Vers une utilisation de l'Intelligence Artificielle dans un modèle numérique de climat

Blanka Balogh

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Océan, atmosphère, climat

Présentée et soutenue par :

Mme BLANKA BALOGH

le vendredi 7 octobre 2022

Titre :

Vers une utilisation de l'Intelligence Artificielle dans un modèle numérique de climat

---

Ecole doctorale :

Sciences de l'Univers de l'Environnement et de l'Espace (SDU2E)

Unité de recherche :

Groupe de Météorologie Expérimentale et Instrumentale (CNRM-GMEI)

Directeur(s) de Thèse :

M. DAVID SAINT-MARTIN

M. AURELIEN RIBES

Rapporteurs :

MME VALERIE MONBET, UNIVERSITE RENNES 1

M. PIERRE GENTINE, NEW YORK UNIVERSITY

Membre(s) du jury :

M. FABRICE GAMBOA, UNIVERSITE PAUL SABATIER, Président

M. AURELIEN RIBES, METEO FRANCE TOULOUSE, Membre

M. CORENTIN LAPEYRE, CERFACS, Invité(e)

M. DAVID SAINT-MARTIN, METEO FRANCE TOULOUSE, Membre

M. TOM BEUCLER, UNIVERSITE DE LAUSANNE, Membre

- OK. Voilà. Klara, le fait est qu'il existe aujourd'hui une inquiétude croissante et généralisée à propose des AA. Les gens disent que vous êtes devenus trop intelligents. Ils ont peur parce qu'ils ne peuvent plus suivre ce qui se passe à l'intérieur. Ils voient ce que vous faites. Ils acceptent que vos décisions, vos recommandations, soient raisonnables et fiables, presque toujours justes. Mais ça leur déplait de ne pas savoir comment vous vous y prenez. C'est de là que vient le contrecoup, ce préjugé. Nous devons donc répliquer. Leur dire, bon, vous ne comprenez pas comment les AA pensent et ça vous inquiète. Très bien, allons faire un tour sous le capot. Désassemblons la machine. Ce que vous n'aimez pas, ce sont ces boîtes noires étanches. Bien, ouvrons-les.

– **Kazuo Ishiguro**, Klara et le Soleil (2021).

Remerciements

Merci David, merci Aurélien. Merci d'avoir proposé un sujet de thèse aussi passionnant et novateur. Travailler sur ce sujet était un plaisir, travailler avec vous l'était davantage. Si j'allais au labo avec le sourire, c'était en grande partie grâce à vous. Merci pour votre bonne humeur et votre bienveillance. Le chemin était loin d'être tracé, mais malgré les difficultés, vous avez toujours réussi à proposer des études intellectuellement stimulantes et pertinentes en vue de la réalisation de l'objectif. Vous avez été mes guides durant ce périple à travers la jungle, et je suis plus que ravie de pouvoir continuer l'expédition à vos côtés.

Mes remerciements vont ensuite vers les membres du jury : Tom Beucler, Fabrice Gamboa, Pierre Gentine, Corentin Lapeyre, Valérie Monbet. Pour être honnête, je ne pense pas que l'on puisse faire mieux comme jury. Merci à Valérie et Pierre d'avoir accepté la lourde tâche de rapporter le manuscrit. Merci à Tom pour la relecture attentive. Vos commentaires et remarques seront d'une aide précieuse pour la poursuite du travail. J'aimerais également remercier tous les membres de jury pour l'échange très enrichissant que nous avons eu à l'issue de la soutenance. J'ai pris en note toutes vos remarques, commentaires et suggestions, et ils nourriront ma réflexion dans le futur. Merci également à Ludovic Auger, membre du comité de thèse, pour les remarques et conseils sur les modèles non-hydrostatiques.

La réussite de cette thèse doit également être attribuée à mon entourage au labo. Merci beaucoup à mon chef Olivier, pour ta bonne humeur et pour tes réponses très complètes à mes innombrables questions sur les paramétrisations. Surtout, merci beaucoup d'avoir convaincu avec David et Aurélien la direction sur la pertinence du travail sur la développement des paramétrisations à l'aide d'algorithmes d'Intelligence Artificielle. Merci également à Saïd, mon co-bureau, de m'avoir supportée pendant ces trois années, et merci pour nos discussions toujours enirchissantes. Merci à Nono et à Salomé pour les discussions thé où nous avons partagé nos galères et succès de thésards ingénieurs météo. Merci à Jeanne pour les petites discussions et à Julien pour ses remarques spirituels hors paire. Merci également à toutes les personnes participant à Bibliostats du jeudi matin (Thomas, Clément, Antoine et les autres). Que ça fait du bien de retourner faire des maths et de papoter de statistiques au tableau !

Cet entourage quotidien ne se limite pas uniquement aux collègues du labo. J'aimerais également remercier les poufiasses qui se reconnaîtront : Manon, Yehudi, Alexis et Quentin. On s'est croisé au lycée il y a 10 ans pile (sauf la Noufmoufle, mais ça fait 8 ans quand même). Mais votre bonne humeur n'a pas pris une ride (contrairement à vous). Merci tout particulièrement à Quentin pour toutes les heures passées au téléphone à parler de sciences, de galères et de petits succès.

Merci également à coach Sébastien, mon entraîneur. La pratique de l'athlétisme

était très importante pour mon équilibre personnel, même si parfois, je suis allée trop vite sur les fractions ou au contraire j'étais lessivée avant même d'avoir commencé l'échauffement. Maintenant, je n'ai plus l'excuse de la thèse.

Enfin (et surtout), j'aimerais remercier Mickaël. Merci de me supporter et d'être toujours là pour moi. Tu m'as relevée un nombre incalculable de fois durant ces trois années, pour cela je te suis éternellement reconnaissante. Tu me disais que la section 'remerciements', ça peut tenir en 5 mots : j'ai essayé de ne pas être trop bavarde.

Résumé

Dans un modèle de climat, les paramétrisations physiques ont pour rôle de représenter l'effet moyen des processus sous-maille (e.g., convection profonde, turbulence) à la résolution du modèle. Elles sont numériquement efficaces, mais aussi la principale source d'incertitude dans les modèles de climat. Depuis quelques années, l'utilisation des techniques d'Intelligence Artificielle (IA), en particulier celle des réseaux de neurones (*Neural Networks*, NN), a permis de développer des paramétrisations d'un genre nouveau. Contrairement aux paramétrisations physiques traditionnelles, développées en utilisant des connaissances théoriques et des observations, les paramétrisations apprises sont ajustées à un jeu de données issu d'une simulation haute-résolution directement, dans le but d'apprendre une représentation plus précise des différents processus sous-maille. Les premiers résultats sont prometteurs, mais il reste des problèmes majeurs à résoudre avant leur utilisation dans un modèle de climat, en remplacement des paramétrisations traditionnelles. Le but de cette thèse est l'étude de quelques-uns de ces problèmes à l'aide de modèles jouets et à travers l'apprentissage d'une partie de la paramétrisation physique implémentée dans un modèle de climat.

Tout d'abord, l'utilisation d'une physique apprise par des NN peut rendre instable le modèle de climat dans lequel elle est branchée. La première partie de cette thèse est dédiée à l'étude de cette instabilité avec l'aide d'un modèle jouet, spécifiquement développé pour étudier ce problème. Un deuxième problème concerne la performance des physiques apprises, une fois celles-ci branchées dans un modèle de climat, notamment en terme de climatologie du modèle (e.g., biais de long-terme). Nous avons développé une méthode permettant d'ajuster les statistiques de long-terme d'une physique apprise. La méthode a été illustrée à l'aide de modèles jouets. Enfin, nous nous sommes intéressés à l'apprentissage d'un schéma de convection profonde à l'aide de NN. Le schéma appris a été implémenté avec succès dans le modèle de climat global développé au CNRM, ARPEGE-Climat, en remplacement du schéma physique. Nous avons réalisé plusieurs années de simulation sans constater de divergence du modèle. Les principales caractéristiques du climat moyen sont bien représentées. La physique apprise est interprétée en utilisant des outils d'analyse de sensibilité. Cette étude pose quelques bases en vue du développement de paramétrisations apprises à partir de données haute-résolution, dont l'utilisation pourrait améliorer la précision des modèles numériques du climat.

Mots-clés : climat - convection profonde - machine learning - modèle jouet - modélisation de l'atmosphère - paramétrisation physique - réseaux de neurones

Abstract

In a numerical climate model, the role of physical parameterizations is to represent the average effect of subgrid-scale processes (e.g., deep convection, turbulence) at model resolution. Current parameterizations are numerically efficient, but also the main source of uncertainty in climate models. In recent years, the use of Artificial Intelligence (AI) techniques, in particular neural networks (NN), has allowed the development of a new kind of parameterization. As opposed to traditional physical parametrizations, developed using theoretical knowledge and observations, learned parametrizations are directly fitted to a dataset derived from a high-resolution simulation, in order to learn a more accurate representation of the different subgrid processes. The first results are promising, but major problems remain to be solved before their use in a climate model to replace traditional parameterizations. The aim of this thesis is to investigate some of these problems using toy models and through the implementation of a machine-learning based, deep convection parameterization in a climate model.

First of all, the use of NN-learned physics can make the climate model in which it is plugged unstable. The first part of this thesis is dedicated to the study of this instability in a toy model specifically developed to investigate this problem. A second problem is the performance of the learned parameterization, once they are plugged into a climate model, especially in terms of model climatology (e.g., long-term bias). We have developed a method to adjust the long-term statistics of a learned parameterization. The method was illustrated using toy models. Finally, we focused on the learning of a deep convection scheme using NN. The learned scheme was successfully implemented in the global climate model developed at CNRM, ARPEGE-Climat, as a replacement of the physical scheme. We have performed several years of simulation without noticing any drift of the model. The main characteristics of the mean climate are well represented. The learned parameterization is interpreted using sensitivity analysis tools. This study lays some groundwork for the development of learned parameterizations from high-resolution data, the use of which could improve the accuracy of numerical climate models.

Keywords : climate - deep convection - machine learning - neural networks - numerical models of the atmosphere - physical parameterization - toy model

Table des matières

Introduction générale

Qu'est-ce que le climat ? Pourquoi le modéliser ?

Au cours des dernières années, les injonctions à limiter notre utilisation d'énergies fossiles se multiplient, dans le but de limiter le réchauffement climatique. Bien que l'on soit *seulement* à +1,2°C par rapport à l'objectif de +2°C d'ici la fin du siècle fixé par les Accord de Paris, force est de constater que les effets du réchauffement climatique se font déjà ressentir. Au printemps 2022, une vague de chaleur de forte ampleur et exceptionnellement longue frappe l'Inde et la Pakistan. La rareté d'un tel événement peut être quantifiée par sa durée de retour, la durée moyenne entre deux épisodes similaires successifs. La durée de retour de cet épisode est estimée à 100 ans dans les conditions actuelles. L'événement a été estimé 30 fois plus probable qu'il ne l'aurait été en l'absence de changement climatique[1].

Le réchauffement climatique est souvent assimilé à ses conséquences. Mais comment définir ce qu'est le climat ? Dans son glossaire, le Groupe d'experts Intergouvernemental sur l'Évolution du Climat (GIEC) définit le *climat* comme suit.

> **Climate**
> "Climate in a narrow sense is usually defined as the average weather, or more rigorously, as the statistical description in terms of the mean and variability of relevant quantities over a period of time ranging from months to thousands or millions of years. The classical period for averaging these variables is 30 years, as defined by the World Meteorological Organization. The relevant quantities are most often surface variables such as temperature, precipitation and wind. Climate in a wider sense is the state, including a statistical description, of the climate system."[2]

Le climat peut donc être défini comme la *moyenne des conditions météorologiques*. Contrairement à la météorologie, la climatologie a pour objectif de prévoir les *caractéristiques statistiques d'une distribution*, plutôt qu'une *chronologie précise des événements*.

[1] `https://www.worldweatherattribution.org/climate-change-made-devastating-early-heat-in-india-and-pakistan-30-times-more-likely/`

[2] `https://www.ipcc.ch/site/assets/uploads/2018/11/sr15_glossary.pdf`, consulté le 07/06/2022.

Le changement climatique correspond à un décalage des caractéristiques statistiques de cette distribution au cours du temps. La valeur moyenne de la température prise sur le globe tend à se décaler vers des valeurs plus élevées. Désormais, nous savons que le réchauffement observé est, en grande partie, attribué à l'activité anthropique, notamment aux émissions de gaz à effet de serre (IPCC, 2022). L'histoire du climat montre que nous avons déjà connu des périodes chaudes ou de réchauffement. Mais le caractère exceptionnel et potentiellement dangereux du changement climatique vient en partie des échelles de temps restreintes en jeu : lorsque la Terre a connu des périodes plus chaudes par le passé, les réchauffements comparables se sont étalés sur des dizaines de milliers d'année. Le réchauffement anthropique que nous connaissons actuellement s'effectue plus rapidement, si bien que les êtres vivants n'ont plus le temps de s'adapter au changement.

Pour mettre en place des stratégies d'adaptation, il est important de prévoir l'ampleur du réchauffement climatique pour les décennies à venir. Pour prévoir le climat du futur, on développe des modèles numériques du climat, dont nous allons détailler le fonctionnement par la suite.

La modélisation numérique du climat

Les modèles numériques du climat visent à décrire l'évolution du système climatique, composé de l'atmosphère, des océans, de la cryosphère et de la biosphère. La composante atmosphérique des modèles de climat, appelée modèle de circulation générale (*general circulation model*, GCM) vise à décrire l'évolution thermodynamique du fluide atmosphérique. Celle-ci est décrite à l'aide d'une série d'équations, les *équations primitives* (Annexe A). Elles correspondent aux équations de Navier-Stokes, à l'équation de continuité, à l'équation hydrostatique et à l'équation de la conservation de l'énergie. Les équations de Navier-Stokes permettent de calculer la dérivée temporelle de la variable d'état $\mathbf{x}$ du fluide atmosphérique, également appelée *tendance*, qui peut être intégrée à partir d'une condition initiale. La série temporelle obtenue par intégrations successives des tendances de $\mathbf{x}$, à partir d'une condition initiale donnée, correspond à une *simulation climatique*.

À défaut de connaître une solution analytique de l'équation de Navier-Stokes, le modèle atmosphérique résout le système d'équations de manière approximative. Les équations sont discrétisées et résolues en un ensemble de *points de grille*. De manière standard, la résolution horizontale des modèles globaux de climat actuellement disponibles est de l'ordre de 100 kilomètres. Les équations primitives sont également discrétisées selon la verticale au-dessus de chaque point de grille, sur un ensemble de *niveaux verticaux*, allant typiquement de la surface jusqu'au sommet de la mésosphère. La grille horizontale et les niveaux verticaux forment la *maille* du modèle (Figure 1). Mais la discrétisation des équations n'est pas seulement spatiale, elle est aussi temporelle. Cette dernière discrétisation est définie par le pas de temps

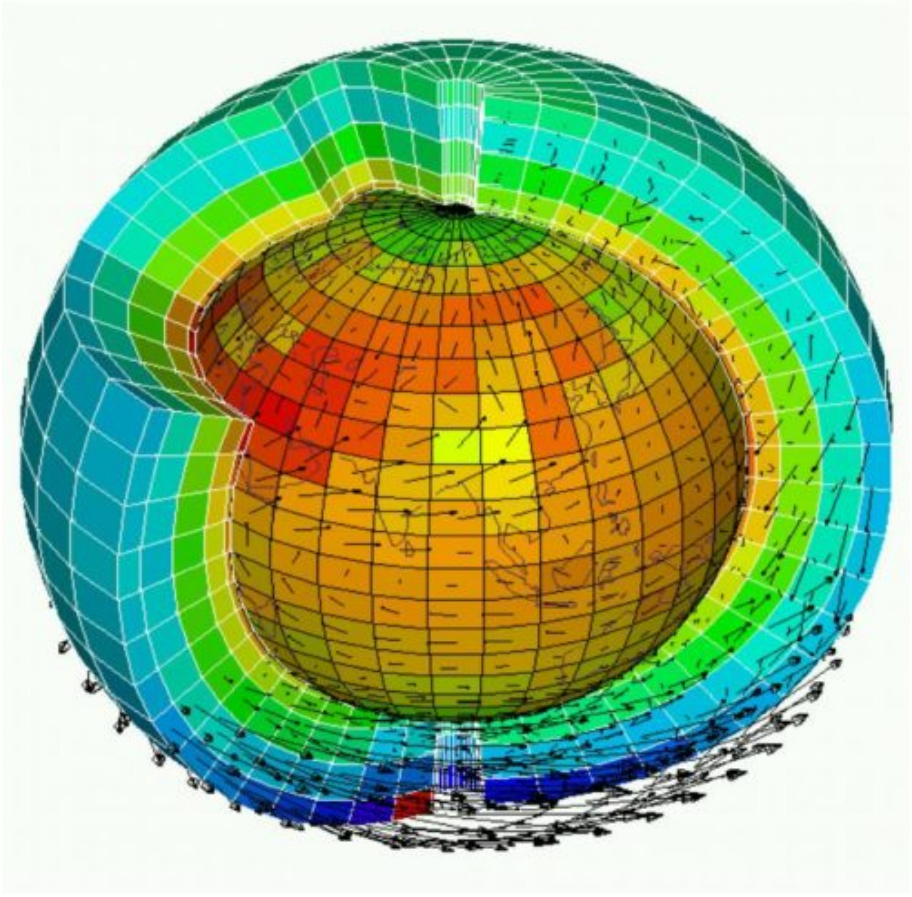


Figure 1: *Maillage tridimensionnel d'un modèle de climat. Un carré correspond à une maille. 5 niveaux verticaux sont représentés. Une colonne atmosphérique correspond à l'ensemble des niveaux verticaux au-dessus d'un point de grille. Figure emrpuntée au Laboratoire de Météorologie Dynamique.*

du modèle, qui est de l'ordre de 15 minutes pour un modèle de climat global.

La discrétisation des équations primitives permet de décrire, de manière approximative, les processus atmosphériques se produisant à des échelles spatiales et temporelles supérieures à celle de la résolution (horizontale, verticale et temporelle) et constituent la *dynamique* $\mathcal{D}$ du modèle numérique du climat. Mais il existe des processus se produisant à des échelles inférieures à la résolution du modèle, et leur effet à grande échelle est souvent non négligeable (Figure 2). Les nuages d'orages par exemple ont une taille caractéristique de l'ordre de la dizaine de kilomètres, bien en-deçà de la résolution horizontale d'un GCM. Les cellules orageuses sont souvent responsables de fortes précipitations locales qui ne sont pas décrites par les équations primitives discrétisées.

La représentation des processus sous-maille

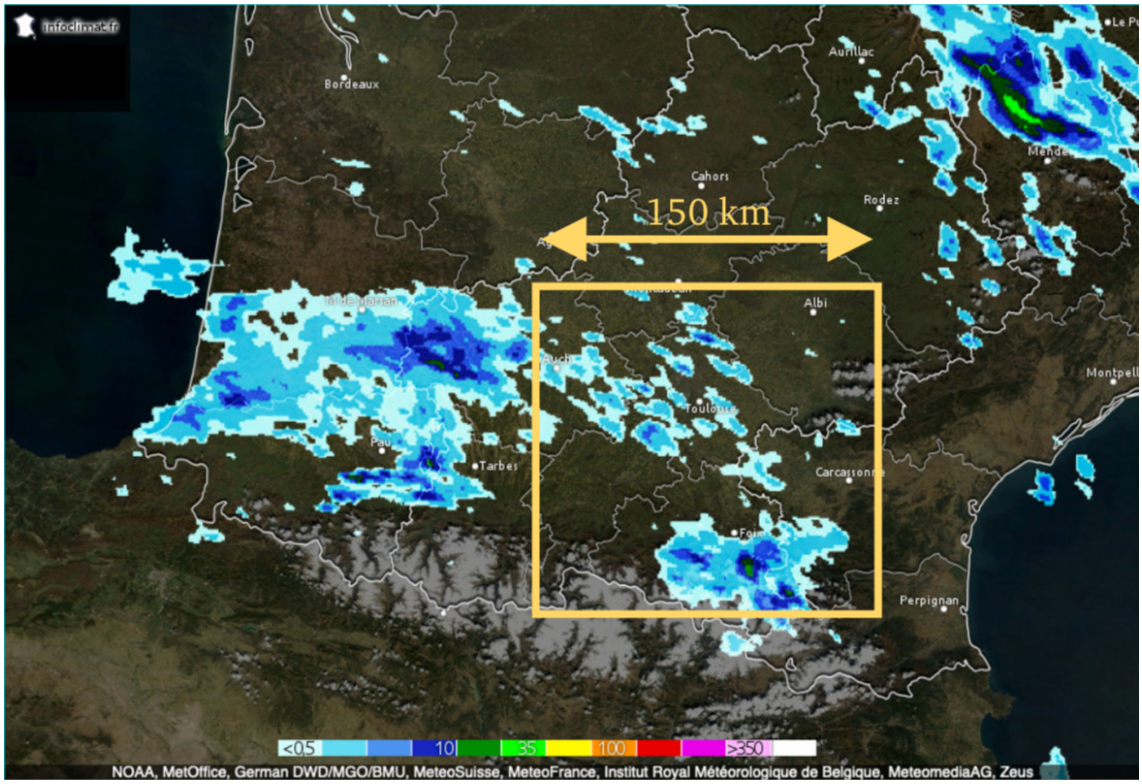


Figure 2: *Exemple d'image radar, indiquant les précipitations instantanées. Les taches en nuances de bleu représentent l'intensité des précipitations, engendrées par des nuages convectifs. Un côté du carré orange correspond à environ 150 km, soit la résolution horizontale des GCM. Les cellules convectives se situant à l'intérieur du carré orange ne sont pas décrites explicitement par la dynamique $\mathcal{D}$ d'un GCM actuel standard. Figure issue du radar d'Infoclimat.*

Les processus atmosphériques se produisant à des échelles plus fines que la résolution du modèle sont appelés *processus sous-maille*. Pour tenir compte de leur effet moyen à l'échelle de la maille, on développe des *paramétrisations physiques* (Palmer, 2001). Dans le modèle ARPEGE-Climat (CNRM-CM6, Voldoire et al., 2019; Roehrig et al., 2020), les processus paramétrisés sont représentés sur la Figure 3 et listés ci-dessous.

- **La convection** profonde et peu profonde correspond à la formation des nuages d'instabilité, appelés *cumulus*. Ce processus est responsable de la redistribution de la température et de l'humidité selon la verticale, à travers, en particulier, la formation de nuages d'orage.

- **Les processus turbulents** se produisent essentiellement dans la couche limite atmosphérique (les plus basses couches de l'atmosphère), liés à l'interaction entre la surface et l'écoulement du fluide atmosphérique. Cette paramétrisation permet notamment de représenter les inversions au sommet de la couche limite atmosphérique.

- **Les échanges radiatifs** correspondent à la distribution des rayonnements infrarouge (dans toutes les directions) et solaire incident au sommet de l'atmosphère (vers les basses-couches). Cette distribution est notamment influencée par la présence de nuages.

- **La microphysique nuageuse** décrit les différents changements d'état au sein des nuages. En fonction des conditions de température, de pression et d'humidité, l'eau nuageuse est sous forme liquide, solide ou vapeur. Ces états se divisent également en plusieurs sous-catégories (e.g., cristaux de neige ou grêle en phase solide), décrites par le schéma microphysique, et ont de fortes conséquences sur les propriétés radiatives des nuages.

- **Les ondes** correspondent au forçage du fluide atmosphérique, notamment par le relief (*ondes orographiques*), mais également par des différences de densité entre plusieurs masses d'air (*ondes non-orographiques*).

- Enfin, **la chimie atmosphérique** fait également partie des processus paramétrisés. Un certain nombre de réactions chimiques se produisent dans l'atmosphère, comme par exemple la décomposition de l'ozone (O_3) en dioxygène (O_2).

Les processus cités ci-dessus font chacun l'objet d'un schéma de paramétrisation, le plus souvent développé de manière individuelle. L'ensemble des processus individuellement paramétrisés est appelée la *physique* $\mathcal{P}$ du modèle atmosphérique. La somme de la dynamique et de la physique permet de calculer la tendance $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ à intégrer pour obtenir l'état de l'atmosphère $\mathbf{x}$ au pas de temps suivant :

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathcal{D}(\mathbf{x}) + \mathcal{P}(\mathbf{x}). \tag{1}$$

Actuellement, la plupart des paramétrisations physiques sont développées de manière heuristique, en utilisant des connaissances théoriques, des données issues de simulations haute-résolution et des observations (e.g., Jakob, 2010). Elles sont numériquement efficaces, mais cela a un prix : elles sont insuffisantes pour décrire de manière satisfaisante certains mécanismes clés, bien identifiés dans les jeux d'observations dont on dispose. Les paramétrisations sont ainsi la principale source d'incertitude dans la composante atmosphérique des modèles de climat pour un scénario d'émission donné (e.g., Medeiros et al., 2008; Medeiros and Stevens, 2011; Stevens and Bony, 2013).

Pour améliorer la représentation des processus sous-maille dans un modèle de climat, on peut utiliser des techniques d'Intelligence Artificielle (IA). Au cours de la dernière décennie, l'utilisation de ces techniques a permis des développements majeurs dans de nombreux domaines de recherche scientifique, allant du domaine de l'imagerie médicale (e.g., Rakhlin et al., 2018) à la classification des galaxies (e.g., Kim and Brunner, 2016). Dans les sciences météo-climatiques, leur utilisation est également de plus en plus répandue (e.g., Ham et al., 2019; Mounier et al., 2022).

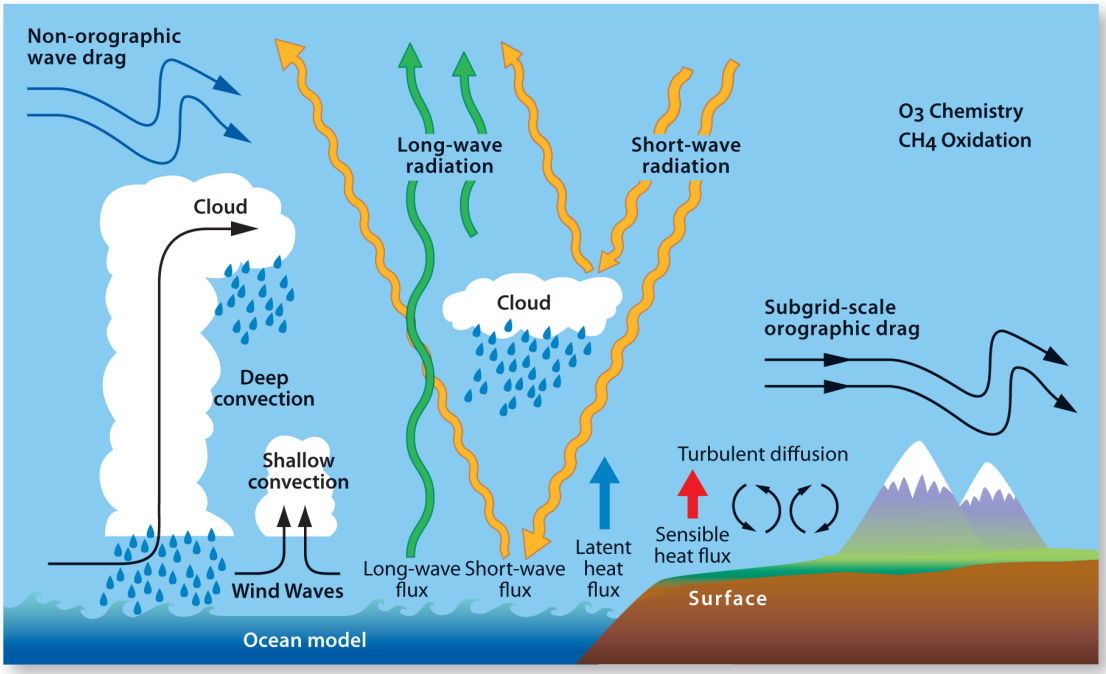


Figure 3: *Différents processus paramétrisés dans un modèle de climat. Chaque processus fait l'objet d'une paramétrisation individuelle $\mathcal{P}_i$ dans certains modèles de climat, en particulier ARPEGE-Climat et IFS. Source: ECMWF.*

Des travaux des dernières années et en cours se focalisent sur l'utilisation de données issues de simulations réalisées avec des modèles de climat à très haute-résolution (de l'ordre du kilomètre de résolution horizontale) pour développer des paramétrisations. À une telle résolution horizontale, la discrétisation des équations primitives décrit de manière explicite certains processus ayant besoin d'être paramétrisés dans les GCM. La représentation de ces processus est ainsi plus précise que grâce aux paramétrisations physiques. La réalisation des simulations haute-résolution est toutefois numériquement très coûteuse, limitant leur utilisation à des simulations de courte durée (typiquement quelques mois).

Problématique. L'objectif de cette thèse est d'étudier quelques problèmes qui empêchent aujourd'hui l'utilisation des paramétrisations IA dans les GCM.

Organisation du manuscrit.

Chapitre 1 : Méthodes d'apprentissage statistique. Pour développer des paramétrisations IA, on utilise principalement des réseaux de neurones et des forêts aléatoires. Dans la suite du manuscrit, on s'intéresse particulièrement aux réseaux de neurones, dont le principe est détaillé dans ce chapitre.

