavaliers avec les dimensions d'échiquier, puis étudier les bifurcations de ses déplacements et traiter les comme des fonctions non linéaires et ça aide à trouver des coups plus intéressants, c'est une probabilité possible lorsqu'on obtient vaste surface de donnée.

Les tests effectués montrent une grande supériorité des approches proposées, par rapport à l'utilisation d'autres techniques. De plus, les résultats obtenus en appliquant ces techniques sur les cas sont excellents : une amélioration significative est apportée, comparativement aux meilleurs résultats trouvés par les autres méthodes.

[1] Dialogues Clin Neurosci. 2007 Sep; 9(3): 279–289.

[2] Strogatz, S. H. (2001). *Nonlinear Dynamics and Chaos: with Applications to Physics, Biology and Chemistry*.

[3] James Meiss (2007), Scholarpedia, 2(2):1629.

[4] Chaves, Julio (2015). *Introduction à l' optique non imageur, deuxième*

édition . CRC Press . ISBN 978-1482206739 .

[5]Robert C. Hilborn, Sea gulls, butterflies, and grasshoppers: A brief history of the butterfly effect in nonlinear dynamics. American Journal of Physics. 72(4):425. 2004.

[6] Phase Space Paperback – August 31, 2003 by Stephen Baxter

[7] Phase Space Paperback – August 31, 2003 by Stephen Baxter

[8]http://turing.cs.pub.ro/auf2/html/chapters/chapter3/chapter_3_4_3.html

[9] https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

[10] Nonlinear dynamics in combinatorial games: Renormalizing Chomp

Eric J. Friedman and Adam ScottLandsberg.

[11] https://www.stormgeo.com/solutions/cross-industry/articles/will-machine-learning-make-us-rethink-chaos-theory/

[12] http://www.univ-tebessa.dz/fichiers/masters/sesnv160039.pdf

[13]https://keras.io/

[14] https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

[15] Optimal Dimensionality Reduction of Complex Dynamics:

The Chess Game as Diffusion on a Free Energy Landscape Sergei V. Krivov .

[16] https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/

[17] https://www.tensorflow.org/api_docs/python/

[18] https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

[19]https://dataanalyticspost.com/Lexique/reseau-de-neurones-convolutifs/

[20] https://openclassrooms.com/en/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun

[21] https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/

[22] N. Witkowski and P. Bergé. Le chaos. Magazine Scientifique Européen Archimède, 13 Janvier 1998.

[23] C. Morel, Analyse et contrôle de dynamiques chaotiques, application à des circuits électroniques non-linéaires, Université D'angers, 6 décembre 2005, pp.1

[24] The Topology of Chaos: Alice in Stretch and Squeezeland 2nd Edition ISBN-13: 978-3527410675.

[25] http://en.wikipedia.org/wiki/Image:Minimax.svg