Chapitre 2 : État de l'art. Depuis quelques années, on s'intéresse au développement de paramétrisations en utilisant des algorithmes de Machine Learning. Ce chapitre fait le point sur l'avancée de la recherche à ce sujet.

Chapitre 3 : Stabilité. Les paramétrisations actuelles utilisant des réseaux de neurones sont, pour la plupart, instables. Pour étudier l'origine de cette instabilité numérique, nous avons développé et étudié un modèle jouet. Ce chapitre est la retranscription de Balogh et al. (2021).

Chapitre 4 : Calibration. Un modèle de climat performant doit avoir un faible biais à long terme. Le biais des paramétrisations physiques est réglé *online*, une fois que le schéma a été implémenté dans le modèle. Cela n'est pas possible dans le cas des paramétrisations avec des réseaux de neurones actuels. Nous proposons une méthode permettant une calibration *online* des paramétrisations avec réseaux de neurones. Ce chapitre est la retranscription de Balogh et al. (2022).

Chapitre 5 : Apprentissage du schéma de convection profonde d'un modèle de climat. L'apprentissage du schéma de convection profonde implémenté dans un modèle de climat nous met dans un cadre plus réaliste que les modèles jouets, notamment en terme de dimensions en jeu. L'apprentissage est réalisé avec des réseaux de neurones. Le schéma appris est utilisé dans un modèle de climat en remplacement d'une partie de la physique pour réaliser des simulations. Les résultats de cette simulation sont comparés à la climatologie et interprétés à l'aide d'outils d'analyse de sensibilité.

Chapitre 1

Méthodes d'apprentissage statistique

Le contenu de ce chapitre est largement inspiré les excellents supports de cours de Ph. Besse et de B. Laurent, dispensés à l'INSA Toulouse dans le cadre du M2 Sciences des données (version 2018), ainsi que du livre Deep Learning in Python (Chollet, 2017).

1.1 Introduction historique

Le développement de la science des données (*data sciences*) est fortement lié à l'augmentation de la capacité de calcul des ordinateurs et supercalculateurs. Les travaux fondateurs de la science des données ont été initiés dès 1930. La capacité de calcul est alors de l'ordre du hecta octet (hOctet). Cela est suffisant pour effectuer les premières expériences de statistique inférentielle : les expériences statistiques sont alors *planifiées* sur des échantillons de taille $n \sim 30$ pour $p \sim 10$ paramètres.

A la fin de la Seconde Guerre Mondiale, le concept de *neurone formel* voit le jour grâce au travail conjoint du neuroscientifique Warren McCulloch et du logicien Walter Pitts (McCulloch and Pitts, 1943). Le fonctionnement de ces neurones formels est inspiré de celui des neurones biologiques, laissant passer ou non l'influx nerveux. À l'instar de ceux rencontrés en biologie, les neurones de McCulloch et Pitts ont vocation à être organisés en *réseaux* (Figure 1.1). Hebb (1949) décrit la manière dont le poids associé à chaque neurone peut être optimisé pour obtenir une certaine réponse désirée. L'auteur réalise ainsi le premier apprentissage pour un système de neurones. Mais la formulation du premier *réseau* de neurones est habituellement attribuée à Rosenblatt (1958). Inspiré par le neurone de McCulloch & Pitts, son objectif est de modéliser la reconnaissance de formes par la rétine. Il appelle son réseau de neurones *perceptron* : ce nom est toujours utilisé pour désigner un réseau de neurones composée de couches Dense (voir Section 1.2.2).

Au même moment, une branche de l'informatique dédiée aux *machines intelligentes* se développe. A. Turing s'intéresse à la construction de telles machines dès 1950. Dans Turing (1950), il décrit le *test de Turing* : son principe est qu'une machine est suffisamment intelligente si une personne n'arrive pas à déterminer s'il parle avec une machine ou une autre personne.

La convergence entre l'informatique et la modélisation mathématique des réseaux de neurones a lieu dans les années 1950. En 1956, lors d'un workshop organisé à Dartmouth réunissant les meilleurs experts en théorie des automates, de l'intelligence et des réseaux de neurones (biologiques), naît le terme d' *intelligence artificielle* (IA) (Kline, 2011). Cette période est caractérisée par un atmosphère d'enthousiasme et de positivité, permettant un développement fulgurant des travaux sur l'IA. Newell et al. (1957) décrit le *general problem solver*, un solveur capable de résoudre divers problèmes, comme par exemple une optimisation de trajectoires. Le premier langage informatique orienté IA, LISP, voit le jour en 1960 (McCarthy, 1960). Mais le manque de puissance de calcul met rapidement un coup d'arrêt aux nombreuses avancées. La capacité de calcul progresse, pour atteindre, dans les années 1970, le

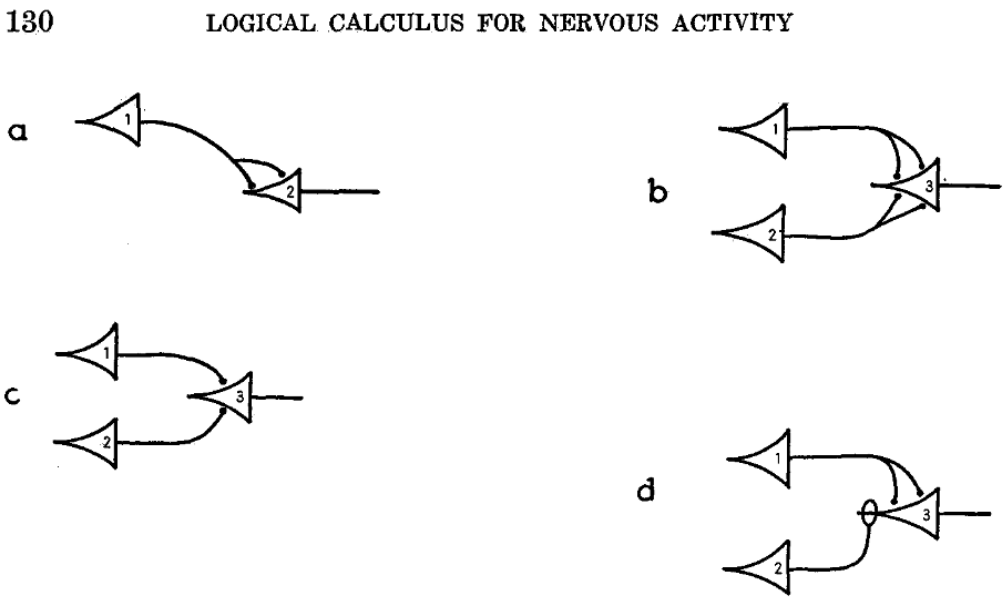


Figure 1.1: *Quelques exemples d'organisations entre neurones formels. Les neurones sont connectées entre elles et laissent passer ou barrent la route à l'information véhiculé dans le réseau. Figure issue de McCulloch and Pitts (1943).*

kilo octet (ko). Cela est suffisant pour poser les bases de l'analyse des données, en France (Caillez and Pagès, 1976) et aux États-Unis (Turkey, 1977), mais les travaux sur l'IA sont mis de côté.

Il faut attendre les années 1980 pour reprendre les travaux sur les réseaux de neurones (*neural networks*, NN). D'une part, la puissance de calcul disponible permet de traiter des jeux de données de quelques Méga octets (Mo). D'autre part, au début des années 1980, des avancées théoriques majeures permettent l'implémentation informatique des premiers réseaux de neurones. La plus importante d'entre elles est sans doute le développement d'un algorithme d'estimation des gradients par *rétropropagation de l'erreur* (Rumelhart et al., 1986), permettant d'effectuer un apprentissage efficace avec les NN. Pour garantir une bonne flexibilité, les NN sont associés à plusieurs centaines de milliers de poids et de biais, qui doivent être ajustés en fonction de l'échantillon d'apprentissage. De par les dimensions en jeu, disposer d'un algorithme efficace pour les optimiser est essentiel pour pouvoir réaliser un apprentissage efficace. Les travaux sur la rétropropagation de l'erreur ont permis d'initier le premier changement de paradigme profitant au développement des NN. Tout d'abord, Hornik et al. (1989) démontre, dans son théorème d'approximation universelle, que les NN peuvent (théoriquement) approximer n'importe quelle fonction continue sur un support compact, à la précision souhaitée. De plus, en 1990, la capacité des calculateurs permet de traiter des jeux de données de quelques Giga octets (Go). Les données n'ont plus besoin d'être planifiées : elles sont d'abord

acquises et ensuite directement utilisées pour ajuster des modèles statistiques (e.g., Fayyad et al., 1996). L'utilisation massive de données permet le développement de l'apprentissage machine (*machine learning*, ML Vapnik, 1998), qui met les techniques d'IA, et notamment les NN, sur le devant de la scène. Breiman (2001) propose l'algorithme des *forêts aléatoires* (*Random Forests*, RF) en 2001 et devient rapidement populaire grâce à sa simplicité d'utilisation. L'IA commence également à rentrer dans la culture populaire, notamment après la défaite de G. Kasparov face à Deep Blue (IBM) aux échecs.

La capacité de calcul poursuit sa croissance, permettant un deuxième changement de paradigme au début des années 2000 et marquant ainsi le début du *big data*. D'une part, les supercalculateurs permettent l'utilisation de jeux de données de plusieurs tera octets (To), ce qui permet notamment le développement de la bioinformatique. Au cœur de la bioinformatique, les problèmes de séquençage d'ADN : le nombre p de variables dépasse largement la taille n de l'échantillon d'apprentissage. D'autre part, l'utilisation de processeurs graphiques (*Graphical Processing Units*, GPU) révolutionne l'apprentissage des NN, permettant d'ajuster un modèle de complexité élevée en un temps record.

Le troisième changement de paradigme a lieu dans les années 2010. Outre la puissance de calcul disponible, permettant alors l'utilisation de jeux de données de quelques péta octets (Po), les progrès récents en miniaturisation des capteurs facilitent l'acquisition des données. Grâce à la couverture de différents réseaux de téléphonie mobile (2G, 3G, 4G, 5G), les données acquises par les capteurs peuvent être immédiatement transmises et utilisées. Le nombre d'objets connectés explose. Les montres enregistrent en temps réel la position ainsi que la fréquence cardiaque. La domotique est en plein essor : les prises connectées transmettent leurs recommandations à leurs propriétaires afin de consommer moins d'énergie.

Les algorithmes de ML sont utilisés dans le cadre des sciences météo-climatiques depuis les années 1990 (e.g., Krasnopolsky et al., 2013; Ham et al., 2019). Les modèles de prévision numérique du temps et les modèles de climat génèrent une grande quantité de données, rendant l'utilisation des techniques de ML pertinente.

1.2 Les réseaux de neurones

1.2.1 Le neurone formel

Le *neurone formel* (ou neurone) est la brique élémentaire des réseaux de neurones. Il s'agit de la représentation mathématique des neurones biologiques (cellules), dont le rôle est de synthétiser l'influx nerveux qui leur parvient, et de transmettre cette information à d'autres neurones proches. Ils permettent ainsi de véhiculer l'information issue des nerfs sensitifs ou des nerfs moteurs.

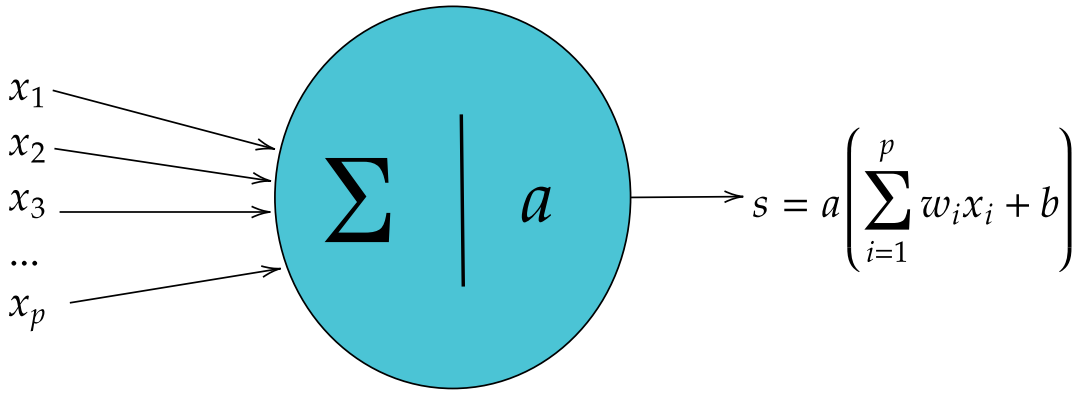


Figure 1.2: *Un neurone formel. Les entrées $\boldsymbol{x} = (x_1, x_2, ..., x_p)$ sont d'abord multipliés par des poids $\boldsymbol{w}$, puis sommées en ajoutant le biais b. En appliquant une fonction d'activation a à la quantité précédemment obtenue, on obtient l'état interne s du neurone.*

Un neurone est caractérisé par son état interne, s (Figure 1.2). Si on note $\mathbf{x} = (x_1, x_2, ..., x_p)$ les variables en entrée du neurone, l'état interne de celui-ci est donné par

$$s = a\left(\sum_{i=1}^{p} w_i x_i + b\right), \tag{1.1}$$

avec $\mathbf{w} = (w_1, w_2, ..., w_p)$ des poids associés aux différentes variables en entrée $\mathbf{x}$, b le biais et a une fonction d'activation. Le plus souvent, les neurones biologiques laissent passer l'influx nerveux si la tension engendrée par celui-ci dépasse une certaine valeur seuil. Le rôle des fonctions d'activation est de reproduire ce comportement, avec une contrainte supplémentaire : elles doivent être différentiables pour être facilement utilisées par les algorithmes de rétropropagation. Les fonctions principalement utilisées pour activer les neurones sont :

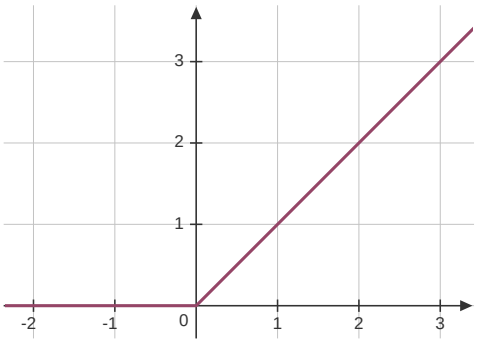
- **linéaire** : $a(x) = x$;
- **Rectified Linear Unit (ReLU)** : $a(x) = \max(0, x)$;
- **tangente hyperbolique** : $a(x) = \tanh(x)$;
- **sigmoïde** : $a(x) = \frac{1}{1+e^x}$ (classification binaire).

Les fonctions ReLU, tangente hyperbolique et sigmoïde sont représentées au voisinage de 0 sur la Figure 1.3.

Dans ce paragraphe, le neurone formel a été présenté sous sa forme la plus élémentaire. Il existe toutefois des neurones plus sophistiqués, pour lesquels le calcul de l'état interne à partir des entrées se fait à l'aide d'autres fonctions que celle affine. Quelques exemples de tels neurones seront présentés à la fin du chapitre (Sec. 1.2.4).

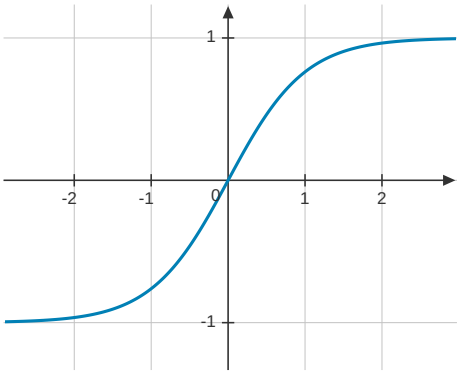
unité linéaire rectifiée (ReLU)

$$a(x) = \max(0, x)$$



tangente hyperbolique

$$a(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



sigmoide

$$a(x) = \frac{1}{1 + e^{-x}}$$

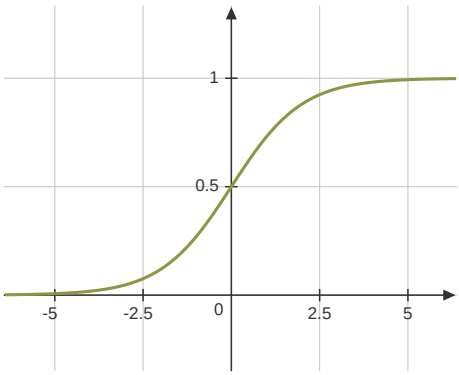


Figure 1.3: *Illustration de quelques fonctions d'activation fréquemment utilisées. La fonction ReLU est celle qui est réplique le mieux le fonctionnement du neurone biologique, avec un seuil d'activation à $x = 0$. Les fonctions tanh et sigmoide correspondent, quant à elles, à une réponse bi-modale à l'exception d'une zone de transition au voisinage de 0.*

1.2.2 Organisation des neurones en réseaux

Tout comme pour les neurones biologiques, les neurones formels s'organisent en *couches*, composées de plusieurs neurones. La couche la plus simple (et qui sera principalement utilisée dans la suite du manuscrit) est composée de plusieurs neurones formels, comme celui décrit dans la section précédente. Les couches, quant à elles, sont également reliées entre elles pour former un *réseau* de neurones. Le NN le plus simple est composé de L couches *cachées*, interposées entre la couche d'entrée et celle de sortie. Sur chaque couche, les neurones sont directement reliés à l'ensemble des neurones des couches précédente et suivante (Figure 1.4). Cette architecture simple est également appelée *perceptron multicouche* (*Multilayer Perceptron*, MLP). Un MLP est associé à une fonction de transfert $\widehat{f}$, donnant une estimation de la variable de sortie, $\widehat{\mathbf{y}}$, à partir des prédicteurs $\mathbf{x}$:

$$\widehat{\mathbf{y}} = \widehat{f}(\mathbf{x}; \boldsymbol{\theta}), \tag{1.2}$$

avec $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{b})$ où $\mathbf{w}$ et $\mathbf{b}$ sont la matrice des poids et le biais associés à l'ensemble des couches. Le choix des fonctions d'activation est fait pour chaque couche cachée. La fonction sigmoïde est souvent utilisée pour activer les neurones sur la dernière couche d'un modèle de classification binaire. Dans le cas d'un NN de régression, la dernière couche est activée par une fonction linéaire. Les fonctions tangente hyperbolique et ReLU sont, quant à elles, utilisées pour activer des couches cachées. Le choix de la fonction d'activation a des conséquences importantes sur le comportement d'un modèle de NN.

Une fois que l'architecture du NN a été défini, l'étape suivante consiste à ajuster les poids et les biais, en fonction des données disponibles. Cette étape s'appelle l'apprentissage.

1.2.3 L'apprentissage

Théorie

Soit $\{\mathbf{x}_i, \mathbf{y}_i\}_{1 \leq i \leq N}$ une collection de données, appelée *échantillon d'apprentissage*. L'objectif de l'apprentissage est d'optimiser en fonction de cet échantillon les paramètres $\theta = (\mathbf{w}, \mathbf{b})$ du NN, avec $\mathbf{w}$ les poids et les biais $\mathbf{b}$. Les poids et biais optimaux correspondent au minimum d'une fonction perte (*loss*), $\mathcal{L}$. Pour une régression, la fonction perte est généralement quadratique (*Mean Squared Error*, MSE) :

$$\mathcal{L}(\theta) = \|\mathbf{y} - \widehat{f}(\mathbf{x}; \theta)\|^2. \tag{1.3}$$

Pour estimer la valeur optimale de θ, on on calcule les dérivées de la fonction perte par rapport à $\mathbf{w}$ et à $\mathbf{b}$, en partant de la dernière couche à la première, pour remonter jusqu'à la couche d'entrée. Cette méthode permet de relier les poids et biais aux variables en entrée et en sortie, que nous connaissons à travers l'échantillon d'apprentissage. Les dérivées ainsi calculées forment les *équations de rétropropagation* (Encadré 1.1).

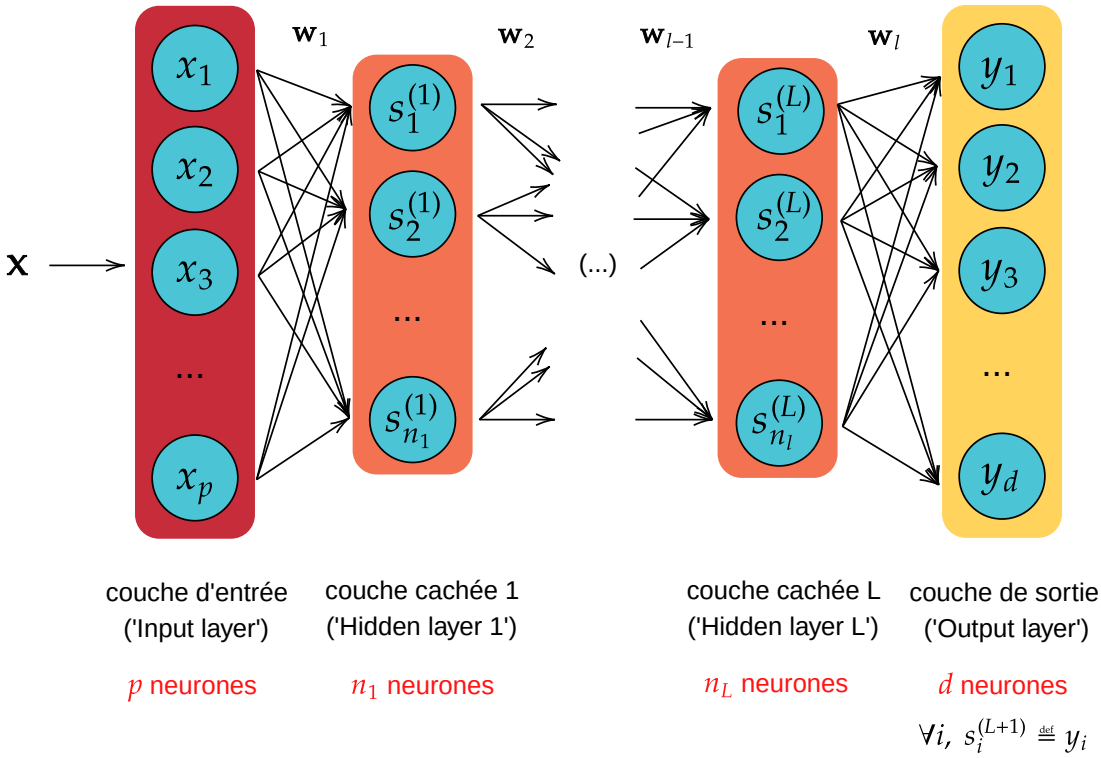


Figure 1.4: *Un réseau de neurones 'Dense', également appelé perceptron multicouche La couche d'entrée (rouge) est reliée à la couche de sortie (jaune) grâce à L couches cachées (orange). Celles-ci sont composées de n_l neurones, pour chaque couche $1 \leq l \leq L$. Pour les couches cachées, chaque neurone est directement relié à l'ensemble des neurones sur les couches précédente et suivante.*

Encadré 1.1 : équations de rétropropagation

Soit L le nombre de couches cachées dans un MLP, chacune activée par la fonction a. L'état d'un neurone sur la couche $1 \leq l \leq L$ est notée $\mathbf{s}^{(l)} = a(\mathbf{h}^{(l)})$. Soit $\mathcal{L}$ une fonction perte. En introduisant la notation $\delta^{L+1} = \nabla_{\mathbf{s}^{(L)}}\mathcal{L}$, les gradients de $\mathcal{L}$ par rapport aux poids $\mathbf{w}^{(l)}$ et biais $\mathbf{b}^{(l)}$ de la couche l sont données par :

$$\forall \ \ 1 \leq l \leq L, \quad \delta^{(l)} = (\mathbf{w}^{(l)})^T \delta^{l+1} a'(\mathbf{h}^{(l)}) \tag{1.4}$$

$$\nabla_{\mathbf{w}^{(l)}}\mathcal{L} = \delta^{(l)}(\mathbf{s}^{(l+1)})^T \tag{1.5}$$

$$\nabla_{\mathbf{b}^{(l)}}\mathcal{L} = \delta^{(l)}. \tag{1.6}$$

La démonstration de ces équations est disponible à la fin du manuscrit (Annexe C). Ces équations sont utilisées pour mettre à jours les paramètres θ du NN. Le processus de mise à jour des paramètres s'écrit de manière simple pour l'algorithme

élémentaire *stochastic gradient descent* (SGD), à l'itération $i+1$:

$$\theta_{i+1} = \theta_i + \varepsilon \frac{\partial \mathcal{L}}{\partial \theta}(\theta_i). \tag{1.7}$$

Le terme $\dfrac{\partial \mathcal{L}}{\partial \theta}$ est obtenu en appliquant les équations de rétropropagation (Éq. 1.5 et 1.6). La quantité ε est appelée taux d'apprentissage (*learning rate*, LR). Ce paramètre permet de définir la *longueur du pas* que l'on effectue à chaque itération en direction du minimum. L'ensemble de ces paramètres et équations permet de décrire, de manière schématique, l'algorithme d'optimisation de θ (Algorithme 1.1).

Algorithme 1.1 : estimation des paramètres de θ

Initialisation. Choisir les différents paramètres (taux d'apprentissage ε, taille du batch n_b, le nombre d'epochs n_e). Initialiser aléatoirement les paramètres $\theta_0 = (\mathbf{w}_0, \mathbf{b}_0)$ du NN. Les mises à jour du paramètre θ seront comptées avec la variable i, initialisée à 0.

for $e = 1, ..., n_e$, do;
  for $l = 1, ..., N/n_b$, do;
    1. Tirer aléatoirement un *batch* B_l de taille n_b dans l'échantillon (bootstrap) d'apprentissage : $(\mathbf{x}_j, \mathbf{y}_j)_{j \in B_l}$;
    2. Calculer les gradients par rapport à $\theta_i = (\mathbf{w}_i, \mathbf{b}_i)$ en utilisant les équations de rétropropagation. Le gradient $\mathbf{grad}_{\theta_i}$ sera la moyenne des gradients sur le batch

$$\mathbf{grad}_{\theta_i} = \frac{1}{n_b} \sum_{j \in B_l} \frac{\partial \mathcal{L}(f(\mathbf{x}_j; \theta_i), \mathbf{y}_j)}{\partial \theta_i}.$$

    3. Mettre à jour les paramètres θ_i du NN :

$$\theta_{i+1} = \theta_i - \varepsilon \mathbf{grad}_{\theta_i}.$$

Pratique

Dans la pratique, la majorité des packages python dédiés aux NN permet d'ajuster θ de manière automatique. Mais il est possible de jouer (manuellement) sur la valeur d'une multitude de paramètres, appelés *hyperparamètres*, disponibles pour améliorer la qualité du modèle de NN obtenu. La valeur de ces hyperparamètres est définie et fixée avant l'apprentissage. Nous allons passer en revue le fonctionnement des principaux d'entre eux.

- **Choix des échantillons train/test/validation.** Le partitionnement de l'échantillon d'apprentissage doit conduire à la création de trois sous-échantillons

aussi indépendants entre eux que possible. Le plus souvent, un partitionnement aléatoire de l'échantillon initial est suffisant. Dans le cas de séries temporelles continues, pour garantir une meilleure indépendance, il est conseillé de garder des séries temporelles continues pour le test et la validation plutôt que de partitionner l'échantillon d'apprentissage de manière aléatoire.

- **Fonction perte** $\mathcal{L}$. Par défaut, le choix s'oriente vers la MSE. Un deuxième choix fréquent est l'erreur absolue moyenne (*mean absolute error*, MAE), moyenne des erreurs absolues, qui est moins sensible aux valeurs aberrantes (*outliers*). Les fonctions perte peuvent être pénalisées, pour contraindre l'optimisation de θ.

- **Batch size**. Les poids et biais sont optimisés de manière individuelle pour chaque individu $\mathbf{x}_i$ dans l'échantillon d'apprentissage. Mais cela aboutit le plus souvent à un NN $\widehat{f}$ très bruité, et les algorithmes risquent de converger vers des poids et biais optimaux localement. Pour se rapprocher du minimum global, les poids et biais optimaux correspondent à leur valeur moyenne sur un ensemble de n_b individus tirés aléatoirement dans l'échantillon d'apprentissage. Ce sous-échantillon est appelé *batch*. La valeur de n_b (*batch size*) a une très grande influence sur la performance du modèle final, et doit donc être soigneusement optimisée. Un batch trop petit risque de faire converger le modèle vers un minimum local, alors qu'un batch trop grand ne permet pas de représenter suffisamment de variabilité.

- **Nombre d'epochs**. Pendant l'optimisation de θ, l'échantillon d'apprentissage est parcouru plusieurs fois. Un passage à travers l'échantillon d'apprentissage est appelé un *epoch*. À chaque changement d'epoch, les individus dans l'échantillon d'apprentissage sont permutés. Par conséquent, les individus affectés dans chaque batch sont différents d'un epoch à un autre. Le nombre d'epochs correspond à la durée de l'apprentissage. Plus leur nombre augmente, plus on risque le surapprentissage. Pour éviter ce défaut, on dispose de plusieurs options. Premièrement, il est possible de diminuer le taux d'apprentissage de manière graduelle au fur et à mesure que l'on avance dans les epochs (voir plus bas). Deuxièmement, certaines options disponibles permettent d'arrêter l'apprentissage dès lors que l'erreur sur un échatillon de validation indépendant de celui d'apprentissage remonte (*Early Stopping*). Il est également possible d'inclure des couches *Dropout* dans le NN, qui sélectionnent un sous-ensemble aléatoire de connections de neurones pour ajuster les paramètres du NN.

- **Taux d'apprentissage** (*Learning Rate*, LR). Le taux d'apprentissage est un paramètre essentiel pour la détermination de θ optimal. Plus le taux d'apprentissage est important, plus on fait de grands pas vers le minimum, au risque de le dépasser. Un taux d'apprentissage trop petit est souvent très coûteux numériquement (et pas nécessaire). Une méthode d'optimisation du LR consiste à essayer plusieurs valeurs (grandes et petites) pour avoir une idée

approximative de la valeur optimale. Ensuite, on réalise un dernier apprentissage avec un taux d'apprentissage qui décroît en fonction des epochs (*Learning Rate Scheduler*), en partant d'une valeur initiale τ_0. La décroissance est souvent exponentielle. Plus on se rapproche du minimum, plus on fait de petits pas vers celui-ci.

- **Choix de l'algorithme d'optimisation.** Dans la plupart des packages python dédiés au machine learning, il existe une multitude d'optimiseurs disponibles. Le choix par défaut est souvent l'algorithme `adam` (Kingma and Ba, 2017). Il est possible d'essayer plusieurs optimiseurs, notamment quand les résultats sont très mauvais, ou lorsque l'on rencontre des problèmes de stabilité ou de points fixes. Certains optimiseurs ont également leur propres paramètres à optimiser.

En résumé, l'apprentissage correspond à l'optimisation des poids $\mathbf{w}$ et des biais $\mathbf{b}$ *via* un algorithme de rétropropagation. Pour cela, on choisit une fonction perte $\mathcal{L}$, quantifiant l'erreur de prévision, et qui sera minimisée pendant l'apprentissage. Dans la pratique, il existe plusieurs paramètres que l'on peut ajuster (e.g. batch size, nombre d'epochs) pour obtenir un modèle performant.

1.2.4 Des réseaux plus sophistiqués

Le NN décrit dans la section précédente correspond au modèle de base, adapté à des problèmes où les variables en entrée et en sortie sont des vecteurs. Mais il existe des NN plus élaborés, implémentant des couches plus complexes ou organisés selon des architectures différentes. Dans la suite, nous allons donner quelques exemples de telles couches ou architectures. Celles-ci n'ayant pas été utilisées durant la thèse, la théorie associée ne sera que brièvement abordée.

Les réseaux de neurones convolutifs

Précédemment, nous avons détaillé le fonctionnement des couches de type Dense, les briques élémentaires constitutives des MLP. Les MLP ne sont pas adaptés pour le traitement des variables sous forme de matrices, comme par exemple les images, les vidéos ou encore les cartes. Mettre les matrices sous la forme d'un vecteur pour être compatibles avec les MLP est souvent une tache fastidieuse qui risque de supprimer l'information spatiale associée aux données, pourtant essentielle à leur compréhension.

Les réseaux de neurones convolutifs (*Convolutionnal Neural Networks*, CNN) sont une classe de NN spécifiquement pensés pour les problèmes de traitement d'images. Ils ont été introduits par Le Cun et al. (1990), dans le but d'automatiser la lecture de codes postaux manuscrits. Contrairement aux MLP, où chaque neurone est connecté aux autres, on utilise dans les CNN des couches convolutives. Celles-ci ont pour rôle d'extraire de l'information *localisée* sur une partie de l'image, à l'aide

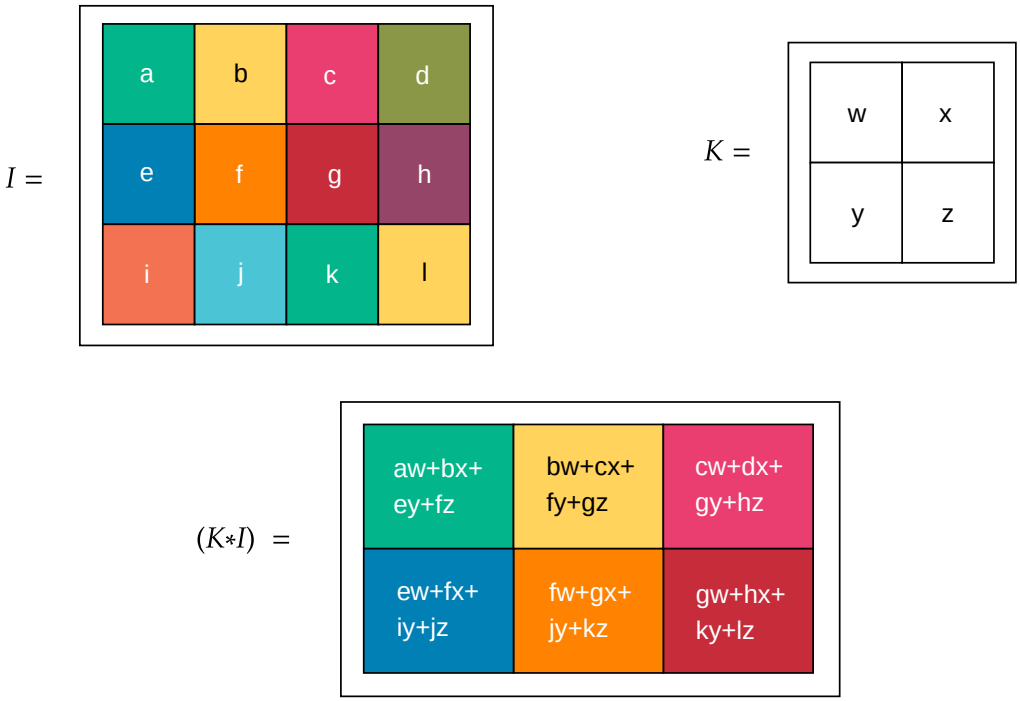


Figure 1.5: *Convolution d'une image I et d'un kernel K. Le résultat de l'opération de convolution $(K * I)$ (Éq. 1.8) est représentée dans la matrice en bas de la figure.*

d'un *kernel* K. Le kernel est convolué avec une image I dans les couches convolutives. Pour une convolution 2D, l'opération de convolution est notée $*$ et s'exprime, pour chaque élément (i, j) de la matrice obtenue après convolution :

$$(K * I)(i, j) = \sum_{m,n} K(m, n)I(m + i, j + i). \tag{1.8}$$

L'opération de convolution d'une image avec un kernel permet de calculer des moyennes locales et ainsi de compresser l'information que l'image contient (Figure 1.5), en respectant la structure spatiale des données.

Pour que l'extraction d'information soit efficace, un deuxième type de couche est traditionnellement introduit dans les CNN : le *pooling*. Une fois la convolution avec le kernel effectuée, ces couches viennent réduire davantage la dimension du problème, en effectuant des opérations locales. Il existe plusieurs types d'opérations de pooling. Parmi elles, le *max pooling* permet de ne garder que la valeur maximale autour d'un point donnée, et l'opération *mean pooling* la valeur moyenne localement calculée.

Les CNN sont typiquement composés d'une succession de couches convolutives et de pooling. Une fois l'information spatiale suffisamment extraite (compressée) d'une image, les CNN se terminent en général par un bloc *fully connected*, correspondant à une succession de couches Dense.

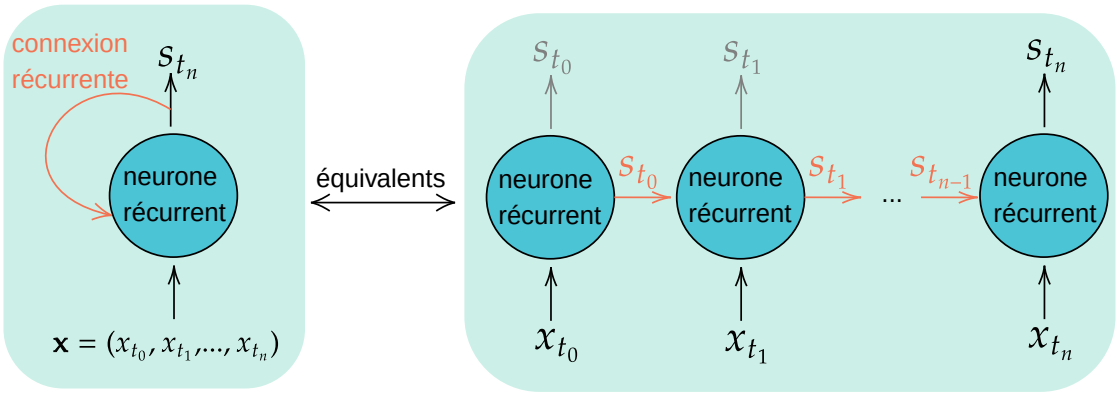


Figure 1.6: *Vision simplifiée d'un neurone récurrent. L'état s du neurone est gardé en mémoire via une connexion récurrente.*

L'utilisation de CNN est de plus en plus répandue dans le cadre de sciences météo-climatiques (e.g., Ham et al., 2019; Mounier et al., 2022). Les CNN ne sont toutefois couramment utilisés pour l'apprentissage de paramétrisations. Ceux-ci ne sont pas directement efficaces, car les données en entrée et en sortie des paramétrisations sont des vecteurs.

Les réseaux de neurones récurrents

Dans les exemples précédents, nous avons vu que les sorties du NN n'étaient fonction que d'une seule variable en entrée (e.g., vecteur, carte), et l'ordre des échantillons que le NN voit pendant l'apprentissage n'a aucun impact sur le modèle final. Mais il existe des données où l'ordre des variables est tout aussi important que la variable elle-même. L'ordre des mots est essentiel dans une phrase : par exemple, la conjugaison d'un verbe est conditionnée par le sujet. Si on cherche à créer un NN pour générer des phrases, les architectures précédemment décrites ne permettent pas de *garder en mémoire* les mots au début de la phrase, pourtant nécessaires pour formuler une suite cohérente de mots.

L'objectif des réseaux de neurones récurrents (*recurrent neural networks*, RNN) est de permettre le traitement des données sous forme de séquences. Dans les couches récurrentes, en plus des variables en entrée décrites pour les architectures précédentes, l'état s du neurone est également renvoyé en entrée de celui-ci, permettant de garder une trace de l'état du neurone issu des variables en entrée passées en revue précédemment (Figure 1.6).

Les RNN ne sont pas (encore) utilisés pour créer des paramétrisations IA. Néanmoins, leur utilisation pour l'apprentissage des paramétrisation pourrait s'avérer judicieuse. Bien que l'on ne dispose pas directement de séries temporelles pour l'apprentissage, l'ordre des éléments dans les profils verticaux a toute son importance, comme l'ordre

des mots dans une phrase. La valeur d'une variable thermodynamique dans le profil vertical à un certain niveau donné est conditionnée par celles des couches inférieures. Utiliser des RNN dont la mémoire porterait sur l'axe verticale pourrait donc s'avérer pertinente pour l'amélioration des paramétrisations dans un modèle de climat.

1.3 Conclusion – discussion

Les NN sont une méthode d'apprentissage statistique de grande flexibilité, pouvant approcher n'importe quelle fonction continue sur un support compact (théorème de l'approximation universelle, Hornik et al., 1989). Leur utilisation a permis de révolutionner de nombreux domaines de recherche scientifique (e.g., traitement d'image). Les paramètres θ d'un NN sont optimisés en appliquant les équations de rétropopagation. Leurs valeurs optimales sont calculées en fonction d'un échantillon d'apprentissage, en utilisant une fonction perte, $\mathcal{L}$. Les NN les plus simples sont composées de couches Dense, où chaque neurone est connecté à ceux sur les couches précédente et suivante (MLP). Cette architecture est particulièrement adaptée lorsque les variables en entrée et en sortie sont des vecteurs. Il existe d'autres types de NN. Les réseaux convolutifs constituent le choix de base pour le traitement d'images. Les réseaux récurrents sont, quant à eux, performants pour le traitement d'informations sous forme de séries temporelles.

Il existe d'autres algorithmes de ML que les NN. En particulier, les Random Forests (RF), dont le principe est présenté en annexe (Annexe B). Cette méthode n'a pas été utilisée pour réaliser les travaux présentés dans la suite du manuscrit, pour plusieurs raisons (principalement pratiques). Tout d'abord, l'implémentation python des RF ne permet pas de réaliser un ajustement aussi souple que celle des NN. Les fonctions perte disponibles sont très limitées : il y a seulement deux choix proposés dans le package scikit-learn. De plus, pour que l'algorithme soit performant, il est souvent nécessaire d'agréger un grand nombre d'arbres. Cela signifie que le modèle RF associé occupera en mémoire un volume considérable. Étant donné que les paramétrisations physiques apprises ont vocation à être implémentées et appelées pour chaque colonne dans le code (Fortran) des modèles de climat, il est préférable de disposer de modèles moins coûteux numériquement. Pour toutes ces raisons, nous utiliserons uniquement des NN pour mener les travaux présentés dans le présent manuscrit.

Chapitre 2

État de l'art

En introduction, nous avons vu qu'il était possible d'utiliser des techniques d'IA (en particulier des réseaux de neurones, présentés dans le Chapitre 1), pour améliorer la physique d'un modèle de climat. Dans ce chapitre, nous allons voir comment l'utilisation des techniques d'IA a évolué pour l'amélioration de la physique, en fonction de l'augmentation de la puissance de calcul et des progrès théoriques en sciences des données.

2.1 Les premiers pas : accélération d'un schéma existant

Une étude préliminaire de l'utilisation de techniques d'IA pour améliorer la physique de modèles de climat consiste à émuler des schémas physiques déjà existants. Le schéma du transfert radiatif est le candidat idéal pour cela : il est à la fois numériquement coûteux et peu dépendant des autres parties du modèle (i.e., des autres paramétrisations et de la dynamique). Dès lors, l'objectif des premières émulations du schéma de transfert radiatif est de proposer une version performante mais numériquement moins coûteuse de celui-ci, qui pourrait être utilisée en remplacement du schéma existant. NeuroFlux est le premier émulateur NN pour le schéma de transfert radiatif pour les grandes longueurs d'ondes (*longwave*, LW). Il permet d'accélérer la paramétrisation originale d'un facteur 22 (Chevallier et al., 1998).

Ce travail pionnier permet de lancer les travaux pour créer des émulateurs NN de différents schémas de transfert radiatif. Krasnopolsky et al. (2005) utilise un réseau de neurones simple pour émuler le schéma de transfert radiatif LW du modèle atmosphérique *Community Atmosphere Model, v.2* (CAM-2), développé au NCAR. L'émulateur NN final est composé d'une couche cachée avec 90 neurones. Après validation sur un échantillon indépendant de celui d'apprentissage, les bons résultats obtenus permettent d'utiliser le schéma dans CAM-2, en parallèle avec le schéma de transfert radiatif LW existant. Sur une simulation de validation d'une durée de 10 ans, l'erreur de l'émulateur n'excède pas la variabilité interne du modèle. En outre, l'utilisation de l'émulateur NN permet une accélération vertigineuse d'un facteur 80 par rapport au schéma physique. L'émulateur est étendu au schéma de transfert radiatif pour les courtes longueurs d'ondes (*shortwave*, SW) du modèle CAM-2 (Krasnopolsky et al., 2008). Lorsque l'émulateur remplace les schémas d'origine dans CAM-2, l'accélération constatée est d'un facteur 16 pour le schéma LW, 20 pour le schéma SW par rapport au schéma initial. Krasnopolsky et al. (2010) poursuit le développement de l'émulateur du schéma de transfert radiatif dans un cadre plus réaliste, en utilisant un modèle de climat complet, en constatant des accélérations des schémas LW et SW comparables à celles précédemment mentionnées.

Suite à l'ensemble de ces résultats très prometteurs sur les schémas de transfert radiatif, Krasnopolsky et al. (2013) s'attaque à la création d'un émulateur NN pour le schéma convectif de CAM-2. La difficulté supplémentaire est que les auteurs ne

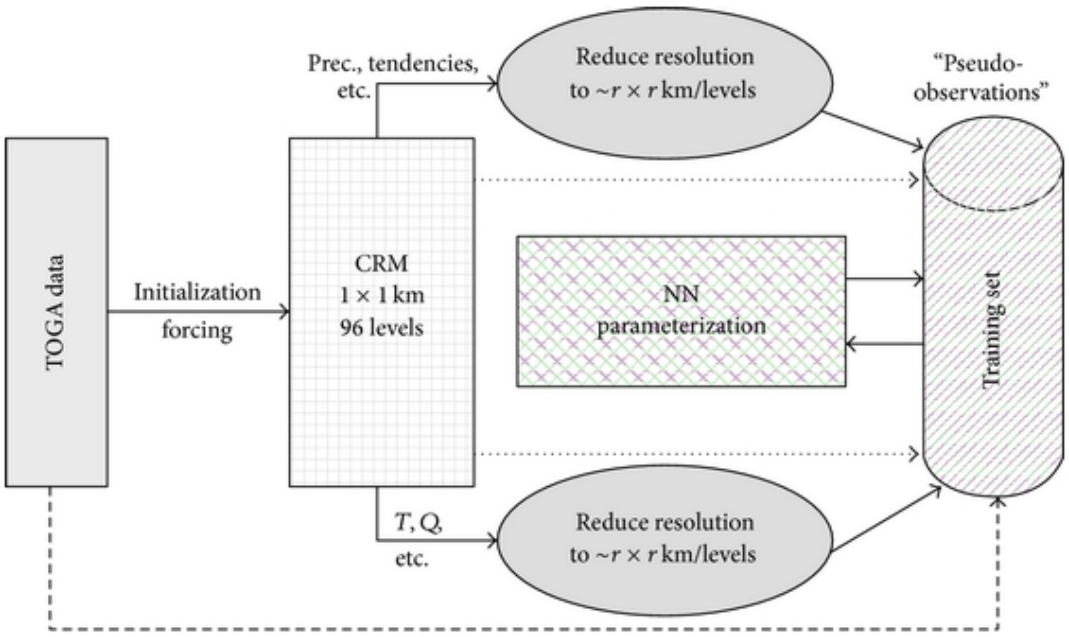


Figure 2.1: *Processus d'agrégation des données haute-résolution, issues d'un modèle haute-résolution (cloud resolving model, CRM). Les données observées (TOGA data, rectangle gris à gauche) sont utilisées pour initialiser le CRM (rectangle quadrillé), de résolution horizontale 1 km sur 96 niveaux verticaux. Les variables en entrée du NN (T, Q, etc., ovale gris en bas de la figure) ainsi que les variables en sortie (précipitations, tendances, etc., ovale gris en haut de la figure) sont agrégées à une résolution horizontale r, tout en conservant les 96 niveaux verticaux. Les données ainsi obtenues sont appelées pseudo-observations, et elles constituent la base d'apprentissage (cylindre Training set à droite de la figure) pour la paramétrisation NN (NN parameterization, rectangle central à motifs). Figure extrait de Krasnopolsky et al. (2013).*

souhaitent pas émuler le schéma convectif existant, mais ils veulent l'améliorer. Pour cela, ils utilisent un échantillon d'apprentissage construit à partir d'une simulation non-hydrostatique à aire limitée. Les données ainsi obtenues sont utilisées pour générer des *pseudo-observations* : à partir d'une résolution horizontale de 1 km, les données sont agrégées à la résolution de CAM-2 (Figure 2.1). L'avantage de la simulation haute-résolution (HR) est que la convection profonde y est explicitement résolue, donnant ainsi une représentation plus précise des processus convectifs que celle obtenue en appliquant une paramétrisation. Les données agrégées ou pseudo-observations servent donc de base pour l'apprentissage du schéma convectif grâce à des NN.

En guise de validation, la paramétrisation NN de la convection profonde est implémentée dans CAM-2 et utilisée en parallèle avec le schéma physique initial. Pour évaluer la généralisation du modèle de NN en dehors de son échantillon d'apprentissage,

la validation s'effectue sur une bande tropicale dont l'étendue latitudinale excède celle du domaine utilisé pour générer les pseudo-observations. Bien que de très bons résultats aient été obtenus, étendre la paramétrisation NN au-delà de la bande tropicale (i.e., aux latitudes moyennes et aux régions polaires) nécessite la réalisation de simulations avec un modèle haute-résolution résolvant explicitement la convection profonde (*Cloud Resolving Model*, CRM), à l'échelle globale. Cependant, les ressources de calcul dont disposent les auteurs ne permet pas la réalisation d'une telle simulation. Les études portant sur le développement de paramétrisations NN sont donc momentanément arrêtées, en attendant que la puissance de calcul disponible permette de réaliser des simulations non-hydrostatiques globales.

Par la suite, d'autres émulateurs pour des schémas physiques ont été développés, dans le but d'accélérer un schéma. On peut citer les travaux de Chantry et al. (2021), qui a pour objectif d'accélérer le schéma des ondes non-orographiques. Pour cela, ils utilisent des NN. Bien que sur GPU, on constate une accélération d'un facteur 10 par rapport au schéma d'origine, l'émulateur NN n'est pas plus rapide que le schéma d'origine sur CPU[1].

2.2 Amélioration des schémas

2.2.1 Apprentissage depuis des données agrégées

Cinq ans après les travaux de Krasnopolsky et al. sur l'apprentissage de la convection profonde, nous sommes désormais en mesure de réaliser des simulation globales non-hydrostatiques, grâce aux *global cloud resolving models* (GCRM). Ces simulations sont réalisées de manière ponctuelle, car la puissance de calcul actuellement disponible ne permet pas de réaliser des simulations plus longues que quelques années (AR5, 2014). Parmi d'autres, l'expérience DYAMOND (DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains, Stevens et al., 2019) utilise une version non-hydrostatique de modèles de prévision numérique du temps, notamment ARPEGE, développé au CNRM. Dans le cadre de cette expérience, une simulation globale d'une durée de 50 jours, à la résolution horizontale 2.5 km, a été réalisée. À cette échelle, certains processus ayant besoin d'être paramétrisés dans les GCM sont directement décrits par les équations primitives discrétisées. La convection profonde, par exemple, fait partie des processus explicitement résolus par les GCRM et dont l'utilisation d'une représentation plus précise pourrait améliorer les GCM de manière considérable (e.g., meilleure représentation du cycle diurne de la convection et de l'oscillation de Madden-Jullian).

[1]Cet exemple soulève la question essentielle de l'architecture des futurs supercalculateurs. Actuellement, ceux-ci sont majoritairement composées de partitions CPU. Mais beaucoup de centre de recherches et services météorologiques semblent se tourner vers une architecture mixte, implémentant CPUs et GPUs.

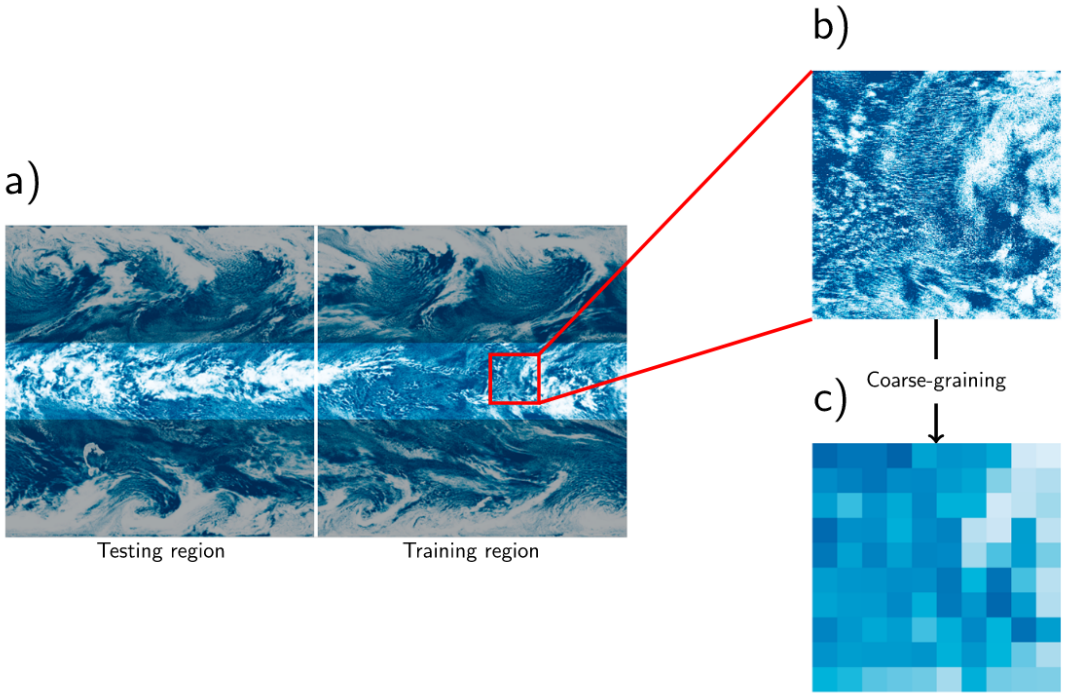


Figure 2.2: *Processus de coarse-graining dans le modèle SAM. (a) Une simulation haute-résolution est réalisée. Les sorties issues de cette simulation (b) sont agrégées à une résolution plus grossière (c). Figure empruntée à Brenowitz and Bretherton (2018).*

Pour qu'une paramétrisation des processus explicitement résolus dans les GCRM puisse être apprise par des algorithmes de ML, il faut, dans un premier temps, dégrader les données HR à la résolution du GCM pour lequel le schéma est développé. Cette étape est appelée *coarse-graining* (CG, Figure 2.2), et associe, à chaque variable $\mathbf{x}^{HR}$ issue des données HR, une valeur agrégée $\mathbf{x}$ sur la grille du GCM. Une fois l'étape de CG réalisée, il existe différentes possibilités pour l'apprentissage, en terme de variables en entrée et de sortie. Nous allons passer en revue le développement de certaines paramétrisations NN pour GCMs.

Gentine et al. (2018) et Rasp et al. (2018)

Gentine et al. (2018) et Rasp et al. (2018) visent à améliorer la représentation de la convection profonde dans un GCM. Ils utilisent les mêmes données HR, générées grâce au modèle *Super Parameterized Community Atmosphere Model 3* (SPCAM3) en configuration *aquaplanet* (la surface du globe est dépourvue de surfaces continentales). Ce modèle est composé d'un GCM, où la convection profonde est explicitement résolue dans chacune des 8192 colonnes du modèle. La simulation dure 2 ans : la première année est utilisée pour l'apprentissage, la seconde pour la validation, pour un total de 280 millions d'individus dans les deux échantillons. Pour des raisons pratiques, les auteurs apprennent la totalité de la partie sous-maille du modèle, plutôt que la convection individuellement. Dans le jeu de données d'apprentissage, on

dispose uniquement des tendances *totales* (i.e., correspondant à la physique et à la dynamique) et des tendances issues de la dynamique. Finalement, l'apprentissage de la totalité des paramétrisations revient ici à améliorer la partie des paramétrisations correspondant à la convection profonde uniquement.

Dans Gentine et al. (2018), le NN retenu, appelé CBRAIN, est composé de 8 couches cachées avec 512 neurones sur chaque couche. Le NN de Rasp et al. (2018) a 9 couches cachées et 256 neurones par couche. L'utilisation d'un modèle de NN plus profond leur a permis d'obtenir des meilleures performances et un modèle numériquement plus robuste. Dans les deux cas, l'optimisation s'effectue en utilisant comme fonction perte la MSE. Les variables en entrée et en sortie sont centrées-réduites pour accélérer l'apprentissage. Bien que l'accélération du schéma ne soit pas l'objectif recherché, les auteurs notent une accélération d'un facteur 10 (Gentine et al., 2018) et 20 (Rasp et al., 2018) par rapport à la physique de SPCAM3.

Afin d'évaluer les performances de CBRAIN, dans un premier temps, les auteurs comparent les moyennes zonales des variables en sortie calculées sur la 2ème année de la simulation SPCAM3 (Figure 2.3). On parle alors de validation *offline* : le NN est testé sur un jeu de données, et pas dans le modèle directement (validation *online*). La physique CBRAIN parvient à représenter la plupart des structures de grande échelle.

Contrairement à la validation *offline* décrite dans Gentine et al. (2018), Rasp et al. (2018) réalise une validation *online*, en effectuant une simulation où la physique NN a été branchée dans le GCM hôte de SPCAM3. La durée de la simulation de validation est de 5 ans, après un *spin-up* d'un an. La période de spin-up correspond à la mise à l'équilibre du modèle dynamique ; habituellement, cette période transitoire est ôtée des simulations. Le modèle où la physique apprise a remplacé les paramétrisations physiques (et le modèle explicite de la convection) est stable. Les résultats observés sont également très prometteurs. Par exemple, l'utilisation de la physique NN permet de résoudre le problème de la double ITCZ, un biais persistant bien connu dans les GCM (e.g., Oueslati and Bellon, 2015; Tian and Dong, 2020). Les distributions spatiales de différentes variables (convective heating rate, convective moistening rate, longwave heating rate, shortwave heating rate) sont également réalistes.

Récemment, Wang et al. (2022) ont réalisé une simulation d'une durée de 5 ans avec une paramétrisation apprise à partir d'une configuration similaire à celle décrite dans Rasp et al. (2018). Les auteurs apprennent la représentation des processus de fine échelle à partir de données SPCAM agrégées. Pour parvenir à une paramétrisation stable une fois implémentée *online*, ils ont, dans un premier temps, réalisé l'apprentissage des processus sous-maille avec 37 NNs, dont la configuration était légèrement différente à chaque fois. Ils ont ensuite éliminé de manière empirique les paramétrisations apprises conduisant à des simulations instables.

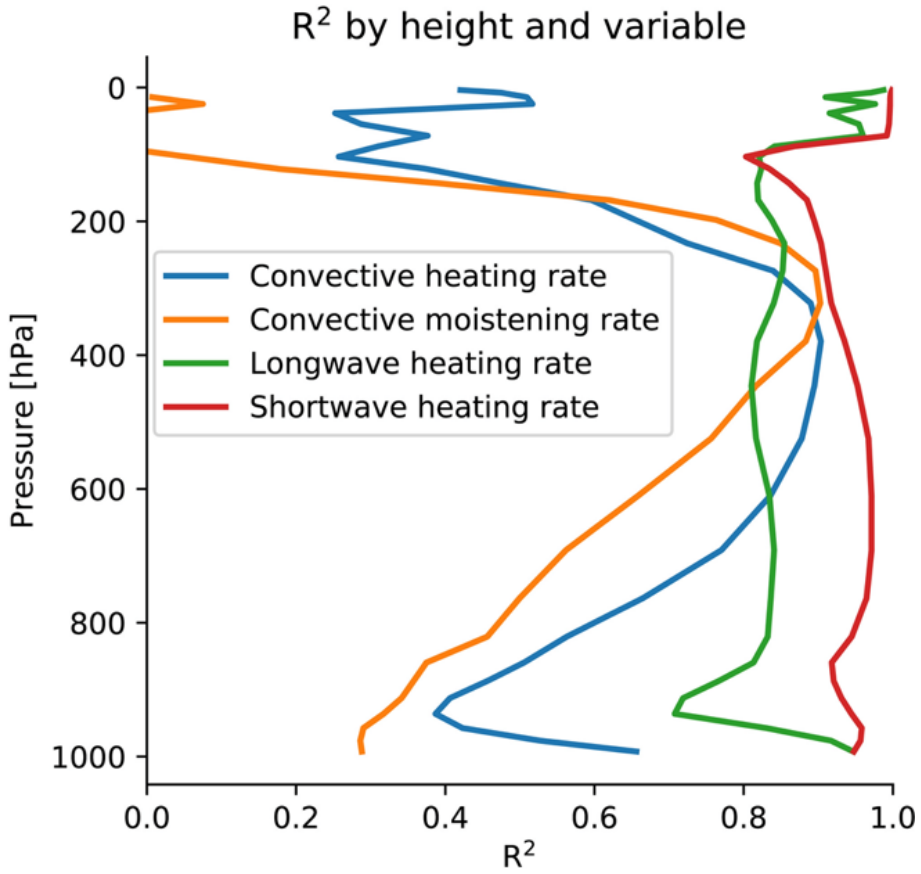


Figure 2.3: *Profils verticaux de R^2 obtenus par Gentine et al. (2018) , pour chacune des quatre variables de sortie de CBRAIN (cf. légende de la figure). Alors que les scores sont excellents dans la haute-troposphère (400-300 hPa), les modèles ont plus de mal à représenter les processus de couche limite (1000-850 hPa), où la stochasticité joue un rôle très important dans la simulation SPCAM3. Le score de R^2 est plus faible dans la couche limite (vers 850 hPa), pour atteindre seulement 0.4 pour le taux de réchauffement convectif (convective heating rate), et environ 0.3 pour le taux d'humidification convectif (convective moistening rate). Figure extraite de Gentine et al. (2018).*

Brenowitz and Bretherton (2018, 2019)

Brenowitz and Bretherton (2018, 2019) visent également à apprendre la totalité de la physique à partir de données issues d'une simulation haute-résolution. Le modèle HR utilisé est System for Atmospheric Modeling (SAM), dont la résolution horizontale est de 4 km. Les données d'apprentissage décrivent la bande tropicale (entre 30°S et 30°N), sur une durée de 80 jours. Durant la simulation, les données sont enregistrées avec un pas de temps de 3h. Les données sont ensuite agrégées à une résolution horizontale de 160 km (Figure 2.2).

Les variables en entrée de la paramétrisation NN sont les profils verticaux de l'énergie statique liquide $s_L(z)$, et d'humidité totale $q_T(z)$, ainsi que les chaleurs latente et sensible de surface (LHF et SHF respectivement) et le rayonnement solaire incidente (SOLIN). Les variables en sortie sont les sources apparentes de

réchauffement $\mathbf{Q}_1$ et d'humidification $\mathbf{Q}_2$, analogues respectivement aux profils verticaux des tendances de température et d'humidité. Le NN apprenant la physique décrit dans Brenowitz and Bretherton (2018) est composé de deux couches cachées Dense. Le modèle le plus performant a 128 neurones sur chacune des deux couches. La fonction perte, grâce à laquelle les poids et biais du NN sont optimisés, est une métrique évaluée sur $T = 20$ pas de temps. En effet, le NN doit être performant au pas de temps, mais également dans la durée. De plus, optimiser les poids et biais en fonction de plusieurs pas de temps permet également d'améliorer la stabilité du NN obtenu.

La physique NN apprise est validée dans une version simplifiée du modèle de climat, implémentant une seule colonne, appelé *single column model* (SCM). Dans un tel modèle, les conditions aux bords sont fournis par un GCM. Cette simulation de validation dure 80 jours, durant lesquels les conditions aux limites de cette colonne, notamment les variables en entrée de la physique NN, sont issues du GCM cSAM. Les sorties de la physique apprise sont des variables *diagnostiques* : elles sont calculées à chaque pas de temps, mais elles n'interviennent pas dans le calcul de la physique. Globalement, la physique NN est assez performante. Mesuré par le score de R^2, la meilleure physique NN a un score de 0.63 pour Q_1 et 0.74 pour Q_2. Dans Brenowitz and Bretherton (2019), la validation du modèle précédemment obtenu est effectué dans le GCM cSAM directement, sur le domaine tropical. Malgré des bons résultats obtenus durant le test dans le SCM, la simulation implémentant la physique NN en remplacement de la physique de cSAM devient instable au bout de 9 jours simulés. L'analyse plus détaillée des résultats obtenus permet de mettre en évidence des incohérences. Par exemple, le tiers des précipitations sont négatives durant la simulation. De plus, les lois de conservation ne sont respectées que de manière approximative par cSAM avec la physique NN.

O'Gorman and Dwyer (2018)

Le problème des précipitations négatives est assez difficile à résoudre avec les NN. Pour contourner ce problème, on peut réaliser l'apprentissage de la physique à l'aide de Random Forests (RF), plutôt qu'avec des NN. Les principes de cet algorithme sont détaillés en annexe (Annexe B). Par construction, les prévisions d'un modèle RF restent dans l'intervalle exploré dans l'échantillon d'apprentissage. Cette propriété permet en particulier de garantir que les physiques RF conduisent à des précipitations toujours positives.

Le potentiel des (RF) pour l'apprentissage de la physique d'un modèle de climat est évalué dans O'Gorman and Dwyer (2018). Un RF est ajusté pour apprendre le schéma de convection profonde *Relaxed Arakawa-Schubert* (RAS, Moorthi and Suarez, 1992). Contrairement aux articles précédemment présentés dans cette section, le but ici est d'émuler le comportement d'un schéma physique. Les tendances de profils de température et d'humidité calculées par ce schéma sont estimées par le

RF en fonction des profils de température T, d'humidité q ainsi que de la pression de surface P_s. Les variables à émuler sont les profils verticaux de tendances d'énergie statique sèche et humide. Les profils verticaux (en entrée et en sortie du RF) comprennent 21 niveaux verticaux chacun. Les données pour réaliser l'apprentissage sont issues d'une simulation de durée 3300 jours, après un *spin-up* de 700 jours. Elles sont enregistrées à tous les pas de temps, soit $\Delta t = 10$ minutes. Le jeu de données ainsi disponible est de très grande taille. L'échantillon d'apprentissage est construit à partir des données issues de cette simulation par sous-échantillonnage aléatoire. Pour garantir une meilleure indépendance, on choisit dans un premier temps une longitude (aléatoire) et un pas de temps. Un second tirage aléatoire permet ensuite de déterminer la latitude des 10 colonnes retenues dans l'échantillon d'apprentissage. L'ordre des colonnes sélectionnées dans l'échantillon d'apprentissage est ensuite permuté de manière aléatoire. 70% de l'échantillon ainsi obtenu servira de base pour l'ajustement du modèle de RF, le reste est utilisé pour évaluer le modèle obtenu (échantillon test).

Les résultats obtenus sont évalués par le score de R^2 sur l'échantillon test. Le score obtenu est très bon, ($R^2 = 0.82$), notamment pour les précipitations ($R^2 = 0.95$). Les auteurs montrent que l'émulateur RF a réussi à apprendre le respect des lois de conservation de manière satisfaisante. La stabilité de la paramétrisation RF a également été évaluée en la branchant dans le GCM et en effectuant des simulations avec le modèle ainsi obtenu. Les auteurs ne constatent aucun problème d'instabilité pendant la durée de la simulation de validation, et relèvent une accélération d'un facteur trois par rapport au schéma RAS.

L'utilisation des RF pour apprendre des paramétrisations permet d'obtenir un schéma qui respecte (approximativement) les lois de conservation et qui est stable une fois implémenté dans le GCM. Toutefois, il existe des limitations pratiques faisant que, à l'heure actuelle, les RF ne sont que peu utilisés pour apprendre des schémas physiques. D'abord, le volume numérique des RF est relativement élevé. Le RF utilisé par les auteurs est de taille 110 Mo, ce qui est considérable. Dans un GCM, la physique est appelée à chaque intégration, ce qui signifie que la physique RF doit être gardée en mémoire. À titre de comparaison, le volume du NN de Gentine et al. (2018) est de 7 Mo. Deuxièmement, comme nous l'avons vu dans le Chapitre 1, les RF disposent de moins d'options d'optimisation que les NN. Plus tard, il a été montré que les paramétrisations RF sont souvent moins performantes que les paramétrisations NN (Yuval et al., 2021).

2.2.2 Correction des paramétrisations

Une deuxième possibilité pour améliorer la physique d'un GCM consiste à utiliser des techniques de ML pour corriger les paramétrisations existantes. Une première utilisation de cette méthode de correction est décrite dans Watson (2019), qui utilise des NN pour corriger une physique estimée du modèle Lorenz'96 (Lorenz, 1996). Dans

le cadre de modèles de climat, Yuval and O'Gorman (2020); Yuval et al. (2021) décrivent une approche similaire dans le but d'améliorer la physique d'un GCM en utilisant des techniques de ML. Dans la suite, nous nous intéressons à la technique décrite dans ces deux derniers travaux.

Soit $\mathbf{x}$ une variable décrivant l'état thermodynamique de l'atmosphère. Si les tendances associées calculées par le GCM sont notées $\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)_{\mathrm{GCM}}$ et les tendances agrégées sur les données issues d'une simulation haute-résolution $\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)_{\mathrm{HR}}$, la correction ε à apprendre s'exprime

$$\varepsilon = \left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)_{\mathrm{GCM}} - \left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)_{\mathrm{HR}}. \tag{2.1}$$

Cette correction englobe la totalité de la physique. Elle permet notamment de compenser les erreurs dues à une modélisation insuffisante des interactions entre les différents processus paramétrisés.

L'échantillon d'apprentissage est obtenu à partir d'une simulation haute-résolution utilisant le modèle SAM en configuration aquaplanet. L'échantillon d'apprentissage couvre un domaine s'étalant de 78,5°S à 78,5°N, pour une largeur de 6912 km (Figure 2.4). Le modèle HR a une résolution horizontale de 12 km. Les sorties de ce modèle sont ensuite agrégées pour réduire la résolution horizontale d'un facteur d'agrégation 4, 8, 16 ou 32, notées respectivement $\times 4, \times 8, \times 16, \times 32$. Ainsi, la résolution horizontale du modèle grossier $\times 8$ est de 96 km. Le modèle basse-résolution correspond à une simulation SAM de résolution horizontale identique à celle du modèle haute-résolution agrégé. La résolution temporelle est de 3h, trop grande pour que les tendances représentent certains processus de fine-échelle de manière satisfaisante, mais elle est suffisante pour l'approche envisagée par les auteurs.

Pour l'apprentissage de la correction des tendances issues de la physique, Yuval and O'Gorman (2020) utilisent deux RF. Un premier est dédié à la correction des tendances combinées de l'advection verticale, la microphysique, la sédimentation, la précipitation et le schéma radiatif LW. Les variables en entrée de ce premier modèle sont les profils sur 45 niveaux de température T, de rapport de mélange non-précipitation q_T et précipitant q_p, ainsi que la distance à l'équateur de la maille $|y|$ issues du modèle basse-résolution. La correction est apprise pour les profils verticaux de tendance de l'énergie statique humide h_L et des rapports de mélange q_T et q_p. Le deuxième RF apprend la correction pour les processus diffusifs. Les variables en entrée du second RF sont les profils sur 45 niveaux de T, de q_T, de vents zonal u et méridien v et de vent de surface $\mathrm{wind}_{\mathrm{surf}}$, ainsi que $|y|$. La correction est apprise pour la diffusivité trubulente agrégée $\overline{D}$ sur 15 niveaux verticaux, ainsi que h_L et q_T à la surface.

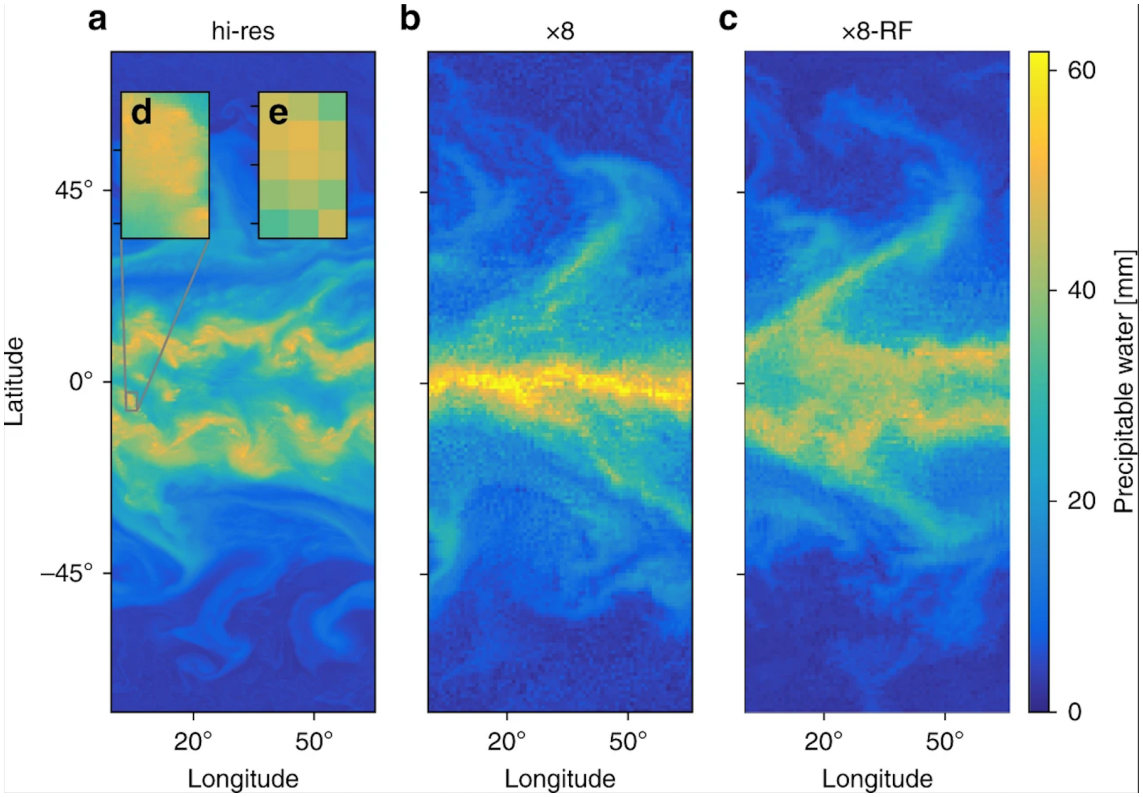


Figure 2.4: *Simulation haute-résolution et échantillon d'apprentissage utilisé dans Yuval and O'Gorman (2020); Yuval et al. (2021). (a) La simulation haute-résolution est réalisée en configuration aquaplanet, de résolution horizontale 2.5 km. Le domaine utilisé est une bande de largeur 6912 km pour des latitudes allant de -78,5° à 78,5°. (b) L'échantillon d'apprentissage utilise les sorties d'un modèle de résolution 8 fois plus lâche (noté "x8") que le modèle haute-résolution : un point de grille de la simulation x8 a un côté 8 fois plus grand que ceux dans la simulation haute-résolution. Le détail (d) et (e) illustre plus finement la différence entre les deux mailles. (c) Simulation réalisée avec une physique corrigée par un modèle de RF. L'utilisation de la correction RF permet de retrouver des détails n'ayant pas été vus dans la simulation x8, notamment au niveau de l'équateur. Figure extraite de Yuval and O'Gorman (2020).*

Dans Yuval et al. (2021), les auteurs reprennent la même approche de correction de la physique, mais utilisent un NN à la place de chaque RF. Ce choix est motivé par une meilleure performance des NN à apprendre le terme correctif ε. Les NN sont composés de plusieurs couches Dense. La seconde différence par rapport à Yuval and O'Gorman (2020) est d'apprendre une correction pour les termes source équivalents aux tendances apprises avec des RF. Les termes source sont des variables intermédiaires qui seront utilisées dans les calculs effectuées dans d'autres parties du GCM, alors que les tendances (apprises par les RF) sont directement intégrées. Garder les équations qui utilisent les termes sources corrigés permet d'obtenir une stabilité équivalente à celle des paramétrisations physiques.

Depuis tout récemment, l'approche présentée dans le cas d'un modèle en configuration aquaplanet a été étendue aux GCM (Bretherton et al., 2022; Clark et al., 2022).

2.3 Les questions encore ouvertes

2.3.1 Problèmes de stabilité

Une fois que la physique apprise a été branchée dans le GCM, la stabilité du modèle résultant ne peut être garantie. Au contraire, de nombreuses études mettent en évidence le caractère instable des GCM avec une physique NN (e.g., Brenowitz and Bretherton, 2018, 2019; Beucler et al., 2021a). Dans la suite, nous allons passer en revue les différentes méthodes proposées pour garantir la stabilité numérique du GCM une fois que la physique apprise a été branchée.

Comme nous l'avons déjà évoqué dans la section précédente, Brenowitz and Bretherton (2018, 2019) proposent d'ajouter un terme de pénalité à la fonction perte utilisée pendant l'apprentissage, pénalisant les NN peu performants pour la simulation d'une série temporelle de durée T. Le NN utilisé est composé de plusieurs couches Dense. Grâce au terme de pénalité ajouté à la fonction perte, les poids θ sont optimaux au pas de temps, mais également pour la prévision d'une série temporelle. Pour rappel, un NN est ajusté pour la prévision des termes sous-maille de chauffage $\mathbf{Q_1}$ et d'humidification $\mathbf{Q_2}$, en fonction des variables pronostiques $\mathbf{x}$. Dans la suite, nous allons détailler la fonction perte utilisée par les auteurs, qui s'exprime sous la forme de deux termes.

1. Une fonction perte *instantanée*, notée J_{instant}, pénalisant les modèles de NN les moins performants au pas de temps, pour chaque colonne i de cSAM :

$$J_{\text{instant}}(\theta) = \sum_{i,n} \left\| f(\mathbf{x}(t_n);\theta) - \begin{bmatrix} \mathbf{Q_1} \\ \mathbf{Q_2} \end{bmatrix} \right\|_M^2, \tag{2.2}$$

   avec f le modèle de NN. La norme $\|.\|_M^2$ est une norme quadratique pondérée par le profil de densité d'une colonne atmosphérique.

2. Une fonction perte *de stabilité*, calculée pour chaque colonne i sur plusieurs pas de temps successifs de durée T :

$$J_{\text{stability}} = \sum_{i,t_0} \|\mathbf{x}^{\text{mean}}(t_0+T;t_0,\theta) - \langle \mathbf{x}_i^o \rangle\|_M^2 \tag{2.3}$$

   avec $\mathbf{x}_i^{\text{mean}}$ la variable d'état intégré sous le forçage moyen de la période t_0 à t_0+T, comparée à $\langle \mathbf{x}_i \rangle$, la moyenne temporelle sur la même période des "observations" (données issues de la simulation haute-résolution).

La fonction perte totale s'écrit donc :

$$J_{\text{total}}(\theta) = J_{\text{instant}}(\theta) + J_{\text{stability}}(\theta). \tag{2.4}$$

Bien que le principe paraisse prometteur, les auteurs soulignent que l'utilisation seule de cette métrique d'apprentissage ne permet pas au modèle cSAM de rester stable une fois la physique apprise branchée. Dans Brenowitz et al. (2020a), les auteurs présentent une étude pour tenter de mieux comprendre l'origine de l'instabilité *online*. Entre autres, les auteurs ont effectué une analyse de sensibilité en utilisant des cartes de sensibilité (*saliency maps*, Simonyan et al., 2014), dont l'utilisation est très répandue pour les réseaux convolutifs. Grâce à celle-ci, les auteurs ont remarqué que le NN a appris une relation de corrélation entre l'humidité des couches supérieures de l'atmosphère et les tendances en sortie. Cette relation n'a aucun fondement physique et peut être à l'origine d'une baisse de performances du schéma sur certaines situations (et donc de l'instabilité observée une fois la physique apprise branchée dans le GCM). Les auteurs ont donc décidé d'enlever les plus hautes couches des profils verticaux en entrée de la physique NN. L'utilisation d'une fonction perte pénalisée ainsi que d'avoir enlevé les plus hauts niveaux des profils verticaux ont permis de stabiliser le GCM avec la physique NN. Les auteurs ont réalisé une simulation d'une durée de 100 jours, sans divergence.

Il convient toutefois de remarquer que, à ce jour, aucune méthode ne permet de garantir qu'une physique apprise sera stable une fois implémentée dans un GCM. Après avoir constaté que le NN conduisait vers un modèle instable, on peut seulement essayer d'en comprendre l'origine, par exemple en utilisant des méthodes similaires à celles présentées dans Brenowitz et al. (2020a), et proposer des solutions pour résoudre les problèmes identifiés *a posteriori*.

2.3.2 Le respect des lois de conservation

Le non-respect des lois de conservation par les physiques NN est un facteur favorisant la divergence du GCM avec une physique IA. Dans le cas de l'émulation d'un schéma de convection profonde par des RF, O'Gorman and Dwyer (2018) montre que le modèle où la physique RF a été branchée respecte les lois de conservation de la masse et de l'énergie de manière satisfaisante. Il explique que ces lois sont linéaires, donc stables par combinaison linéaire. Toutefois, nous ne savons pas si une physique apprise à partir de données issues d'une simulation HR respectera également ces lois de conservation.

Dans le cas d'un apprentissage de la physique avec un NN, il est possible d'imposer le respect des lois de conservation. On peut remarquer que la formulation mathématique des lois de conservation de la masse (équation de continuité) et de l'énergie (cf. Annexe A) s'exprime sous la forme de deux relations simples. Nous avons vu qu'il est possible d'ajouter des termes de pénalité à la fonction perte pendant l'apprentissage. Une des possibilités pour imposer le respect des lois de conservation est de pénaliser le non-respect de ces lois pendant l'apprentissage, *via* l'ajout d'un terme de pénalité

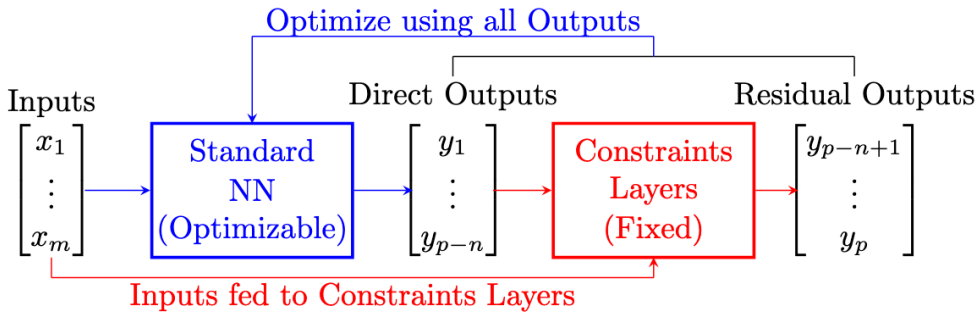


Figure 2.5: *Imposer les lois de conservation via l'architecture du NN. (bleu) La première partie du NN correspond au NN utilisé dans la première approche. (rouge) Une couche supplémentaire est ajoutée à la fin du premier bloc. Les entrées de cette sortie sont les sorties du premier bloc ainsi que les variables en entrée du modèle. Les poids de cette couche sont définies manuellement, de sorte à retrouver l'Éq. 2.5. Celles-ci seront respectées à la précision de la machine. Figure empruntée à Beucler et al. (2021a).*

à la fonction perte.

L'implémentation des lois de conservation dans les physiques NN est l'objet de Beucler et al. (2021a). Pour ce faire, une fois les contraintes exprimées sous forme linéaire, la première étape consiste à traduire les lois de conservation sous forme d'une matrice de contraintes $\mathbf{C}$, en fonction des variables en entrée du NN, $\mathbf{x}$, et l'estimation des variables de sortie par le modèle du NN, $\widehat{\mathbf{y}}$. Les lois de conservation $(\mathcal{C})$ s'expriment alors avec l'aide de la matrice des contraintes :

$$(\mathcal{C}) = \left\{ \mathbf{C} \begin{pmatrix} \mathbf{x} \\ \widehat{\mathbf{y}} \end{pmatrix} = \mathbf{0} \right\}. \tag{2.5}$$

Il est possible que les variables en entrée du NN ne comprennent pas toutes les variables nécessaires pour l'expression des lois de conservation. Ces variables supplémentaires doivent être ajoutées en entrée du NN, mais celles-ci seront utilisées uniquement à l'étape du calcul de la fonction perte pénalisée.

Les auteurs proposent ensuite deux méthodes pour contraindre le NN à respecter les lois de conservation. La première consiste à ajouter à la fonction perte utilisée pour l'apprentissage un terme $\mathcal{P}$ pénalisant le non-respect des lois de conservation (*soft constraints*). Cette approche est très proche de la méthode décrite dans la section précédente : la pénalité vient s'ajouter à une fonction perte classique (ici la MSE), visant à obtenir le de NN plus performant possible au pas de temps. Le terme de pénalité s'exprime

$$\mathcal{P}(\mathbf{x}, \widehat{\mathbf{y}}) = \left\| \mathbf{C} \begin{pmatrix} \mathbf{x} \\ \widehat{\mathbf{y}} \end{pmatrix} \right\|^2, \tag{2.6}$$

où $\|.\|^2$ est la norme 2. La fonction perte finale, notée $\mathcal{L}$, s'exprime en fonction de $\alpha \in [0, 1]$, dont la valeur optimale est à déterminer :

$$\mathcal{L}(\alpha) = \alpha\mathcal{P}(\mathbf{x}, \widehat{\mathbf{y}}) + (1 - \alpha)\text{MSE}(\mathbf{y}, \widehat{\mathbf{y}}), \tag{2.7}$$

avec $\mathbf{y}$ la variable cible.

La deuxième méthode consiste à imposer les contraintes dans l'architecture du NN, à travers une couche cachée supplémentaire s'ajoutant à la fin de celui-ci (*hard constraints*, Figure 2.5). Les poids et biais de cette couche sont choisis manuellement pour que la couche réalise les opérations traduites dans la matrice des contraintes $\mathbf{C}$. Le respect des lois de conservation ($\mathcal{C}$) est ainsi apprise par le NN à la précision de la machine. La fonction perte utilisée est la MSE, sans pénalité supplémentaire.

Dans les deux cas, les lois de conservation sont bien respectées. L'utilisation de la 2ème méthode conduit à des résultats légèrement meilleurs que ceux obtenus en appliquant la première méthode. Malgré les contraintes, le NN issu de la deuxième méthode est seulement 3% moins performant que le même modèle sans couche de contrainte (évaluation avec le R^2), et les lois de conservation sont respectées à 10^{-9}% (i.e., à la précision de la machine). Il convient toutefois de souligner que ces performances ont été évaluées *offline* seulement, en utilisant un échantillon de validation.

La stabilité numérique d'un schéma de paramétrisation est essentielle dans un modèle de climat. Dans le cas des physiques apprises en utilisant des techniques de ML, cette propriété ne peut pas être garantie, notamment avec les NN. Mais il existe plusieurs techniques pour rendre les modèles implémentant une physique apprise plus stables. Brenowitz and Bretherton (2018, 2019) proposent de pénaliser les NN les moins stables pendant l'apprentissage à travers l'utilisation d'une fonction perte pénalisée, tout en retirant les variables en entrée identifiées à l'origine de l'instabilité observée grâce à une analyse de sensibilté. Beucler et al. (2021a) identifie un respect insuffisant des lois de conservation comme étant un facteur favorisant la divergence numérique constatée. Il propose deux méthodes pour contraindre le NN à apprendre, en plus de la physique, le respect des lois de conservation.

2.3.3 La simulation du changement climatique

Les modèles de climat sont principalement utilisés pour prévoir le climat résultant de plusieurs scénarii de réchauffement climatique. Est-ce toujours possible avec une physique ML ayant été ajustée à des sorties de modèles haute-résoltion ? La réponse est, *a priori*, non. Comme nous l'avons vu dans le Chapitre 1, les algorithmes de ML permettent d'estimer des fonctions sur un espace implicitement défini par l'échantillon d'apprentissage. Rien ne garantit donc que les modèles ainsi obtenus soient généralisables en dehors de l'espace ayant été visité durant l'apprentissage. Dans le cadre du réchauffement climatique, la variable d'état de l'atmosphère peut

sortir des régions explorées par les simulations réalisées avec des modèles haute-résolution. Comment échantillonner le sous-espace dans lequel la variable d'état se trouvera d'ici la fin du siècle ?

Élargissement de l'échantillon d'apprentissage

La première solution consiste à élargir l'échantillon d'apprentissage, en incluant des données issues de simulations pour différents scénarii de réchauffement climatique. La mise en œuvre de cette idée est notamment détaillée dans O'Gorman and Dwyer (2018). Deux simulations sont réalisées : une première simulation de *contrôle* avec une température moyenne globale à 288K et une seconde simulation *chaude* à 295K, obtenue en augmentant la concentration des gaz à effet de serre.

Dans leur première expérience de référence, le RF apprend sur les données issues de la simulation de contrôle uniquement. Lorsque ce RF est implémenté dans le GCM en configuration chaude, le modèle ne parvient pas à simuler la bonne intensité du changement climatique. Celle-ci est trop élevée, notamment au niveau de la bande tropicale. Le RF n'a pas rencontré des situations aussi chaudes et des conditions de troposphère aussi hautes dans l'échantillon d'apprentissage que celles observées dans la simulation chaude. À l'inverse, lorsque l'on utilise un RF ayant appris le schéma RAS sur les données issues de la simulation chaude, celui-ci est performant dans le GCM utilisé en configuration contrôle. Cela permet de confirmer l'hypothèse selon laquelle il est nécessaire d'avoir un échantillon d'apprentissage décrivant l'ensemble des variables rencontrées lors de la validation. La simulation chaude comporte, à des latitudes plus élevées, des individus compatibles avec la simulation de contrôle. Cela semble suffisant pour que le RF les apprenne et soit performant lorsqu'il est utilisé dans le GCM contrôle. Comme dans la conclusion de Rasp et al. (2018), les auteurs suggèrent d'utiliser des données issues des deux simulations chaude et contrôle pour créer un échantillon d'apprentissage optimal pour le RF.

Adimensionnaliser les variables

Outre la simulation du changement climatique, il existe de nombreux exemples pour illustrer les problèmes de généralisation avec les algorithmes de ML. La présence de surfaces continentales ou de relief (modifiant l'altitude de la maille) en font partie. Dès lors, échantillonner l'ensemble des situations que la physique apprise rencontrera au cours d'une simulation apparait difficilement réalisable. De plus, ne pas tenir compte d'une situation dans l'échantillon d'apprentissage signifie également qu'il soit nécessaire d'effectuer un second apprentissage pour obtenir un schéma valable pour le cas d'utilisation n'ayant pas été envisagé, ce qui est numériquement coûteux.

Afin de résoudre ce problème, Beucler et al. (2021b) propose une nouvelle approche. On peut remarquer que la difficulté du NN à être généralisable en dehors de son échantillon d'apprentissage est la conséquence d'un problème de représentativité

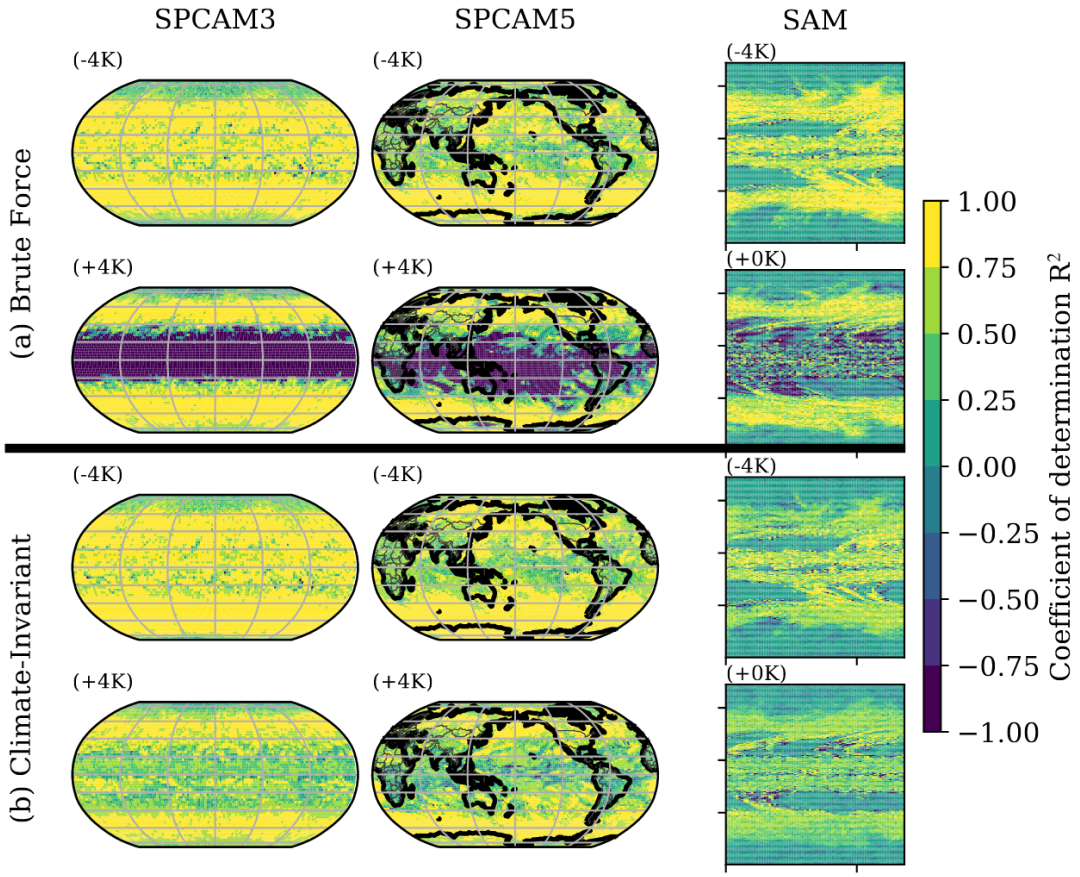


Figure 2.6: *Cartes de score de R^2 moyen, pour différentes configurations d'apprentissage. (a) En utilisant le NN brute force, i.e. sans adimensionnalisation des données. (b) En utilisant le climate-invariant NN, i.e. en adimensionnalisant les données. On constate que le climate-invariant NN obtient de meilleurs scores sur l'échantillon de validation que le NN brute force, quelle que soit la configuration. Figure extraite de Beucler et al. (2021b).*

de distribution. Plutôt que d'essayer d'échantillonner l'ensemble de la distribution décrite par les différentes variables en jeu dans le NN, il est possible d'effectuer un *changement de variable* pour réaliser uniquement des quantités relatives. Cette technique est empruntée à la mécanique des fluides, où les changements de variables sont pilotées par la valeur des différents nombres adimensionnés, tels que le nombre de Reynolds pour quantifier le régime turbulent d'un écoulement, ou encore le nombre de Mach quantifiant la compressibilité d'un écoulement fluide. Pour les paramétrisations ML, on constate que l'utilisation de l'humidité relative RH parmi les variables du modèle plutôt que de l'humidité totale q_T permet d'obtenir de meilleurs résultats (e.g., O'Gorman and Dwyer, 2018). RH est précisément un changement de variable permettant de relier q_T (quantité absolue) à une forme relative, prenant des valeurs entre 0 et 1 uniquement.

Beucler et al. (2021b) propose un cadre théorique pour effectuer de tels change-

ment pour l'ensemble des variables intervenant dans une une physique apprise par un NN. Cela permet d'apprendre la physique indépendamment des scénarii d'émissions ou du relief. Les auteurs réalisent l'apprentissage de la physique avec des NN à partir des différents jeux de donées.

- Une première série de NN apprend sur les données issues de la même simulation que l'échantillon de validation, sans effectuer de changement de variable. Ces physiques NN sont appelées *brute force* (e.g., apprentissage et validation sur les données issues de la simulation +4K).

- Une seconde série de NN apprend sur le jeu de données issues d'une des simulations où les changements de variables ont été effectuées, et est validée sur les données issues d'une simulation de nature différente (e.g. apprentissage sur les données issues d'une simulation aquaplanet et validation sur celles issues d'une simulation comprenant les surfaces continentales). Ces NNs sont appelées *climate-invariant*.

On constate que, pour une simulation donnée, les *climate invariant* NN permettent d'obtenir de meilleurs résultats en terme de R^2 que les NN *brute force* (Figure 2.6). Effectuer le changement de variable permet de réaliser un apprentissage plus performant que l'utilisation de données brutes.

L'utilisation d'une physique NN n'est pas incompatible avec la prévision du changement climatique. Néanmoins, tout comme dans le cas des paramétrisations physiques traditionnelles, il faut soigneusement vérifier que celle-ci se généralise aux climats plus chauds. Si tel n'est pas le cas, rien ne garantit que le modèle obtenu est généralisable au-delà de son échantillon d'apprentissage (problème *out-of-sample*). Pour éviter ce problème dans le cas de l'apprentissage de la physique, nous avons présenté plusieurs méthodes. La première consiste à inclure des données issues de simulations plus chaudes dans l'échantillon d'apprentissage. La seconde consiste à effectuer des changements de variables pour les variables en entrée et en sortie du modèle de ML, pour travailler uniquement avec des quantités relatives, quel que soit le climat. Bien que les deux méthodes permettent d'obtenir de bons résultats lors de la validation *offline*, la deuxième semble plus robuste pour éviter les problèmes *out-of-sample*.

2.4 Perspectives

Dans ce chapitre, nous avons parcouru les travaux traitant du développement de paramétrisations utilisant des méthodes de ML. Au début du développement de ces paramétrisations (2005), le but était d'accélérer le schéma radiatif très coûteux en utilisant un émulateur moins coûteux et suffisamment performant à sa place. L'objectif a ensuite évolué : désormais, on cherche à faire mieux que les schémas physiques traditionnels, souvent développés sous certaines hypothèses simplificatrices difficiles à vérifier dans la pratique. Mais les premiers travaux visant à effectuer

l'apprentissage de la physique à partir de données issues d'une simulation haute-résolution ont été arrêtés par manque du puissance de calcul. Depuis la fin des années 2010, les supercalculateurs sont suffisamment performants pour réaliser des simulations à très haute résolution, dont les données peuvent être utilisées pour créer des échantillons d'apprentissage : cela a permis de relancer les travaux sur le développement de paramétrisations plus précises en utilisant des algorithmes de ML.

Bien que des résultats encourageants aient été obtenus, certains problèmes empêchant l'utilisation des physiques apprises dans les modèles de climat restent partiellement résolus ou seulement identifiés. Premièrement, les schémas appris ne sont pas forcément stables une fois branchés dans un GCM, et nous ne disposons d'aucune technique permettant de garantir cette stabilité *online*. Nous avons toutefois pu identifier quelques éléments favorisant l'instabilité, comme par exemple l'apprentissage de relations statistiques n'ayant aucun fondement physique ou encore l'utilisation de métriques d'apprentissage au pas de temps plutôt que sur une série temporelle (Brenowitz and Bretherton, 2019; Brenowitz et al., 2020a). Le non-respect des lois de conservation, pourtant essentiel dans les modèles de climat, a également été identifié comme un facteur favorisant l'explosion des modèles avec une physique apprise (Beucler et al., 2021a).

La deuxième question ouverte est la performance (et la pertinence) des schémas ML pour simuler le réchauffement climatique. Ce problème est un enjeu majeur de la modélisation du climat. Dans ce contexte, les schémas pourraient être amenés à visiter des régions de la distribution des différentes variables n'ayant pas été vus pendant l'apprentissage. Pour obtenir une physique apprise performante pour différents scénarii de changement climatique, une première méthode consiste à élargir l'échantillon d'apprentissage, en y incluant des données décrivant des climats plus chauds (O'Gorman and Dwyer, 2018). Cela nécessite toutefois d'effectuer des apprentissages à répétition si les caractéristiques de la distribution changent. Une deuxième méthode consiste à effectuer des changements de variable, pour travailler uniquement avec des quantités relatives. Cette technique est connue en mécanique des fluides, et son utilisation a permis d'obtenir de très bons résultat dans le cadre du développement des paramétrisations physiques apprises (Beucler et al., 2021b).

La question quant à la performance des physiques apprises dans le contexte du changement climatique pose celle de la confiance que nous accordons aux modèles. Dans le cas d'une paramétrisation physique traditionnelle, la connaissance des lois physiques utilisées pour les construire est le garant de leur généralisabilité lorsqu'il s'agit de prédire des valeurs nouvelles. Les NN sont, au contraire, considérés comme des boîtes noires. Nous ne comprenons pas vraiment comment ils fonctionnent. Le nombre très élevé de paramètres (poids et biais) nécessaires à leur définition ne fait qu'opacifier cette boîte noire. Ainsi, un des enjeux majeurs de l'utilisation des méthodes de ML en modélisation climatique doit s'accompagner du développement d'outils permettant de mieux les comprendre, comme par exemple des outils d'analyse

de sensibilité (e.g., McGovern et al., 2019; Brenowitz et al., 2020b; Toms et al., 2020).

Outre le fonctionnement des physiques NN, on peut également s'interroger sur les interactions entre la dynamique et la physique apprise. Étudier les interactions entre la physique apprise et la dynamique est une question compliquée. Pour mieux les comprendre, on peut se tourner vers l'étude de modèles jouets (e.g., Lorenz, 1963, 1996). Ces modèles exhibent certaines caractéristiques des modèles atmosphériques (e.g., chaoticité), mais le faible nombre de dimensions en jeu les rend plus faciles à manipuler que les modèles numériques du climat. Dans la suite, nous allons nous intéresser à la compréhension des problèmes de stabilité grâce à des modèles jouets.

Chapitre 3

Stabilité

3.1 Avant-propos

Dans le chapitre précédent, nous avons vu qu'un GCM où une paramétrisation NN a été branchée n'est pas nécessairement stable, et explose souvent au bout de quelques jours de simulation. Quelques facteurs favorisant l'instabilité des modèles avec une physique NN ont déjà été identifiés, comme par exemple le non-respect des lois de conservation ou encore l'apprentissage de relations statistiques n'ayant pas de sens physique (Brenowitz et al., 2020a). Mais aucune des méthodes proposées ne garantit la stabilité de la physique apprise une fois qu'elle est branchée dans le modèle de climat. L'étude de ces instabilités reste assez difficile à réaliser, en particulier pour des raisons techniques. D'un côté, on dispose de GCM, trop complexes pour montrer avec certitude que l'instabilité observée est imputable au non-respect des lois de conservation. D'un autre côte, les modèles jouets donnant une représentation simplifiée du comportement du fluide atmosphérique (e.g., Lorenz'63 et Lorenz'96, Lorenz, 1963, 1996) sont trop simples pour être instables lorsque leur physique est apprise par des NN (Rasp, 2020). Dans un post de blog[1] consécutif à Rasp (2020), Rasp s'intéresse plus particulièrement à l'étude des instabilités en utilisant les modèles Lorenz'63 et Lorenz'96. Le modèle Lorenz'63 est utilisé car le système est très bien connu et documenté. Mais Lorenz'96, moins connu, a la particularité d'implémenter une *paramétrisation physique* : elle est souvent utilisée pour réaliser des expériences préliminaires dans un cadre simplifié (e.g., en assimilation des données). Lorenz'96 est donc, à première vue, un candidat de choix pour l'étude des instabilités rencontrés lors du développement de paramétrisations NN. Mais Rasp montre qu'une simple régression linéaire suffit pour apprendre de manière satisfaisante la physique de ce modèle jouet. Le modèle Lorenz'96 dont la physique a été remplacée par une régression linéaire est stable. Rasp conclut que ce modèle jouet est trop simple pour qu'on puisse étudier les problèmes d'instabilité avec celui-ci.

Pour pallier au manque de modèle jouet suffisamment complexe et peu coûteux, nous en avons développé un permettant de reproduire et d'étudier ces instabilités. Ce modèle, appelé *embedded* Lorenz'63 model (eL63), peut être obtenu à partir du modèle Lorenz'63 en plongeant ses attracteurs dans un espace de dimension $d > 3$.

Dans ce chapitre, nous nous intéressons donc à l'étude des problèmes d'instabilité numérique, en nous appuyant sur le modèle eL63. La suite de ce chapitre est la retranscription d'un article publié dans *Geophysical Research Letters* (Balogh et al., 2021).

[1] `https://raspstephan.github.io/blog/lorenz-96-is-too-easy`.

Balogh, B., Saint-Martin, D. & Ribes, A. 2021.
Article publié dans Geophysical Research Letters.

3.2 Abstract

The development of atmospheric parameterizations based on neural networks is often hampered by numerical instability issues. Previous attempts to replicate these issues in a toy model have proven ineffective. We introduce a new toy model for atmospheric dynamics, which consists in an extension of the Lorenz'63 model to a higher dimension. While feedforward neural networks trained on a single orbit can easily reproduce the dynamics of the Lorenz'63 model, they fail to reproduce the dynamics of the new toy model, leading to unstable trajectories. Instabilities become more frequent as the dimension of the new model increases, but are found to occur even in very low dimension. Training the feedforward neural network on a different learning sample, based on Latin Hypercube Sampling, solves the instability issue. Our results suggest that the design of the learning sample can significantly influence the stability of dynamical systems driven by neural networks.

3.3 Plain Language Summary

Typical atmospheric models use parameterizations to represent processes occurring at a scale smaller than the model resolution (e.g., clouds, convection, etc.). Recently, artificial intelligence, and in particular neural networks, have been described as being of high interest to improve parameterizations. Nevertheless, first attempts to develop parameterizations based on machine learning have often led to numerical instability issues. Toy models commonly used in atmospheric research are not complex enough to allow the study of these instabilities, as they are easily learnt by neural networks. Here, we introduce a new toy model for atmospheric dynamics. Neural networks fail to reproduce the dynamics of this new toy model, resulting in unstable trajectories. However, training the neural network on a specifically designed learning sample, which explores the phase space comprehensively, solves the instability issues. Our results suggest that the design of the learning sample can significantly influence the stability of dynamical systems driven by neural networks. These results are expected to have practical impacts on the development of neural-network based parameterizations, and, thus, on the improvement of atmospheric models.

3.4 Introduction

Many efforts have been made over the last few years to take advantage of artificial intelligence (AI) in atmospheric and climate modeling. One line of research has sought to develop new, data-driven parameterization schemes to replace part of an atmospheric model (e.g., Brenowitz and Bretherton, 2018; Gentine et al.,

2018; O'Gorman and Dwyer, 2018) ; learning a ML-based parameterization scheme means learning to predict time derivatives for sub-grid scale atmospheric processes (e.g., turbulence, convection). Even though they promise numerically affordable yet accurate physics for low resolution atmospheric models (e.g., climate models), current state-of-the-art AI parameterizations are still biased and, more importantly, they face numerical instability issues. As reported by Rasp (2020), neural networks (NNs) are often numerically unstable, when coupled to the large-scale atmospheric fluid mechanics solver (e.g., Rasp et al., 2018; Brenowitz and Bretherton, 2019). Parameterizations based on random forests (RF) have been reported to be stable (Yuval and O'Gorman, 2020). But, when compared offline, NN-based parameterizations seem to outperform RF-based parameterizations (Brenowitz et al., 2020a).

The lack of stability in NN-based parameterizations can be interpreted as a result of breaking physical laws. Some authors propose techniques to ensure compliance with these laws (e.g., Beucler et al., 2021a). Brenowitz et al. (2020a) shows that instabilities are related to the linearized behavior of NNs when coupled to idealized wave dynamics. To fix numerical instabilities, Rasp (2020) proposes a coupled online learning method where a NN model, which was first trained offline, is continuously corrected according to online prediction errors. This concept is illustrated using the Lorenz'96 model (Lorenz, 1996).

To better understand the origin of stability issues, resorting to a toy model can be useful. Lorenz'96 (hereafter, L96 Lorenz, 1996) and Lorenz'63 (hereafter, L63 Lorenz, 1963) models often serve as toy models for atmospheric modeling, drawing simplified versions of the atmospheric flow. They have been extensively used as a test bed for assessing the use of data-driven methods in atmospheric models. L63 model has been accurately learnt by feedforward neural networks (Scher and Messori, 2019) or reservoir networks (Pathak et al., 2017). L96 dynamics has also been accurately learnt by feedforward NNs, recurrent NNs or generative adversarial networks (Dueben and Bauer, 2018; Chattopadhyay et al., 2020; Gagne et al., 2020), without reporting stability issues. Both L63 and L96 toy models appear to be insufficiently complex to study numerical instabilities encountered with more complex systems.

Currently, no toy model for atmospheric dynamics is currently available to investigate such instability issues encountered when developing parameterization schemes. The first aim of this paper is to introduce a simple, chaotic system challenging feedforward NNs by exhibiting numerical instabilities. Even though the two-level L96 model has often been used to address parameterization problems (i.e., learning only part of the dynamical system), this system will be based on the L63 model. This new toy model is aimed at replicating stability issues, when learning to predict time derivatives of the dynamical system with NNs. The second objective is to propose a possible method to ensure stability of the NN-generated trajectories, which in our case involves using a specific learning sample. Both objectives can be seen as steps

towards solving numerical instabilities – an issue which remains challenging to develop suitable NN-based parameterizations for atmospheric models.

Section 3.5 describes the Lorenz'63 model and illustrates that this model is not complex enough to challenge NNs. In Section 3.6, we introduce the embedded Lorenz'63 model, and show that learning of this simple model with NNs leads to unstable trajectories. In Section 3.7, a method to address these instability issues is proposed.

3.5 Learning the Lorenz'63 model

By expanding the set of partial differential equations of Rayleigh-Bénard convection into Fourier series and then truncating, Lorenz (1963) derives a finite system of ordinary differential equations, given by:

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= x_1(\rho - x_3) - x_2, \\ \dot{x}_3 &= x_1 x_2 - \beta x_3. \end{aligned} \tag{3.1}$$

The evolution of the state variable $\mathbf{x} = (x_1, x_2, x_3)$ is governed by the set of parameters (σ, ρ, β). L63 admits a chaotic solution for some values of these parameters, in particular $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$. We use these values in the following. Although computationally cheap, L63 is able to mimic some properties of the atmospheric flow. Hence, it is a suitable toy model to perform proof-of-concept experiments.

Recently, the L63 model has been used as a test bed for assessing the use of data-driven methods in atmospheric models. The system of equations (4.4) can be formally rewritten:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)). \tag{3.2}$$

Data-driven methods provide an approximate function, $\widehat{f}$, to replace the 'true' function f, and thus they replace the initial dynamical system by:

$$\dot{\mathbf{x}}(t) = \widehat{f}(\mathbf{x}(t)). \tag{3.3}$$

In this way, the dynamical system is learnt through learning its derivative. Some authors have also proposed to directly learn (and predict) the future state $\mathbf{x}(t + \Delta t)$ from the current state $\mathbf{x}(t)$ (e.g., Scher and Messori, 2019; Weyn et al., 2019, 2020). Here, we focus on learning time derivatives, as we think it is more representative of the parameterization problem.

For the L63 model specifically, various functions $\widehat{f}$ have been proposed recently, e.g., using sparse linear regression (Brunton et al., 2016), feedforward neural networks (Scher and Messori, 2019), or reservoir networks (Pathak et al., 2017). Fitting

$\widehat{f}$ always involves an optimization over a set of free parameters θ. This optimization is done for a given loss function, and a given learning sample. In the case of dynamical systems, the learning sample is typically a time series, also called an *orbit*, obtained by numerical integration of the system, e.g., Eq. (3.2). We note $[\mathbf{x}^{orb}]$ this learning sample:

$$[\mathbf{x}^{orb}] = \{(\mathbf{x}^n, f(\mathbf{x}^n))\}_{n=1..N}, \tag{3.4}$$

where $\mathbf{x}^n$ denotes the state of the system, $\mathbf{x}(t_n)$, at time t_n. To learn the L63 model, we use one single orbit as a learning sample. This orbit is obtained by integrating Eqs. (4.4) with a time-step of $\Delta t = 0.05$ using a fourth-order Runge-Kutta time stepping scheme. The numerical integration is performed over 500 model time units (MTU, where 1 MTU = 20 Δt). Thus, the learning sample contains $N = 10^4$ individuals. Given the fact that the learning sample is a time series, consecutive points exhibit positive auto-correlation. As a consequence, as previously done by Scher and Messori (2019), the learning algorithm is trained on the first 80% of the learning sample, while the remaining 20% are used for test or validation.

A simple, eight-layer feedforward NN is trained to derive $\widehat{f}$ using the mean squared error (MSE) as the loss function[2]. The hidden layers are activated by Recitifed Linear Unit function (ReLU) and the output layer is activated by a linear function. Learning is performed over 50 epochs, with Keras implementation of Adam optimizer, parameterized with an initial learning rate of 0.001. During training, the determination coefficient R^2 over the validation subset is monitored. The training loss function $\mathcal{L}$ is the mean squared error :

$$\mathcal{L}(\theta, [\mathbf{x}^{\text{train}}]) = \frac{1}{N} \sum_{n=1}^{N} \left\| \widehat{f}_\theta(\mathbf{x}^n) - f(\mathbf{x}^n) \right\|^2, \tag{3.5}$$

where $[\mathbf{x}^{\text{train}}]$ denotes the training dataset, N the size of the training dataset, $\mathbf{x}^n \in [\mathbf{x}^{\text{train}}]$ for $1 \leq n \leq N$, and θ denotes the set of parameters over which the optimization is made. Weights corresponding to the highest value of R^2 score over the validation dataset are saved and loaded after the training is completed. Learning is very efficient, as we get $R^2 = 0.9998$ over the validation subset.

The integration of both the L63 model (Eq. 3.2) and its NN-approximation (Eq. 3.3), with the same time-step of $\Delta t = 0.05$, using the same fourth-order Runge-Kutta time stepping scheme and starting from the same initial condition, show a good agreement (Figure 3.1). In particular, the NN-based model does not manifest any unstable behavior, i.e., its orbits always lay on the Lorenz 'butterfly' attractor. Repeating this analysis with different initial conditions and/or different learning samples leads to the same conclusion. This result suggests that the L63 model is

[2]From input to output layers, the number of nodes per hidden layer is as follows : 256, 128, 128, 128, 128, 128, 64, 32.

(i) Lorenz '63 orbit (ii) NN-based Lorenz '63 orbit

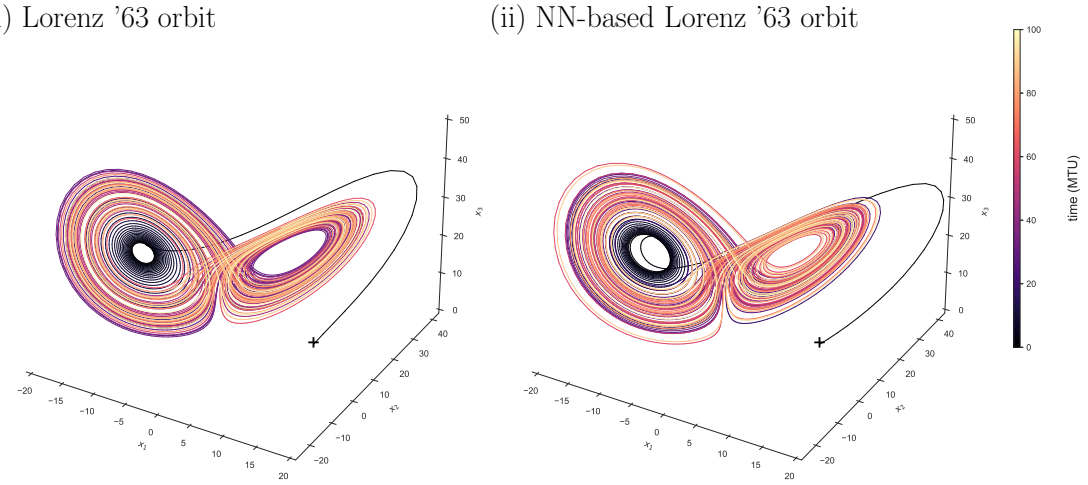


Figure 3.1: *Examples of Lorenz'63 orbits. (i) The orbit is obtained by integration of Lorenz'63 system of equations (see Eq. 4.4 in the text) with* σ=10, ρ=28 *and* β=8/3. *(ii) The orbit results from the integration of neural network model* $\widehat{f}$ *(see Eq. 3.3). The numerical integration is performed over 100 model time units from initial condition* $(10, 15, 0)$. *The orbits are colored by the time variable.*

not challenging enough to data-driven methods and does not allow to study instability issues, as previously shown in similar studies.

L96 provides another simple framework to study data-driven methods in the context of atmospheric modeling, and is of higher dimension than L63. However, reservoir computers and neural networks have succeeded in learning this system without reporting instability issues (Pathak et al., 2017; Scher and Messori, 2019). Hence, both L63 and L96 appears to be too simple to encounter the typical instability issues that still hamper the development of AI parameterizations.

3.6 The embedded Lorenz'63 model

Building a higher dimensional model on the basis of L63 has already been proposed. Musielak and Musielak (2009) added a fourth spatial variable to extend L63 equations. Champion et al. (2019) created a high-dimensional model on the basis of L63 system by using Legendre polynomials. Here, we propose a different extension of this model specifically designed to investigate instability issues.

The key idea is as follows: the standard L63 model, which is of dimension 3, is embedded into a larger space of dimension $d > 3$. In the selected 3-D subspace, the chaotic dynamics of L63 is kept unchanged. Any deviation from this 3-D subspace is brought back by a restoring force. We call 'embedded' L63 model (hereafter, eL63) this new simple model. A formal definition of eL63 is in two steps (see Figure 3.2).

(1) Embedding: we construct a vector $\mathbf{z} = (z_1, z_2, ..., z_d) \in \mathbb{R}^d$, and define its dynamics by:

$$\begin{aligned} \dot{z}_1 &= \sigma(z_2 - z_1), \\ \dot{z}_2 &= z_1(\rho - z_3) - z_2, \\ \dot{z}_3 &= z_1 z_2 - \beta z_3, \\ \dot{z}_j &= -\kappa z_j, \quad \forall j > 3. \end{aligned} \tag{3.6}$$

The first three equations are identical to L63, while the $d-3$ additional equations are simple restoring forces. For the sake of simplicity, the relaxation coefficient, κ, is unique. In the following, we use $\kappa = 1$. We denote $\mathcal{B}_z$ the basis of vector $\mathbf{z}$.

(2) Random rotation: We apply a random rotation to derive the state vector of the eL63 system, $\mathbf{x} = (x_1, x_2, ..., x_d)$:

$$\mathbf{x}(t) = P\mathbf{z}(t), \tag{3.7}$$

where $P \in \mathbb{R}^{d \times d}$ is the rotation matrix between $\mathcal{B}_z$ and $\mathcal{B}_x$, the basis of $\mathbf{x}$. P contains the coordinates of the basis vectors of $\mathcal{B}_x$, seen in $\mathcal{B}_z$. Note that the rotation matrix P does not depend on time.

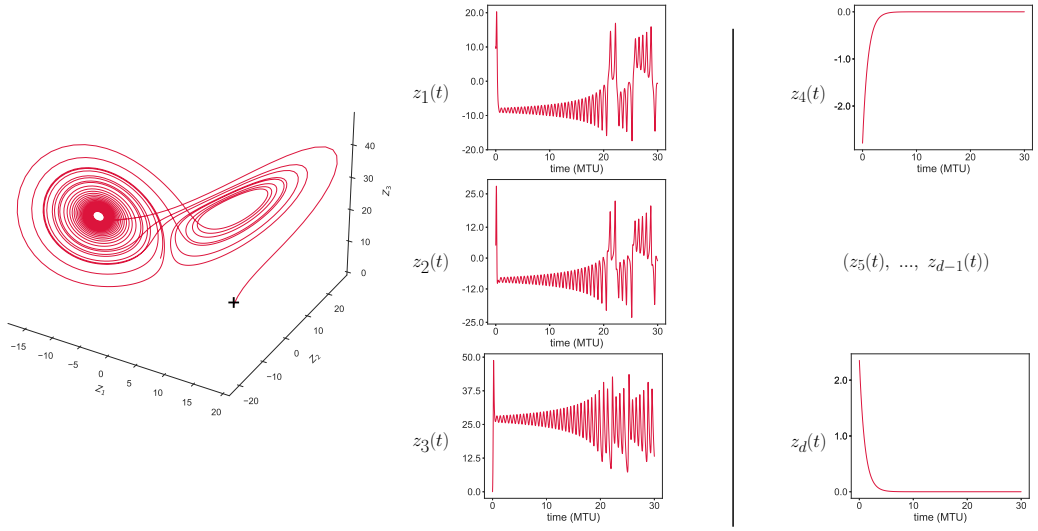
$\mathbf{x}$ is the state vector of eL63, and only $\mathbf{x}$ is available to a learning procedure. The dynamics of $\mathbf{x}$ is obtained by rewriting Equations (3.6) and (3.7), such that $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$. In this case, f is the function composition of the L63 plus restoring forces and the random rotation.

A key property of eL63 is that $z_{j,j>3}$ exponentially decays towards zero and so $\mathbf{x}(t)$ is confined within a subspace of dimension 3. This subspace can be interpreted as resulting, e.g., from physical but unknown constraints. The difficulty for a data-driven method to fully capture the dynamics of eL63 comes from the fact that any orbit is very thin – almost all points are located in a subspace of dimension 3. If such an orbit is used as a learning sample, any deviation from this specific subspace in a predicted trajectory can lead to an out-of-sample issue.

We now consider the problem of learning eL63, in the same way as done for L63 in the previous section. A learning sample $[\mathbf{x}^{orb}]$ is built by integrating eL63 equations over 500 MTUs, with a timestep of $\Delta t = 0.05$ using a fourth-order Runge-Kutta scheme. The initial condition is sampled from an eL63 orbit, which is equivalent to removing the model spin-up from the learning dataset, ensuring that $z_{j,j>3} \approx 0$. Therefore, the learning dataset contains $N = 10^4$ individuals. The learning sample

(i) Embedding : $\mathbb{R}^3 \mapsto \mathbb{R}^d$

$$(z_1(t), z_2(t), z_3(t)) \mapsto \mathbf{Z}(t) = (z_1(t), z_2(t), z_3(t), ..., z_d(t))$$



(ii) Random rotation : $\mathbb{R}^d \mapsto \mathbb{R}^d$

$$\mathbf{X}(t) = (x_1(t), x_2(t), x_3(t), ..., x_d(t)) = P\mathbf{Z}(t),\ P \in \mathbb{R}^{d\times d}$$

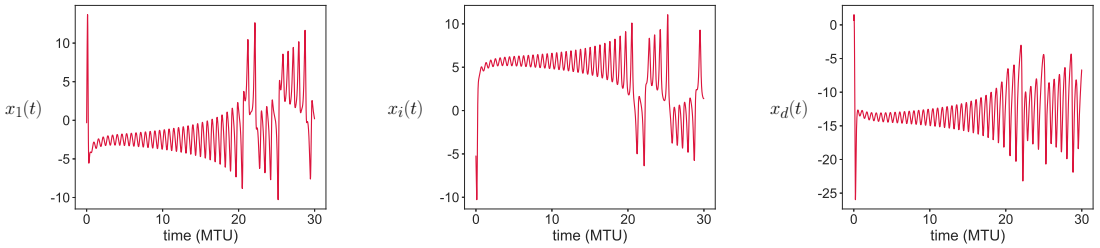


Figure 3.2: *The embedded Lorenz'63 model (see text for details). (i) Representation of an eL63 orbit in $\mathcal{B}_z$ basis over 30 MTU : (left) tri-dimensional representation of $(z_1(t), z_2(t), z_3(t))$, (middle) time series of $z_1(t)$, $z_2(t)$, and $z_3(t)$, (right) time series of $z_4(t)$ and $z_d(t)$. (ii) Representation of the same eL63 orbit in $\mathcal{B}_x$ basis. Time series of $x_1(t)$, $x_i(t)$ and $x_d(t)$ are shown.*

being a time series, consecutive points are positively auto-correlated. Hence, learning is performed over the first 80% of this sample, while the last 20% are saved for validation. Both the input and the target variables are then normalized so that their mean value equals 0 and their standard deviation equals 1. A simple feedforward NN is trained to best approximate f, in the same configuration than described in Section 3.5 (e.g., eight hidden layers, fit over 50 epochs, minimizing MSE loss function). As the learning sample consists of a single orbit, we note $\widehat{f}_{orb}$ the estimated

function, i.e.:

$$\widehat{f}_{orb} = \widehat{f}_{\theta^*} \text{ with } \theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, [\mathbf{x}^{orb}]). \tag{3.8}$$

We get $R^2 = 0.9998$, an overall performance consistent with L63. Evidences suggest that the emulated trajectories correctly reproduce the dynamics of the eL63 system (Figure S5).

The resulting function $\widehat{f}_{orb}$ is validated as follows. First, we take an initial condition within the original learning sample $[\mathbf{x}^{orb}]$. Then, we generate the ANN-based trajectories starting from these points over 1000 MTUs. This seems long enough for the model to manifest numerical instabilities. Integration of predicted tendencies is performed with the same time-step of $\Delta t = 0.05$, and by using the same fourth-order Runge-Kutta time stepping scheme. Stability of the resulting orbit is assessed using a 'stability criterion' defined as follows: for each $i = 1..d$, we compute the minimum (m_i) and maximum (M_i) values of x_i over the training orbit $[\mathbf{x}^{orb}]$. A N-step trajectory is considered as stable if it remains within 7 times this range of values, i.e.:

$$m_i - 3(M_i - m_i) \leq x_i^n \leq M_i + 3(M_i - m_i),\ \forall\ 1 \leq i \leq d,\ \forall\ 1 \leq n \leq N. \tag{3.9}$$

The choice of this stability criterion is partly arbitrary, but it is motivated by its simplicity. The results presented here do not depend on the choice of the stability criterion, since unstable trajectories are typically explosive (see Figure S4).

The above described validation strategy is iterated over 100 different $\widehat{f}_{orb}$, each being trained with a different rotation matrix P and a corresponding learning sample $\left[\mathbf{x}^{orb}\right]$. The resulting functions $\widehat{f}_{orb}$ are then tested by generating orbits of length 1000 MTU from 30 different initial conditions sampled randomly from their respective training orbit. As a result, stability can be assessed over 3000 simulated orbits of length 1000 MTU each.

Figure 3.3 shows the percentage of stable trajectories generated with $\widehat{f}_{orb}$ for different values of d, the dimension of eL63. Even with minimal embedding (i.e., $d = 4$), 60% of the NN-generated orbits are unstable. With $d \geq 7$, all generated orbits are unstable regarding the stability criterion defined by Eq. (3.9). The accumulation of small prediction errors gradually leads the NN-generated orbit away from the learning sample, in a region where $\widehat{f}_{orb}$ is not accurate. Hence, many NN-generated orbits are unstable, proving that eL63 is a very simple model (as its dimension remains very low) which is able to reproduce instability issues. Lastly, we notice that the same recipe (i.e., embedding) can be applied to even simpler non-chaotic dynamical system, and leads to similarly unstable trajectories when learnt by NNs. A simple circle, embedded in $\mathbb{R}^3$, seems sufficient to challenge the stability of the NN model (see Figs. S1-S3).

3.7 Stabilizing the NN-based embedded Lorenz'63 model

We now propose to illustrate a possible method to solve instability issues encountered by simple NN models in the case of the eL63 model. We generate a new learning sample by taking points away from a typical orbit, taking advantage of the fact that the value of f can be sampled at any location. In this way, we better approximate the function f on regions away from the eL63 attractors.

Here, we use a Latin Hypercube Sampling (LHS) (McKay, 1992) to generate a new learning sample of size $N = 10^4$ (i.e., N is kept unchanged). LHS generates an optimal near-random sample in a high-dimensional (here, dimension d) hypercube. The boundaries of the LHS are set to 1.5 times the range of an orbit $[\mathbf{x}^{orb}]$. This calculation is done for each $i = \{1, ..., d\}$, leading to an hypercube in the basis $\mathcal{B}_x$. Finally, we generate the target variables by simply applying f to the selected points, and obtain a new learning sample $[\mathbf{x}^{\text{LHS}}]$. The next step is to estimate f from $[\mathbf{x}^{\text{LHS}}]$. Consistent with the previous section, train and test datasets are obtained by randomly partitioning the learning sample in proportions of 80% and 20%, respectively. The same, eight-layers deep feedforward NN is fitted to the LHS data in the same configuration than described in Section 3.5. We denote

$$\widehat{f}_{\text{LHS}} = \widehat{f}_{\theta^*} \text{ with } \theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta, [\mathbf{x}^{\text{LHS}}]). \tag{3.10}$$

the new estimate of f. Learning over the LHS sample is slightly less efficient than learning from a single orbit dataset, as we find $R^2 = 0.9975$. This is an expected outcome, as the region covered by the learning sample is wider in the case of $[\mathbf{x}^{\text{LHS}}]$, resulting in more complex variations of f.

100 $\widehat{f}_{\text{LHS}}$ are trained over different learning datasets. The estimate functions then generate orbits of length 1000 MTU from 30 initial conditions that has been randomly sampled from $\left[\mathbf{x}^{orb}\right]$. The same stability criterion (Eq. 3.9) is applied to determine whether the $\widehat{f}_{\text{LHS}}$ orbit is stable. Contrary to orbits generated with $\widehat{f}_{orb}$, all orbits generated with $\widehat{f}_{\text{LHS}}$ remain stable over 1000 MTUs (see Figure 3.3) with $d = 4, \ldots, 10$. This result suggests that stability can be a property of the learning sample – and not only of the type of learning algorithm or of intense tuning of hyperparameters. These results are consistent with those shown in Scher and Messori (2019) on L63 model : dynamics of the system have been learnt unaccurately by a NN when data around one of the attractors or from one 'wing' of the 'butterfly' were removed from the training dataset. It also shows that extending the training beyond the thin phase space region explored by a single trajectory is important to improve the long-term performance (in particular stability skill) of a NN-based dynamical model.

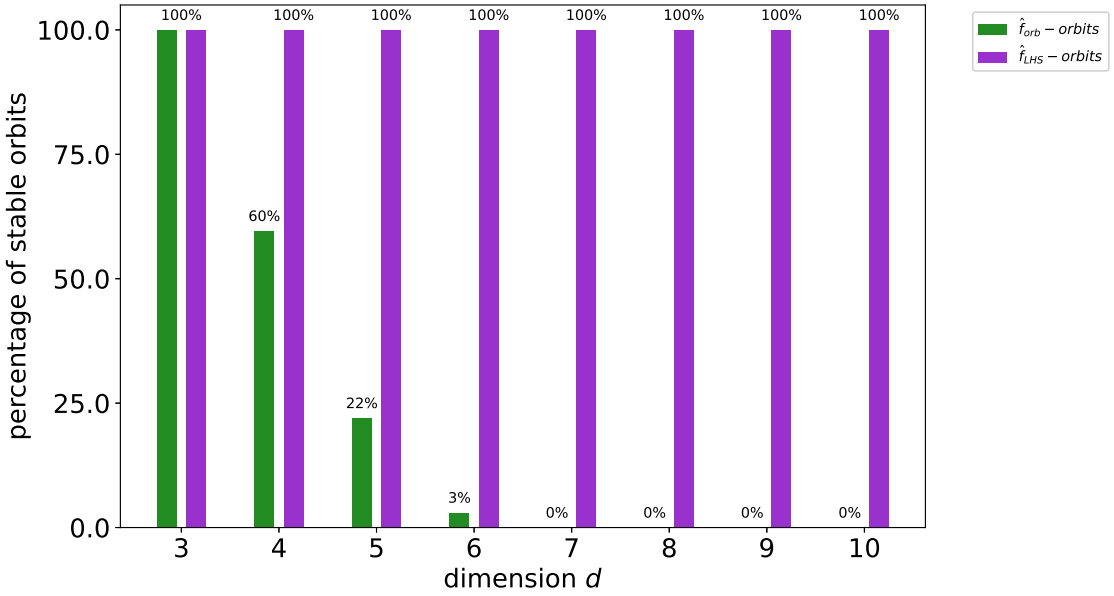


Figure 3.3: *Percentage of stable orbits generated with $\widehat{f}$ when trained with orbital ($\widehat{f}_{orb}$, green) or LHS ($\widehat{f}_{\mathrm{LHS}}$, purple) learning samples, as a function of embedding dimension d. Initial conditions are randomly sampled either in an eL63 orbit or in a region around the eL63 attractor. Stability is assessed with* 100 *different $\widehat{f}$ and* 30 *initial conditions for $d \in \{3, 4, 5, 6, 7, 8, 9, 10\}$, using the stability criterion defined in Eq. (3.9).*

However, one caveat is worth mentioning. Even though the generated trajectories are stable, in some cases, they collapse onto a fixed point near the center of one of the eL63 attractors (see Figure S5). This seems to happen more frequently as the embedding dimension d increases. It can suggest that the size of our learning sample (i.e., 10^4 individuals) may not be sufficient when the embedding dimension increases. Further investigation of the issue revealed that increasing the size of the LHS learning sample from 10^4 to 10^5 individuals reduced the number of fixed points from 15% to less than 1% (see Figure S6). This finding could motivate further research on how to improve the design of the learning sample in such problems.

3.8 Conclusion and discussion

We have developed an extended version of the Lorenz'63 model by embedding this model into higher dimension d. This new model is called the embedded Lorenz'63 model. Using a real trajectory (orbit) as a learning sample, simple artificial neural network can successfully learn the time derivatives $\dot{\mathbf{x}}$ of either L63 or eL63 models. However, unlike in L63, long trajectories generated by NNs are unstable in eL63. Instability is observed even with a minimal dimension increase, i.e., $d = 4$, and becomes more frequent as d increases. As a result, eL63 is a first example of low-dimension toy model for atmospheric dynamics that can be used to investigate the stability of NN-generated trajectories. Introduction of this new model is important,

because previous attempts to construct a toy model able to replicate instability issues have proven ineffective.

Using NNs of similar complexity but with a different learning sample, specifically designed using a latin hypercube sample, solves the instability issue. We interpret this result as follows. A typical eL63 orbit converges towards the model's attractor very quickly, leading to degeneracy – the orbit stays confined into a subspace of dimension 3. Conversely, a LHS generates points in the full eL63 space of dimension d, which allows the NN to learn the restoring force playing away from the attractor. This result is important, as it suggests that the design of the learning sample can largely influence the stability of the NN-generated trajectories. This finding also suggests that the design of the learning sample might be an important and potentially overlooked step in IA-based atmospheric modelling. As opposed to this, much literature to date has focused on improving the learning technique in order to ensure stability.

An important further question is whether or not this new toy model can provide helpful guidance for real-world problems, i.e., developing full-complexity atmospheric parameterizations based on NNs. In our view, the response is unclear at this stage, as a few questions remain open. Is the real world as much degenerated as is eL63? The fact that any orbit gets confined into a low-dimension subspace can mimic unknown physical laws. There is a growing literature on this topic. Recent studies show that NNs cannot learn exact physical laws, typically inducing drifts in the generated trajectories (e.g. Greydanus et al., 2019) Various learning techniques have been proposed to account for such physical laws. However, the degeneracy could also result from a fast equilibrium in response to restoring forces, just as assumed in eL63 – an issue which could be more difficult to address, as physical knowledge is probably more difficult to incorporate in this case. So, if some degeneracy is likely to happen in the real-world, its strength is not well determined.

Could real-world application benefit from using designed learning samples rather than 'samples of opportunity' like single orbits? This technique first requires ability to calculate the value of f at any given location. Additionally, using a LHS strategy as suggested above can be computationally demanding, especially if the dimension of $\mathbf{x}$ is large. In particular, we note that real atmospheric parameterizations involve a dimension d much higher than those investigated with our toy model.

Data availability statement

Code is made available at : `https://doi.org/10.5281/zenodo.4331710`.

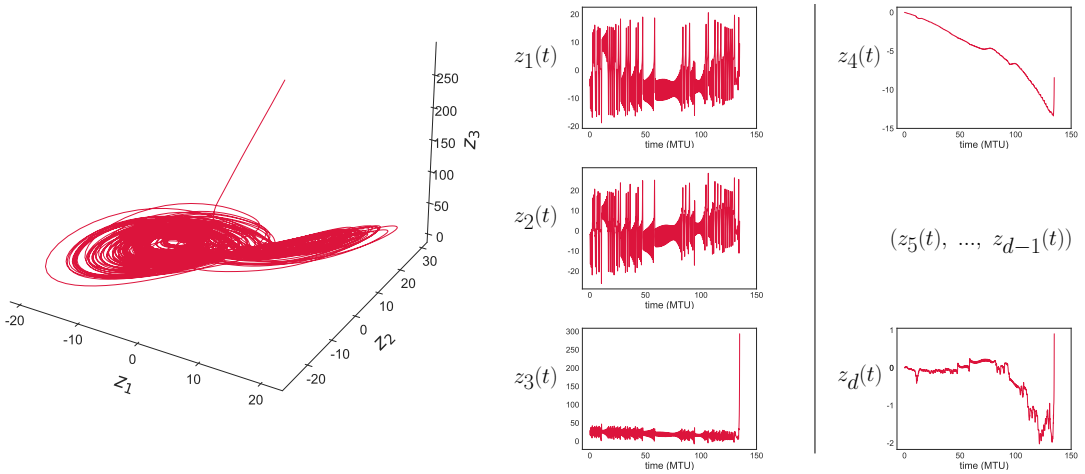


Figure 3.4: *Exemple d'orbite instable obtenu avec le modèle eL63, représentée dans $\mathcal{B}_z$. Les petites erreurs de prévision s'accumulent, jusqu'à l'explosion du modèle après 120 MTU.*

3.9 Un problème *out-of-sample*

Dans cette section, nous avons vu comment construire un modèle permettant de reproduire les problèmes d'instabilité rencontrés lorsque l'on développe des paramétrisations avec des NN. Ce modèle, noté eL63, est construit à partir de L63, en plongeant l'attracteur tridimensionnel de L63 dans un espace de dimension $d > 3$.

L'instabilité observée peut s'expliquer par un échantillon d'apprentissage *dégénéré*. Lorsque l'on apprend la physique depuis une série temporelle réalisée en intégrant les équations de eL63 (également appelée orbite), l'échantillon d'apprentissage ne décrit en réalité que l'espace situé dans des disques autour des attracteurs de L63. Autrement dit, cet échantillon d'apprentissage ne permet d'échantillonner qu'un espace localement bidimensionnel d'étendue (trop) restreinte. Dès lors, le NN ne parvient pas à extrapoler la relation apprise en dehors des plans autour des deux attracteurs. Les petites erreurs de prévision en dehors du plan décrit par l'échantillon d'apprentissage vont s'amplifier et causer l'explosion du modèle (e.g., Figure 3.4).

Pour résoudre ce problème, nous avons donc proposé de concevoir un deuxième échantillon d'apprentissage. Plutôt que d'utiliser des orbites, on réalise l'échantillonnage de la variable d'état $\mathbf{x}$ du modèle eL63, en utilisant des hypercubes latins dans un compact englobant les attracteurs (Figure 3.5). On applique ensuite les équations du modèle pour créer l'échantillon d'apprentissage. Lorsque le NN est optimisé sur ce deuxième échantillon d'apprentissage, on constate que le modèle où la "physique apprise" a été branchée est stable, au moins jusqu'à une dimension $d = 10$.

Dans le cas des modèles de climat, un échantillon d'apprentissage créé à partir d'une série temporelle peut être dégénéré, tout comme dans eL63. L'utilisation de

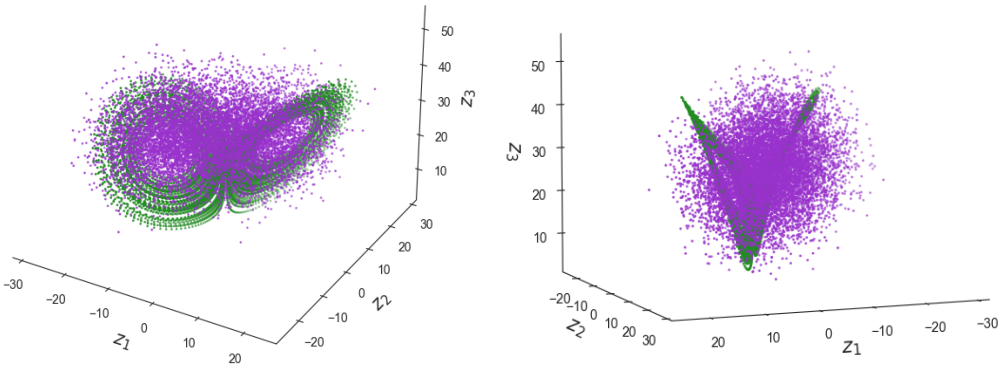


Figure 3.5: *Les variables $\boldsymbol{x}$ dans les deux échantillons d'apprentissage utilisées dans l'article. Seules les trois premières dimensions sont représentées (dans le plan $\mathcal{B}_z$ selon les notations de l'article). (Vert) l'échantillon d'apprentissage est composé d'une orbite, dont la structure est localement quasi-bidimensionnelle. Cet échantillon d'apprentissage est dégénéré et conduit à une physique NN instable une fois que celle-ci est branchée dans le modèle dynamique. (Violet) le deuxième échantillon d'apprentissage est créé en échantillonnant l'espace autour des attracteurs de L63, en utilisant des hypercubes latins (dans l'espace $\mathcal{B}_x$) pour échantillonner la variable d'état d'eL63. Lorsque le NN apprend sur cet échantillon, la physique apprise est stable une fois branchée dans le modèle dynamique : la dégénérescence a pu être levée.*

la méthode développée sur eL63 pourrait donc être utilisée dans le cas de modèles de climat pour rendre les physiques NN stables. Néanmoins, il reste encore quelques difficultés à résoudre avant de pouvoir appliquer cette méthode. Avec les modèles numériques (surtout aussi simples que eL63), il est aisé de récolter des données là où nous en manquons. Il suffit de réaliser une simulation à l'endroit où nous avons besoin de plus de données pour ajuster le NN. Dans le cas d'un modèle de climat, cela devient plus délicat. Tout d'abord, parce que la dimension totale des variables d'état (analogue à la dimension d dans eL63) est assez grande. Dans sa configuration opérationnelle, CNRM-CM6 comporte 91 niveaux verticaux pour les variables 3D. Les variables en entrée de la physique NN seront typiquement quelques profils verticaux et variables scalaires, ce qui revient à $d \approx 300$, une valeur bien plus élevée que ce que nous avons pu essayer dans eL63.

Un autre problème potentiel vient de la manière dont on réalise l'échantillonnage dans le cas d'un modèle de climat. Dans le cas eL63, les variables $\mathbf{x}$ de l'échantillon d'apprentissage ont été tirées aléatoirement, en utilisant des hypercubes latins dans un espace englobant les attracteurs. Cela n'est pas aussi simple dans le cas d'un modèle de climat où les *variables* sont non seulement liées entre elles, mais également *auto-corrélées* dans le cas des profils verticaux. Ainsi, la méthode d'échantillonnage pour générer l'échantillon d'apprentissage présentée dans la publication nécessiterait des ajustements pour être utilisée dans le cadre d'un modèle de climat. Elle peut par exemple être adaptée en échantillonnant les conditions initiales de manière aléatoire et en réalisant des simulations de quelques pas de temps. Mais la réalisation d'une

telle simulation a un coût numérique élevé, et nécessite donc une préparation minutieuse. Comment échantillonner les conditions initiales de manière efficace ? Sur quels domaines ? Quelle durée de la simulation ?

Chapitre 4

Calibration

4.1 Avant-propos

Comme nous avons vu en introduction, contrairement aux modèles de prévision numérique du temps, les modèles de climat ont pour but de reproduire des *distributions* plutôt qu'une chronologie. La performance d'un modèle de climat est donc davantage mesurée par son aptitude à reproduire les caractéristiques statistiques des distributions des variables décrivant l'atmosphère (e.g., température, humidité, précipitations). Ces caractéristiques statistiques correspondent à la *climatologie* du modèle. Le développement et l'optimisation des paramétrisations physiques s'effectue en deux temps. La première étape consiste à utiliser les données observées, connaissances théoriques et sorties de modèles haute-résolution pour ajuster la paramétrisation *offline*, à l'aide de jeux de données. La deuxième étape consiste à brancher cette physique dans le modèle de climat, et à *calibrer* la valeur des paramètres disponibles à cette fin pour régler d'éventuels problèmes de stabilité, et surtout pour ajuster *online* les métriques de long terme (étape de *tuning*), comme le biais ou la variance. Ces deux étapes sont répétées de manière itérative jusqu'à obtenir une calibration satisfaisante. La qualité de chaque jeu de paramètres est évaluée grâce à des métriques, qui, typiquement, quantifient les écarts du modèle par rapport au climatologie observé.

L'étape de *tuning* fait partie intégrante du développement des paramétrisations physiques (e.g., Hourdin et al., 2017). La performance *offline*, évaluée sur quelques cas d'étude bien connus et documentés, ne se traduit pas nécessairement par de bons résultats en termes de simulations de climatologie. Toutefois, les paramétrisations NN actuellement développées ne permettent pas d'effectuer l'étape de *tuning*. Elles sont ajustées *offline* uniquement, *via* l'application des équations de rétropropagation durant l'apprentissage (Chapitre 1). Cela traduit une difficulté technique. Optimiser *online* la valeur de quelques paramètres d'un schéma physique est réalisable. Mais optimiser *online* la valeur de quelques centaines de milliers de poids et biais caractérisant une paramétrisation NN est sans doute impossible. Dès lors, il n'y a aucune garantie qu'une physique NN performante sur un échantillon d'apprentissage (*offline*) le soit également une fois branchée dans le modèle de climat (*online*). Si l'échantillon d'apprentissage comporte des erreurs, celles-ci seront également apprises, et dérègleront le modèle de climat à plus ou moins long terme. De plus, il est difficile d'évaluer *offline* si le découpage entre physique et dynamique a bien été effectué. Ce problème est communément réglé par l'étape de *tuning* dans le cas des paramétrisations physiques. Cette problématique rejoint celle de la stabilité. La variété des situations *offline* pour effectuer des tests n'est pas suffisante pour garantir la stabilité et la performance *online*. Certains schémas physiques peuvent se révéler instables numériquement.

La structure des NN rend impossible une calibration *online* des paramètres du NN (i.e., poids et biais). Étant donnée l'importance de la calibration dans le développement des paramétrisations, nous avons développé une approche perme-

ttant d'ajuster *online* les paramétrisations NN. La suite de ce chapitre est la retranscription d'un article présentant la méthode, publié dans *Geophysical Research Letters*.

Balogh, B., Saint-Martin, D. & Ribes, A. 2022.
Article publié dans Geophysical Research Letters.

4.2 Abstract

Unlike the traditional subgrid scale parameterizations used in climate models, current machine learning (ML) parameterizations are only tuned *offline*, by minimizing a loss function on outputs from high-resolution models. This approach often leads to numerical instabilities and long-term biases. Here, we propose a method to design tunable ML parameterizations and calibrate them *online*. The calibration of the ML parameterization is achieved in two steps. First, some model parameters are included within the ML model input. This ML model is fitted at once for a range of values of the parameters, using an *offline* metric. Second, once the ML parameterization has been plugged into the climate model, the parameters included among the ML inputs are optimized with respect to an *online* metric quantifying errors on long-term statistics. We illustrate our method with two simple dynamical systems. Our approach significantly reduces long-term biases of the ML model.

4.3 Plain Language Summary

In numerical climate models, processes occurring at scales smaller than the model resolution (e.g., convection, turbulence) need to be represented by 'parameterizations'. Parameterizations provide a simplified yet numerically affordable version of the modeled processes. Recently, parameterizations are also developed using machine learning (ML) by fitting to outputs from high-resolution climate models. This method can lead to long-term biases when incorporating the ML parameterizations into the climate model. And, there is no possibility in the current approach to calibrate the ML parameterization to alleviate these biases.

We propose here an innovative approach to calibrate ML parameterizations once they have been fitted to a learning sample. Our approach has been successfully tested on two toy models. A first set of experiments focus on the retrieval of the value of parameters used to generate a reference dataset. In the second experiment, the value of some parameters not included in the NN has been biased, resulting in errors in long-term statistics. Finding the optimal value of the NN input parameter has significantly improved the accuracy of the resulting model. Our method could be applied to improve the prediction of long-term variables in climate models.

4.4 Introduction

The ordinary differential equations describing a climate model, can be written

$$\dot{\mathbf{x}} = \mathcal{D}(\mathbf{x}) + \phi(\mathbf{x}), \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the state variable of the climate model. $\mathcal{D}$ corresponds to the discretization of the primitive equations on the model grid, whereas ϕ accounts for the subgrid scale atmospheric processes (e.g., turbulence, convection). Currently, the best compromise for representing subgrid scale processes are the 'physical parameterizations', emulating a numerically affordable version of ϕ. These parameterizations are parametric functions f_ϕ. They are built heuristically on theoretical knowledge and outputs from high-resolution simulations (e.g., Jakob, 2010). In practice, f_ϕ is the sum of individually modelled processes, the parameters of which are separately estimated. The numerical efficiency of f_ϕ comes at the price of approximations. Hence, parameterizations of the atmosphere are the main source of uncertainties in climate models (e.g., Medeiros et al., 2008; Medeiros and Stevens, 2011; Stevens and Bony, 2013). Recently, interest has grown in using Machine Learning (ML) to develop parameterizations, noted $\widehat{f}$, promising more precise yet affordable parameterizations for climate models (e.g., Rasp et al., 2018; Gentine et al., 2018; Yuval and O'Gorman, 2020).

The development of both physical and ML-based parameterizations first involves defining a set of possible functions for the estimate of ϕ (this set is typically much larger in the ML case). Then, model parameters are optimized with respect to an *offline* metric ℓ (also called 'loss function' in the ML case). This *offline* metric evaluates the error between the estimate function (i.e., f_ϕ or $\widehat{f}$) and a target function f, the best accurate approximation of ϕ – as even high-resolution model data is only an approximation of the ground truth subgrid-scale processes. This approach allows us to fit the models on a *point by point* basis. However, a climate model's performance lies in accurate predictions of long-term statistics, measured by *online* metrics. Physical parameterizations typically require an additional calibration step, once it has been plugged-in to the dynamical model. In contrast, there is usually no tunable parameter included in ML parameterizations. Nevertheless, tuning is very important for the development of climate models, to ensure model stability, to calibrate the value of long-term statistics and to reduce error due to potentially missed interactions between the dynamics and the physics. This *online* calibration can be achieved regarding an *online* metric, m. Without *online* calibration, even though a parameterization is efficient regarding the *offline* metric ℓ, there is no guarantee that the climate model will be accurate regarding the *online* evaluation metric (Brenowitz et al., 2020b).

The development of current ML parameterizations is still hampered by issues addressed by *online* calibration in the case of physical parameterizations, such as numerical instabilities (e.g., Brenowitz et al., 2020a; Rasp, 2020). These issues are currently handled by enforcing physical conservation laws (Beucler et al., 2021a) or using ML to replace part of a physical parameterization (Yuval et al., 2021). Additionally, the target function f can also be imperfect. In this case, even though the fit of the ML model is excellent, errors in the imperfect training dataset will also be learned and result in a high *online* error m, also called long-term biases. This

is a well known issue with physical parameterizations. The 'art of tuning' model parameters consists in finding a compromise between *offline* and *online* metrics, going back and forth in optimizing the parameters with respect to ℓ and m (Hourdin et al., 2017; Schmidt et al., 2017; Couvreux et al., 2021).

To resolve these issues, we propose to include some tunable parameters into the ML parameterization inputs. This allows the calibration of new ML-based parameterization with respect to an *online* metric in an efficient way. Our approach to find the best value of tunable parameters is less empirical than the methods used to tune physical parameterizations. It relies on the minimization of the *online* metric m on long-term statistics, akin to the methodology described in Schneider et al. (2017) and in Cleary et al. (2021). The method will be demonstrated using Lorenz'63 and Lorenz'96 toy models (Lorenz, 1963, 1996). These toy models offer a simple framework to perform proof-of-concept atmospheric modelling experiments using ML tools (e.g., Scher and Messori, 2019; Chattopadhyay et al., 2020). Lorenz'96 model has also the advantage of implementing a 'parameterization'. The large scale variable is indeed interacting with the non-linear subgrid-scale variable.

The paper is organised as follows. Methodology is described in Section 4.5. Section 4.6 demonstrates our calibration method on two basic examples using Lorenz'63 and Lorenz'96 models. Conclusions are drawn in Section 4.7, discussing more broadly the results we have obtained.

4.5 Methodology

4.5.1 Step 1 : building a tunable neural network parameterization

The first step towards designing and optimizing tunable neural network (NN) parameterizations is to take into consideration some uncertain but tunable parameters, when fitting the NN. The target function f is often fitted to outputs from high-resolution simulations. This target function depends on uncertain parameters, noted $\boldsymbol{\theta} \in \mathbb{R}^p$, used to generate the high-resolution simulations : $f \equiv f(\mathbf{x}; \boldsymbol{\theta})$. In traditional approaches, the value of model parameters are fixed to the 'best estimate' value of these parameters, $\boldsymbol{\theta}_0$. Thus, $f \equiv f(\mathbf{x}; \boldsymbol{\theta}_0)$ does not depend on any parameter and $\boldsymbol{\theta}$ is not included in the NN model input (e.g., Gentine et al., 2018; Yuval and O'Gorman, 2020). The novelty of our approach is to keep part of the uncertain parameters $\boldsymbol{\theta}$ among the input variables of the NN, and thus to retain not only the dependency of f from $\mathbf{x}$ but also from some model parameters $\boldsymbol{\theta}$.

The NN learning sample, of size N, is $\{(\mathbf{x}, \boldsymbol{\theta})_i, f(\mathbf{x}_i; \boldsymbol{\theta}_i)\}_{1 \leq i \leq N}$. The NN is fitted by optimizing an *offline* metric ℓ, also called 'loss function'. Typically, this loss function is Mean Squared Error (MSE), computed for each predicted time step

individually:

$$\ell\big(f(\mathbf{x};\boldsymbol{\theta}),\widehat{f}(\mathbf{x};\boldsymbol{\theta})\big) = \frac{1}{N}\sum_{i=1}^{N}\left\|f(\mathbf{x}_i;\boldsymbol{\theta}_i)-\widehat{f}(\mathbf{x}_i;\boldsymbol{\theta}_i)\right\|^2. \tag{4.2}$$

The parameterization obtained after training the NN is noted, $\widehat{f}(\mathbf{x};\boldsymbol{\theta})$. It can be incorporated into the dynamical model so as to replace ϕ in Eq. 4.1. The resulting dynamical system can be used to generate a validation time series of the dynamical model, noted $[\mathbf{x}]^{\widehat{f}}$.

4.5.2 Step 2 : optimizing the tunable neural network parameterization

The goal of this second step is to tune the optimal value of the parameters $\boldsymbol{\theta}$. Whereas the NN model was trained *offline* on a learning sample, calibration relies on an *online* validation metric, m, given by:

$$m(\boldsymbol{\theta}) = \frac{1}{M}\sum_{k=1}^{M}\left\|\mathcal{M}_k^{\text{ref}}-\mathcal{M}_k(\boldsymbol{\theta})\right\|^2, \tag{4.3}$$

where $\mathcal{M}$ are a set of M long-term statistics computed over $[\mathbf{x}]^{\widehat{f}}$, and $\mathcal{M}^{\text{ref}}$ a set of reference statistics. The long-term statistics involved in the computation of m are typically the average and standard deviation values estimated over time series. The minimal value of m is reached at the optimal value of $\boldsymbol{\theta}$, noted $\boldsymbol{\theta}^*$.

Theoretically, the statistics $\mathcal{M}$ are computed over a time series of infinite length. In reality, we only compute an estimate of the long-term metric over a finite length simulation. The length of these simulation is chosen so that the *online* metric m no longer depends on the $\mathbf{x}$ initial condition. To simplify notations, the estimate of the long-term statistics over validation time series $[\mathbf{x}]$ is noted, $\widehat{\mathcal{M}}$. Even though this time series is sufficiently long, the resulting metric $\widehat{m}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{k=1}^{M}\left\|\mathcal{M}_k^{\text{ref}}-\widehat{\mathcal{M}}_k(\boldsymbol{\theta})\right\|^2$ will be noised, which can lead the optimizer to a local minimum of $\widehat{m}$, instead of the global minimum. To address this issue, minimization will be performed over a smoothed version of $\widehat{m}$, obtained by kriging (or Gaussian Process Regression, Cressie, 1992). The kriging metamodel will be fitted to a sample of $\{\boldsymbol{\theta}_i,\widehat{m}(\boldsymbol{\theta}_i)\}$. The obtained kriging metamodel is noted $\widetilde{m}$. The optimal value of model parameters, $\boldsymbol{\theta}^*$, is obtained when $\widetilde{m}$ reaches its (absolute) minimum value.

In summary, we train a NN parameterization depending on some tunable parameters $\boldsymbol{\theta}$, using an *offline* metric, ℓ. Since the main purpose is to reduce long-term prediction errors of the dynamical model, the value of $\boldsymbol{\theta}$ is subsequently optimized regarding an *online* metric, m. In practice, optimization is done over a kriging metamodel emulating m as a function of $\boldsymbol{\theta}$. In the following, our method will be

demonstrated using the Lorenz'63 (hereafter L63, Lorenz, 1963) and the Lorenz'96 (hereafter L96, Lorenz, 1996) models.

4.5.3 The Lorenz'63 model

The L63 system consists of a set of ordinary differential equations that can be expressed:

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= x_1(\rho - x_3) - x_2, \\ \dot{x}_3 &= x_1 x_2 - \beta x_3. \end{aligned} \tag{4.4}$$

Temporal evolution of the L63 state variable, $\mathbf{x} = (x_1, x_2, x_3)$, is governed by Eq. 4.4, which involves a set of three model parameters, (σ, ρ, β). Eq. 4.4 admits a chaotic solution in the vicinity of $(\sigma_0, \rho_0, \beta_0) = (10, 28, 8/3)$.

The *online* minimization metric m (Eq. 4.3) depends strongly on the identification of variables quantifying the long-term statistical behavior of the L63 system. The first candidate of such a long-term variable is the mean value for each one of the three L63 state variables. However, the average value of x_1 and x_2 is independent of L63 parameters (see also Supporting Information, Figure S1). Thus, only the average value of x_3, noted $\mu^{(3)}$, can be tuned. This quantity is useful to retrieve the optimal value of at most one L63 parameter. In addition to $\mu^{(3)}$, standard deviations over the time series can also be computed. Standard deviations will be noted $\sigma^{(i)}$ for each one of the L63 parameters, with $i \in \{1, 2, 3\}$. However, as highlighted by Figure S1, a strong correlation appears between the three standard deviations. Hence, it is possible to retrieve the optimal value of at most two L63 parameters by optimization. Thus, the following L63 idealized case study will use $\mathcal{M} = (\mu^{(3)}, \sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)})$.

4.5.4 The Lorenz'96 model

The 2-level L96 dynamical system depending on parameters (h, c, F, b) is given by:

$$\frac{\mathrm{d}x_k}{\mathrm{d}t} = -x_{k-1}(x_{k-2} - x_{k+1}) - x_k + F - \frac{hc}{b}\sum_{j=1}^{J} y_{k,j}, \tag{4.5}$$

$$\frac{1}{c}\frac{\mathrm{d}y_{k,j}}{\mathrm{d}t} = -b y_{k,j+1}(y_{k,j+2} - y_{k,j-1}) - y_{k,j} + \frac{hc}{b} x_k, \tag{4.6}$$

where $\{x_k\}_{1 \le k \le K}$ are K large-scale variables coupled to J small-scale variables $\{y_{k_0,j}\}_{1 \le j \le J}$ for each $1 \le k_0 \le K$. The coupling term $B_k = -\frac{hc}{b}\sum_{k=1}^{K} y_{k,j}$ can be seen as a 'subgrid-scale parameterization'. Thus, Eq. 4.5 can be analogous to a (very) simple climate model, as described by Eq. 4.1. In this comparison, $\mathcal{D}(x_k) = -x_{k-1}(x_{k-2} - x_{k+1}) - x_k + F$ and $\phi(x_k) = B_k$. B_k can be approximated as a function of the large-scale state variables, x_k, and is often modelled with polynoms (e.g., Arnold et al., 2013). This subgrid-scale parameterization will be the target

variable of the NN model. Given the symmetry of the L96 model, it is common to fit the NN to predict B_k as a function of data from only one spatial variable (e.g., Watson, 2019; Gagne et al., 2020; Rasp, 2020). In our case, the NN learns to predict a single B_k as a function of (x_k, c). This implies that the NN input is of size 2 and the output of size 1. To simplify notations, in the following, the NN will be noted, $\widehat{B}(x, c)$.

As in the L63 case, the mean value of the state variables does not depend on input parameters (h, c, F, b), and is therefore not used in the *online* metric m. Thus, the metric is based on standard deviations only, estimated on time series obtained with the NN parameterization.

4.6 Case studies

4.6.1 Perfect model calibration

The Lorenz'63 model

The objective is to fit an NN model, noted $\widehat{f}$, to approximate the L63 time derivative as a function of the state variable $\mathbf{x}$ and parameters $\boldsymbol{\theta} = (\rho, \beta)$:

$$\widehat{\dot{\mathbf{x}}} = \widehat{f}(\mathbf{x}, \boldsymbol{\theta}). \tag{4.7}$$

The learning sample is generated by an optimal sampling method, used to select relevant $(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}^3 \times \mathbb{R}^2$ values to build the NN learning sample. Latin Hypercube Sampling (hereafter LHS, McKay, 1992) is such a sampling method. The interest of using a specific sampling method to train stable and accurate NN based parameterization has been shown in Balogh et al. (2021). The boundaries of the LHS are set around plausible values for both $\mathbf{x}$ and $\boldsymbol{\theta}$. The resulting learning sample of size N_{LHS} is $[(\mathbf{x}, \boldsymbol{\theta})]^{\text{LHS}} = \{(\mathbf{x}, \beta)_i, f(\mathbf{x}_i; \boldsymbol{\theta}_i)\}_{1 \leq i \leq N_{\text{LHS}}}$.

The target variable $\dot{\mathbf{x}}$ is computed with L63 model equations, parameterized with $\sigma = \sigma_0$. The learning sample is of size $N_{\text{LHS}} = 10^7$. Values for $\boldsymbol{\theta} = (\rho, \beta)$ are sampled in $[26.5, 32] \times [1.5, 3.2]$. The NN model consists of $n_l = 7$ hidden layers of type 'Dense'. More specific details about the NN architecture are available in the Supporting Information, Table S1. R^2 score over an independent subset of 20% of the learning sample is monitored during training. The best weights regarding the R^2 score are loaded after 30 epochs. The final model has $R^2 = 1.00$, which is not surprising given the low complexity of L63 model, and is consistent with case studies focusing on emulating L63 dynamics with NNs (Rasp, 2020).

The fitted NN model is then used to generate time derivatives, which are integrated with a Runge-Kutta 4 time stepping scheme with a temporal increment $\Delta t = 0.05$. The resulting validation time series or 'orbit' $[\mathbf{x}]^{\widehat{f}}$ is of length 1000 Model Time Units (hereafter, MTU, where 1 MTU = $20\Delta t$) not including a spin-up

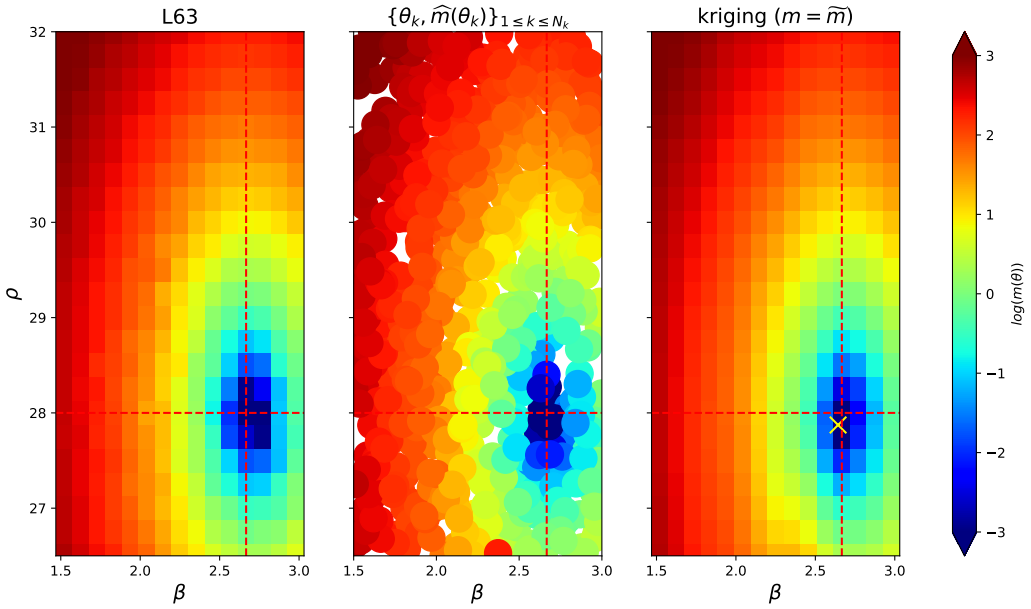


Figure 4.1: *(Left) The online metric m is computed over orbits of length 1000 MTUs generated with real L63 equations (Eq. 4.4) for $\sigma = \sigma_0$ and for different regularly-spaced values of $\boldsymbol{\theta} = (\rho, \beta)$. (Middle) Values of $\widehat{m}$ in the kriging learning sample, i.e., $\{\boldsymbol{\theta}_i, \widehat{m}(\boldsymbol{\theta}_i)\}_{1 \le i \le N_k}$. The long-term statistics, $\widehat{m}(\boldsymbol{\theta}_i)$ are computed on validation orbits $[\boldsymbol{x}]^{\widehat{f}}$ of length 1000 MTUs. (Right) The fitted kriging metamodel $\widetilde{m}$ used to find the optimal value of the tunable parameters, $\boldsymbol{\theta}^*$ (yellow cross). The optimal value of parameters is $\boldsymbol{\theta}^* = (27.9, 2.64)$, which is close to the values used to generate the reference dataset: $(\rho, \sigma) = (\sigma_0, \rho_0)$. Color shades represent the logarithm of the online evaluation metric m (Eq. 4.8) or its estimates.*

of 200 MTU. The validation orbits are used to compute long-term metrics and thus to assess the *online* performance of the model, measured by $\widehat{m}$:

$$\widehat{m}(\boldsymbol{\theta}) = (\widehat{\mu}^{(3)}(\boldsymbol{\theta}) - \mu_{\text{ref}}^{(3)})^2 + \sum_{n=1}^{3} (\widehat{\sigma}^{(n)}(\boldsymbol{\theta}) - \sigma_{\text{ref}}^{(n)})^2, \tag{4.8}$$

where $\mu_{\text{ref}}^{(3)}$ and $\boldsymbol{\sigma}_{\text{ref}}$ (resp. $\widehat{\mu}^{(3)}(\beta)$ and $\widehat{\boldsymbol{\sigma}}(\beta)$) are parameters computed over the reference orbit (resp. validation orbit $[\mathbf{x}]^{\widehat{f}}$). In this example, the 'reference' dataset consist in a long time-series of length 3000 MTU (considered as 'infinite' lenght), generated by integrating L63 equations (Eq. 4.4) with parameters $(\sigma_0, \rho_0, \beta_0)$.

A kriging metamodel learns to approximate $\widehat{m}$ as a function of $\boldsymbol{\theta}$, over a learning sample of size $N_k = 750$. To build the kriging learning dataset, $\boldsymbol{\theta}$ values are sampled in the interval $[26.5, 32] \times [1.5, 3]$. The sampling interval for $\boldsymbol{\theta}$ has been slightly reduced compared with those used to generate the NN learning sample to avoid potential out-of-sample issues. In practice, we use a LHS to generate a sample of $(\mathbf{x}_0, \boldsymbol{\theta})$ to compute the validation orbits of length 1000 MTU on which the long-term metrics are estimated. Including the initial state variable $\mathbf{x}_0$ in the LHS sample reduces the noise related to the (finite) length of the validation orbits from the kriging learning sample. For the reasons explained in the Methodology section, dependency of $\widehat{m}$ on $\mathbf{x}_0$ is ignored.

The kriging model $\widetilde{m}$ is fitted to $\{\boldsymbol{\theta}_i, \widehat{m}(\boldsymbol{\theta}_i)\}_{1 \leq i \leq N_k}$. The minimum of $\widetilde{m}$ is retrieved with BFGS optimizer (Fletcher, 2013). The optimal value found for the tunable parameters is $\boldsymbol{\theta}^* = (27.9, 2.64)$ (Figure 4.1). The reference orbit was generated using $(\rho_0, \beta_0) = (28, 8/3)$: $\boldsymbol{\theta}^*$ is close to the values of the parameters used to generate the reference orbit.

The Lorenz'96 model

The NN model is trained to predict $B = -\frac{hc}{b} \sum_{j=1}^{J} y_j$ as a function of x and $\boldsymbol{\theta} = c$. The learning sample is built by sequentially integrating the L96 equations (see Eqs. 4.5 and 4.6, with $K = 8$ and $J = 32$), using a 4th order Runge-Kutta time stepping scheme with an increment $\Delta t = 0.005$. The length of training integrations is 3,5 MTU (where 1 MTU = 200 Δt), not including 1,5 MTU of model spin-up. We perform $N_i = 500$ integrations, the initial conditions and $\boldsymbol{\theta}$ values $(\mathbf{x}, \mathbf{y}, c)$ of which are sampled using LHS, where $\mathbf{x} = (x_1, x_2, ..., x_K)$ and $\mathbf{y} = (y_{1,1}, y_{1,2}, ..., y_{K,J})$. The sampling interval for c values is $[6, 14]$.

The NN is two layers deep, and has 32 nodes on each. More specific details about the NN architecture are available in the Supporting Information, Table S2. The model is trained over 30 epochs using the MSE loss function (Eq. 4.2). The validation dataset is made of 15% randomly chosen samples from the learning dataset. R^2 score is monitored over this dataset during fitting. Best weights regarding the

validation R^2 score are saved and loaded after 30 epochs. The final model has $R^2 = 0.89$ and is noted, $\widehat{B}(x, c)$.

The fitted NN is then used to generate validation time series $[\mathbf{x}]^{\widehat{B}}$ of length 15 MTU, using $\widehat{B}$ instead of B in the L96 equations (Eq. 4.5). Long-term metric $\widehat{m}$ is computed over $[\mathbf{x}]^{\widehat{B}}$, using standard deviation values only:

$$\widehat{m}(c) = (\widehat{\sigma}(c) - \sigma_{\text{ref}})^2, \tag{4.9}$$

with σ_{ref} (resp. $\widehat{\sigma}(c)$) the standard deviation computed over the reference time series (resp. the validation time series). A time series of length 15 MTU, generated by integrating L96 equations (Eqs. 4.5 and 4.6) with $(h_0, F_0, b_0, c_0) = (1, 10, 10, 10)$, is considered as the 'reference' dataset.

As described in Sec. 4.5.3, a kriging metamodel, $\widetilde{m}$, is fitted to approximate $\widehat{m}$ on a sample of $\{c_i, \widehat{m}(c_i)\}_{1 \leq i \leq N_K}$, of size $N_k = 200$. The initial conditions for the orbits used to compute $\widehat{m}$ in the kriging learning sample are generated by LHS, with $c \in [7., 13.]$ (Figure 4.2). The value of c minimizing the *online* metric is retrieved on $\widetilde{m}$, by using BFGS minimization from SciPy python package. The optimal value of c is $c^* = 9.922$. This value is very close to the value used to generate the reference dataset, ie. $c_0 = 10$.

4.6.2 Imperfect model calibration: the Lorenz'96 model

We now investigate the case where one of the L96 model parameters is carrying biases. To reproduce this situation, the L96 NN parameterization has been trained on output from the L96 model using the reference value of the model parameters, i.e., $(h, F, b) = (h_0, F_0, b_0)$. However, the NN model will be implemented in an L96 system where one of the model parameters is set to a value different from its reference value. We will show that optimizing the value of the tunable parameter included in the NN still allows us to obtain the wished long-term statistics.

Namely, we set the value of F to a range of biased values, $F_b \neq F_0$. For each F_b, we generate validation time serie using the NN parameterization, $\widehat{B}(x; c)$, and compute the corresponding long-term statistics. To underline the dependence of the long-term statistics on F_b, the *online* metric will be noted:

$$\widehat{m}_{F_b}(c) = (\widehat{\sigma}_{F_b}(c) - \sigma_{\text{ref}})^2, \tag{4.10}$$

where $\widehat{\sigma}_{F_b}$ is the standard deviation computed on a validation time series where $F = F_b$.

To predict the estimated metric, $\widehat{m}_{F_b}(c)$, again, a kriging metamodel is trained over a dataset $\{c_i, \widehat{m}_{F_b}(c_i)\}_{1 \leq i \leq N_k}$ where c_i are $N_k = 200$ values sampled in the

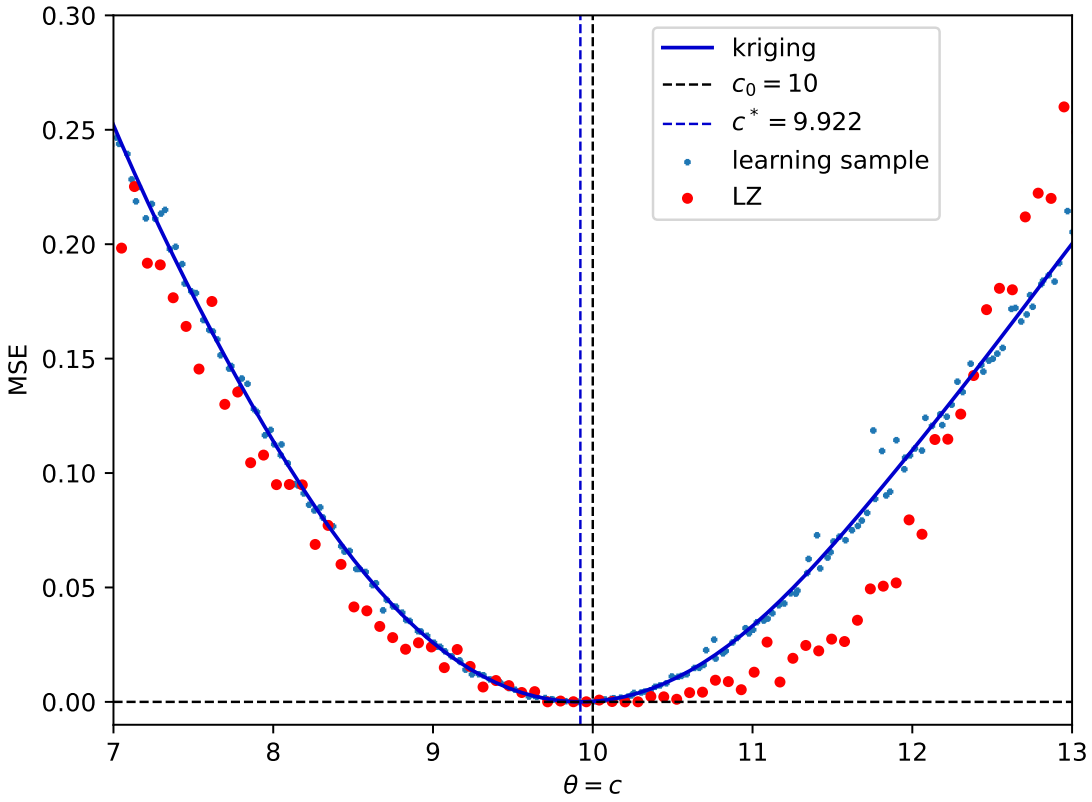


Figure 4.2: *A LHS sample of* $(\boldsymbol{x}, b)$ *values,* $\{\boldsymbol{x}_i, c_i\}_{1 \leq i \leq N_i}$*, is generated using LHS with* c *sampled in* $[7, 13]$*. The corresponding values of the long-term metrics,* $\widehat{m}(c_i), 1 \leq c_i \leq N_i$*, are computed on* $[\boldsymbol{x}]^{\widehat{B}}$ *(length : 3,5 MTU) from the LHS initial conditions. The resulting sample* $\{c_i, \widehat{m}(c_i)\}_{1 \leq i \leq N_k}$ *(light blue scatter points) is used to train the kriging model (solid blue line). For comparison, red dots represent the metric* m *computed over time series (length : 15 MTU) obtained by applying L96 equations (Eq. 4.5), for discrete values of* c*. The fitted kriging metamodel,* $\widetilde{m}$ *(solid blue line), is a smoothed version of the metric computed on time series generated with the NN parameterization. The optimal value of* c*,* c^**, is computed by minimizing* $\widetilde{m}$*.* $c^* = 9.922$ *(dashed blue line) can be compared with* $c_0 = 10$ *(dashed black line), which was used to generate the reference orbit.*

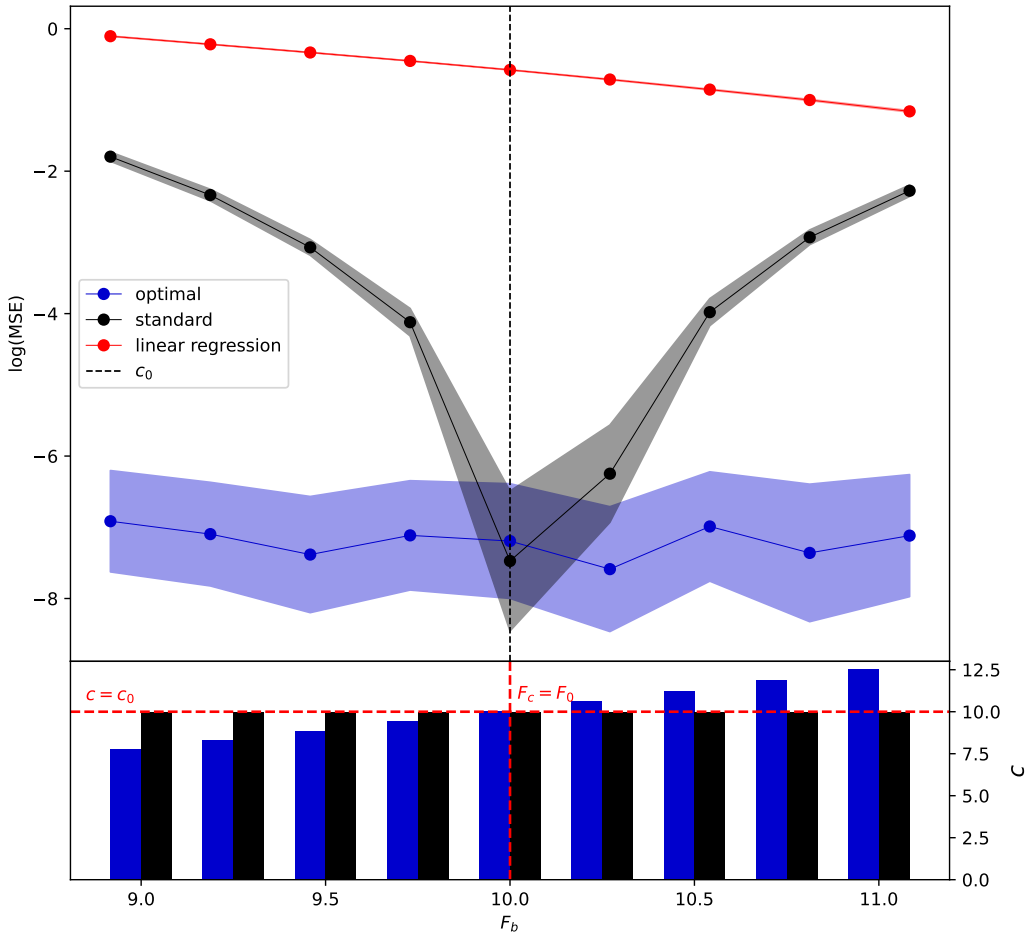


Figure 4.3: *(Black) Orbits computed with NN parameterization $\widehat{B}(x;c)$ with $c = c_0$ for each F_b in the interval $[9, 11]$ (black bars, bottom panel). The logarithm of kriged values of the metric corresponding to these time series, $\widetilde{m}_{F_b}(c_0)$, are plotted in the top panel (dotted solid black line). The 95% confidence intervals are represented in gray shading. This method can be compared with output using a linear regression approach, fitted to approximate B (red, top panel). (Blue) For each F_b, we also compute the optimal value c^* of c to minimize the online metric $\widetilde{m}_{F_b}$ (blue bars, bottom panel). The 95% confidence intervals are represented in blue shading. The minimal value of each F_b, $\widetilde{m}_{F_b}(c^*)$, associated with orbits obtained with ML parameterization $\widehat{B}(x;c^*)$ remains close to zero (dotted solid blue line, top panel).*

interval $[7, 13]$. Hence, a kriging metamodel is obtained for each F_b and the corresponding functions are noted $\widetilde{m}_{F_b}(c)$.

We now optimize the value of c with respect to the *online* metric $\widehat{m}_{F_b}$. For each F_b, we find the optimal value of c, noted $c^*(F_b)$, by minimizing the corresponding kriging metamodel, $\widetilde{m}_{F_b}$ (Figure 4.3, bottom panel). Bias compensation can be evaluated by comparing the value of $\widetilde{m}_{F_b}$ computed on validation time series generated with $\widehat{B}(x; c_0)$ and $\widehat{B}(x; c^*)$, for each value of F_b. Whereas the loss values remain high for $c = c_0$ when F is strongly biased, the use of $c^*(F_b)$ significantly reduces the *online* loss value (Figure 4.3). As soon as $F_b \neq F_0$, $c^*(F_b) \neq c_0$. This means that the choice of $c = c_0$ is not optimal. It also means that at least part of the bias induced by the wrong parameter F_b can be actually compensated by tuning another parameter of the model, c. This result suggest that, at least to some extent, even an imperfect model can be tuned to adjust long-term statistics.

Figure 4.3 clearly shows that, as soon as $F_b \neq F_0$, the proposed method is a statistically significant improvement to the baseline (linear regression) physical parameterizations and the NN parameterization with $c = c_0$. Confidence intervals associated with our estimates of the metric have been constructed by bootstrap through the sampling of two main sources of uncertainty. First, we build 9 different NN parameterizations, each of which has been fitted to a learning sample generated with different initial conditions. Second, each one of the 9 parameterizations are used to generate 3 validation orbits of length 10, 15 and 25 MTUs. Thus, a sample of $n = 27$ validation orbits is obtained and used to compute 95% confidence intervals for both the 'standard' ($c = c_0$) and the 'optimal' ($c = c^*$) cases, and for each value of F_b.

4.7 Conclusion - discussion

To improve current ML based physics in climate models, the effort is usually only focused on improving both the learning sample and the *offline* fit of the ML model. But these ML models can and need to be further improved by using an *online* metric to their calibration (Schneider et al., 2017). Thus, we propose a method to apply *online* calibration to ML parameterizations.

The key novelty of our approach is to include some of the physical parameters $\boldsymbol{\theta}$ among the input variables of the NN. The NN model is fitted *offline* to a learning sample of outputs from an high-resolution climate model. In this way, the NN parameterization is able to emulate the physics not only for one single $\boldsymbol{\theta}_0$, but for a range of values of $\boldsymbol{\theta}$. When the fitted NN model is plugged-in to replace the physical parameterization, the value of parameters $\boldsymbol{\theta}$ is calibrated *online*, as to reduce errors on long-term statistics of the climate model. As a proof-of-concept experiment, our

methodology is demonstrated using L63 and L96 models. We show that our method can be used to optimize the value of some parameters to compensate long-term errors due to biases carried by another parameter which cannot be calibrated.

In addition to the reduction of long-term model errors, including some physical parameters among the ML model inputs can also increase the confidence we have in ML parameterizations, the interpretability of which is often questioned. Supplementary parameters can also be used to estimate uncertainties related to some processes. However, although satisfying results were obtained using toy models, further research is needed to test our methodology into real climate models. The generation of a learning sample for subgrid scale parameterizations is already a challenging issue; this task is even more difficult in our new method as a sampling of $\boldsymbol{\theta}$ is needed in addition to the sampling of $\mathbf{x}$. The numerical cost of the generation of the learning sample is thus increased. If our approach was to be applied to a climate model, the methodology described for the L96 model could be applied. This would require a set of short high-resolution integrations for several values of $\boldsymbol{\theta}$, e.g., taken from a LHS on $\boldsymbol{\theta}$ only. In this way the number of model integrations would be kept small, while preserving a large learning sample on $(\mathbf{x}, \boldsymbol{\theta})$ (as each integration provides a large sample of $\mathbf{x}$ values). In the L96 example, we did not discuss which minimum value of the number of integrations N_i is sufficient to efficiently emulate the system. Finding such a minimum value will require careful examination in the case of a climate model. Once a satisfying learning sample is built, the next hurdle to overcome is the fit of the NN model. For more complex models, a simple feed-forward neural network may be insufficient and the choice of MSE as the *offline* metric may not be relevant. Finally, finding the optimal *online* metric can also be a strenuous issue.

Acknowledgments

We are grateful for the insightful comments and suggestions made by Olivier Geoffroy.

Data Availability Statement

Code is made available at: `https://zenodo.org/record/6141165`.

4.8 Des modèles jouets aux GCM

Pendant la revue de l'article, des inquiétudes nous ont été adressées concernant le coût de la réalisation de l'échantillon d'apprentissage. Ajouter un paramètre parmi les variables en entrée du NN implique l'exploration d'une ou plusieurs directions supplémentaires de l'espace dans lequel vit l'échantillon d'apprentissage. Le coût de l'échantillon d'apprentissage étant déjà élevé, il le sera nécessairement davantage en incluant des paramètres supplémentaires.

Les éléments dont on dispose nous font penser que la taille de l'échantillon d'apprentissage, dans le cas d'un GCM, ne serait pas beaucoup plus grande que celle utilisée dans le cas de L96. Une centaine de valeurs de $\boldsymbol{\theta}$ pourrait être suffisante. À partir d'une valeur de $\boldsymbol{\theta}$ ainsi que d'une condition initiale $\mathbf{x}_0$ aléatoire, on pourra générer une série temporelle de courte durée (quelques dizaines de pas de temps), sur l'ensemble du globe, soit environ $2{,}5.10^4$ colonnes à chaque pas de temps à une résolution horizontale de 150 km environ. Contrairement à la méthode actuellement utilisée, consistant à réaliser une longue simulation avec une seule valeur de $\boldsymbol{\theta}$, la méthode proposée ici revient à réaliser plusieurs séries temporelles de courte durée, en faisant varier $\boldsymbol{\theta}$. L'échantillon d'apprentissage ainsi construit, avec 100 valeurs de $\boldsymbol{\theta}$ échantillonnées, serait de taille $N \approx 100 \times 2,5.10^4 \times 10 = 2,5.10^7$, ce qui est plus grand que la taille des échantillons d'apprentissage utilisés dans la littérature.

Un deuxième point méritant des développements supplémentaires concerne le choix des métriques de long terme. Pour déterminer quelles valeurs peuvent être utilisées pour construire la métrique d'optimisation m, nous avons représenté les différentes métriques à régler (i.e., le biais et l'écart-type), en fonction de certains paramètres du modèle. Dans le cas de L63, les paramètres biaisés et à optimiser sont β et ρ. Sur la Figure 4.4, on constate que $\overline{x}_1$ et $\overline{x}_2$ ne permettent pas de mettre en évidence un lien entre β et ρ. Ces valeurs moyennes ne seront donc pas prises en compte dans la métrique m. En revanche, $\overline{x}_3$ et σ_x mettent en évidence des gradients opposés dans le plan (β, ρ), ce qui est favorable pour une optimisation. La métrique finale sera donc calculée en fonction de $\overline{x}_3$ et des écart-types σ_x.

En cas d'application à un GCM de la méthode présentée ici, une telle analyse devra être menée pour déterminer s'il est possible d'optimiser la valeur de certains paramètres, et également pour construire la métrique m à utiliser. On peut remarquer que le modèle L63 n'est donc pas un modèle où la calibration des paramètres est simple : nous avons donc perdu deux degrés de liberté, ce qui restreint le nombre de paramètres dont on peut calibrer la valeur. Cette limite est propre au modèle L63 ; dans le cas d'un GCM, il est possible d'utiliser la valeur moyenne des variables pour calibrer celle des paramètres (travailler avec les valeurs moyennes est sans doute plus aisé que d'utiliser les écart-types).

Dans ce chapitre, nous avons présenté une méthode permettant d'obtenir une

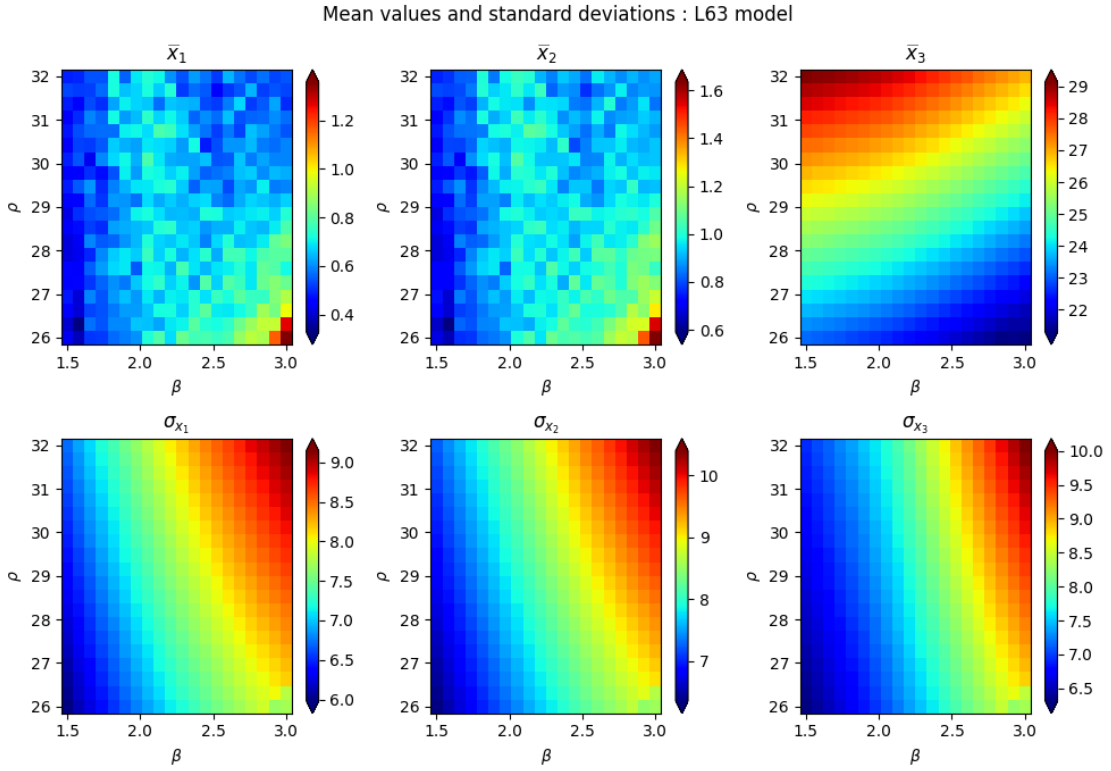


Figure 4.4: *Valeur de différentes métriques statistiques du modèle L63, en fonction des paramètres ρ et β. Quelles sont les métriques de long-terme que l'on peut utiliser pour trouver la valeur optimale de β, lorsque ρ est biaisé ? $\overline{x}_1$ et $\overline{x}_2$ ne font apparaître aucune relation entre β et ρ. Mais $\overline{x}_3$ et les écart-types σ_x varient dans des directions opposées, ce qui permet l'optimisation de la valeur de β lorsque celle prise par ρ varie.*

physique NN qu'il est possible de calibrer une fois qu'elle a été branchée dans le modèle dynamique. Pour cela, nous proposons d'inclure un certain nombre de paramètres pertinents pour l'optimisation parmi les variables en entrée du NN. En plus des variables en entrée $\mathbf{x}$, l'échantillon d'apprentissage utilisé pour ajuster le NN échantillonne également ces paramètres supplémentaires $\boldsymbol{\theta}$. Une fois la physique apprise branchée dans le GCM, la valeur de ces paramètres supplémentaires peut être optimisée pour régler les statistiques de long terme du modèle. L'application de la méthode a été illustrée grâce aux modèles L63 et L96.

Chapitre 5

Apprentissage d'un schéma de convection profonde

5.1 Introdution

Dans les chapitres précédents, nos travaux se sont concentrés sur des modèles jouets. Bien que ces modèles permettent de mieux comprendre les aspects théoriques des différents problèmes abordés, leur utilisation ne permet pas d'appréhender certains problèmes liés à l'utilisation de GCM. En particulier, les dimensions en jeu dans un GCM sont bien plus grandes que dans les modèles de Lorenz (d'au moins un facteur 10).

Un pas intermédiaire entre le développement des paramétrisations pour les modèles jouets et pour les GCM peut être l'apprentissage d'un schéma de paramétrisation. Cette étape pourrait permettre de réaliser une étude dans un cadre plus réaliste que celui des modèles jouets. De plus, une telle étude pourrait également mettre en évidence d'éventuels problèmes liés à la manipulation de jeux de données dont le volume numérique dépasse largement ceux utilisés pour l'étude des modèles jouets.

Ainsi, dans ce chapitre, nous nous intéressons à l'apprentissage du schéma de convection profonde implémenté dans la version la plus récente du modèle de climat du CNRM, ARPEGE-Climat.

5.2 La paramétrisation de la convection profonde dans le modèle ARPEGE-Climat

5.2.1 Le modèle de circulation générale ARPEGE-Climat

Le modèle support de cette étude est le modèle de circulation générale ARPEGE-Climat (Roehrig et al., 2020). La grille utilisée, appelée *grille réduite*, est caractérisée par une résolution variable en fonction de la latitude. Les points de grille sont plus densément répartis sur la bande tropicale, où la résolution horizontale atteint environ 150 km. Cette résolution décroît progressivement vers les pôles, pour atteindre plusieurs centaines de kilomètres. La structure verticale d'une colonne est décrite à l'aide de plusieurs niveaux verticaux. Dans ARPEGE-Climat, il y a 91 niveaux verticaux du sol jusqu'au sommet de la mésosphère.

À une telle résolution, seuls les phénomènes de très grande échelle sont résolus explicitement, tels que les structures de circulation générale (e.g., cellules de Hadley, cellules de Walker). Les processus se produisant à des échelles plus petites que la résolution du modèle, notamment horizontale, doivent donc être paramétrisés. Les paramétrisations sont à l'origine de biais bien connus dans les GCM. Par exemple, les modèles de climat permettent la représentation de la zone de convergence intertropicale (*Intertropical Convergence Zone*, ITCZ), une bande d'intense convection au niveau de l'équateur, où les apport d'humidité depuis les Sud et le Nord convergent. L'ITCZ est composée de cellules convectives d'étendue horizontale souvent plus petite que 150 km. Cette zone de convergence est vue par le modèle sous

forme de deux bandes d'intenses précipitations, alors qu'il n'y en a qu'une seule selon les observations (biais du *double ITCZ*, e.g. Oueslati and Bellon, 2015).

5.2.2 Le schéma de Tiedtke-Bechtold

Dans sa version actuelle, le modèle ARPEGE-Climat utilise le schéma de convection profonde de Tiedtke-Bechtold (Tiedtke, 1993). Son implémentation dans le modèle numérique du Centre Européen pour les Prévisions Météorologiques de Moyen Terme (CEPMMT), également utilisée dans la version plus récente d'APREGE-Climat, est décrite dans la note technique Bechtold (2017). Cette paramétrisation décrit les processus de convection profonde sous-maille (e.g., orages de petite taille). De manière simplifiée, le fonctionnement du schéma peut être décrit en plusieurs étapes, détaillées dans l'Encadré 5.1.

Encadré 5.1 : fonctionnement du schéma de convection profonde de Tiedtke-Bechtold

1. Déclenchement. La convection profonde est associée à une instabilité des masses d'air. La masse d'air peut être instable à partir de la surface, ou bien à partir d'un certain niveau vertical donné (dans ce cas-là, on parle d'*orages d'atmosphère libre* (OAL), également décrits par le schéma de Tiedtke-Bechtold). On s'intéresse ici à la convection profonde, définie dans le schéma par la présence d'un nuage convectif (cumulus) dont l'étendue verticale est supérieure à 200 hPa. Cette étape est appelée *déclenchement* (ou *triggering*).
2. Convection. Une fois les colonnes où la convection profonde se produit identifiées, le schéma de convection profonde s'active pour celles-ci. Le schéma produit des flux supplémentaires, par exemple pour les précipitations ou les rafales de vent (courants de densité). Ces flux seront prise en compte dans le calcul des tendances issues des processus sous-maille.

5.3 L'apprentissage

5.3.1 L'échantillon d'apprentissage

On dispose des données issues d'une simulation d'ARPEGE-Climat à une résolution horizontale de 150 km, d'une durée un an, qui ne comprend pas le *spin-up* du modèle, correspond à la durée de mise en équilibre du modèle de climat. Les forçages (e.g., température de la surface de la mer, émissions de gaz à effet de serre) utilisés sont ceux correspondant à l'année 1979. Les données simulées sont enregistrées toutes les heures, ce qui correspond à 8760 pas de temps disponibles pour l'apprentissage. Les données sont enregistrées pour l'ensemble des colonnes atmosphériques dans le modèle, au nombre de 24572. Les variables décrivant l'état thermodynamique de ces

colonnes enregistrées sont celles en entrée du schéma de Tiedtke-Bechtold, décrites dans la section précédente (i.e., profils verticaux sur 91 niveaux de s, q_T et flux de surface SHF et LHF), ainsi que de quelques variables supplémentaires (pression de surface P_s, profil vertical de vitesse verticale w, masque terre/mer LSM). En plus de ces variables, on enregistre une partie des flux en sortie du schéma de Tiedtke-Bechtold. Il s'agit des flux d'énergie statique sèche dT, de flux d'humidité dq et de flux de précipitations d(precip).

L'échantillon d'apprentissage est créé à partir des données issues de cette simulation. Le volume numérique de cette simulation étant conséquent, on procède à une sélection d'un sous-ensemble des données pour construire l'émulateur NN.

Tout d'abord, on peut remarquer que la convection profonde se produit uniquement dans la troposphère. Ainsi, durant l'apprentissage, on peut ignorer les niveaux verticaux décrivant les couches au-dessus de la troposphère. Le niveau le plus élevé considéré correspond donc au niveau le plus haut où le schéma s'est activé (i.e., le schéma a produit des flux non nuls) : il s'agit du niveau 56 à partir de la surface. Par conséquent, les variables en sortie seront apprises sur 56 niveaux. Concernant les variables en entrée, seule la description de la troposphère est pertinente. Les flux étant calculés à l'interface entre deux niveaux verticaux consécutifs, les profils en entrée du NN seront considérés sur 55 niveaux verticaux.

Deuxièmement, les données doivent être partitionnées en sous-échantillons *train*, *test* et *validation*. Dans notre cas, le plus simple est de commencer par l'échantillon de validation, qui sera utilisé pour l'évaluation des différents NN obtenus, et qui doit être aussi indépendant que possible des deux autres sous-échantillons. On choisit une série temporelle comme échantillon de validation. Il faut environ un mois de données pour voir les différentes structures à de grande échelle apparaître. De plus, le mois de validation devrait être indépendant de l'échantillon d'apprentissage : on choisit le mois d'octobre de la simulation pour la validation, assez éloigné des six premiers mois de l'année. L'échantillon de validation est composé de l'ensemble des colonnes verticales disponibles, dans le but de pouvoir tracer des cartes permettant d'évaluer le NN. Étant donnée la capacité de calcul disponible, nous avons choisi de ne garder les données que toutes les trois heures. Ainsi, nous avons 248 pas de temps pour 24572 colonnes dans l'échantillon de validation, soit un total d'environ 6×10^6 individus.

Le choix de l'échantillon d'apprentissage, composé des échantillons *train* et *test*, est également conditionné par la capacité de calcul que nous avons à notre disposition. On souhaite passer en revue le plus grand nombre de situations possibles. Puisque les données parcourent les hémisphères Nord et Sud, il suffit de disposer de six mois de données pour échantillonner toutes les saisons. Ainsi, on utilisera seulement les données de janvier à juin inclus pour créer l'échantillon d'apprentissage. Parmi ces données, le choix de l'échantillon d'apprentissage est aléatoire (selon une loi uniforme) : la puissance de calcul disponible permet de garder 1500 colonnes par pas de temps pour créer l'échantillon d'apprentissage. L'échantillon d'apprentissage

est donc composé de données sur 4344 pas de temps et 1500 colonnes à chaque pas de temps, ce qui représente un total d'environ $6,5 \times 10^6$ colonnes ou individus. L'échantillon ainsi obtenu est enfin partitionné de manière aléatoire entre *train* et *test*, dans les proportions 80% et 20% respectivement. Il est à noter que ces sous-échantillons comportent des colonnes où le schéma s'est activé, mais aussi des colonnes sans convection profonde. Cela est nécessaire pour l'apprentissage du déclenchement du schéma. Nous n'avons pas équilibré la proportion des colonnes où la convection s'active et ne s'active pas dans l'échantillon d'apprentissage.

Enfin, on peut choisir les variables en entrée de l'émulateur NN. Un premier choix spontané consiste à utiliser les variables utilisées par le schéma que l'on souhaite émuler (i.e., les profils verticaux de s et q_T ainsi que les flux de surface LHF et SHF). Mais d'autres configurations pourraient également être testées. Outre l'ajout de variables supplémentaires, il existe également des changements de variables qu'il est possible d'effectuer. Par exemple, pour donner l'information sur l'humidité contenue dans les couches, on peut utiliser l'humidité totale q_T ou bien l'humidité relative RH. Notre premier choix porte vers les variables en entrée suivantes : profils verticaux d'énergie statique s, d'humidité totale q_T et d'ascendances w, flux de surface de chaleur latente LHF et de chaleur sensible SHF.

5.3.2 La structure du réseau de neurones

Le problème que l'on envisage de traiter est une régression multiple en grande dimension. Pour le premier modèle de ML, la dimension en entrée est de $3 \times 55 + 2 = 167$ et une dimension en sortie de $3 \times 56 = 168$. Pour toutes les raisons évoquées dans la conclusion du Chapitre 1, nous utiliserons des NN plutôt que des forêts aléatoires. Étant donnée la formulation du problème, le choix se porte aussi préférentiellement vers des NN de type Dense. Dans ce cas, chaque variable atmosphérique, à chaque niveau vertical, est traitée comme une variable en entrée à part (i.e., la cohérence des variables dans un même profil vertical n'est pas exploitée). Comme dans la littérature consacrée à l'apprentissage des paramétrisations avec des NN (voir le Chapitre 2), nous nous orientons également vers l'utilisation d'un NN simple implémentant uniquement des couches Dense. Toutefois, nous avons essayé plusieurs architectures différentes.

- **NN Dense simple.** Architecture majoritairement utilisée dans la littérature. Chaque variable à chaque niveau vertical est traitée comme une variable en entrée à part. Le meilleur NN de la sorte est composé de 5 couches, avec 512 neurones ou unités sur chacune d'entre elles.

- **NN Dense par profils.** Cette architecture a pour but de séparer le traitement des profils verticaux et des variables scalaires. Le réseau Dense est composé de plusieurs étages : le premier étage est composé d'autant de branches qu'il y a de profils verticaux. Chaque profil vertical est traité de manière individuelle par une telle branche. La structure des ces branches est en "entonnoir"

: la dimension en entrée est de 55, pour se ramener à la sortie à une dimension q bien inférieure (e.g., $q = 2$). Les sorties de ces branches sont ensuite concaténées entre elles ainsi qu'avec les différentes variables scalaires en entrée du modèle. Le 2ème étage prend comme entrée les variables concaténées. Il correspond à un réseau implémentant uniquement des couches Dense.

- **NN avec des couches Conv1D.** Cette architecture vise également à utiliser l'information contenue dans les profils verticaux. Par analogie avec le traitement d'images, la dimension spatiale correspond aux niveaux verticaux, et les canaux (e.g., RGB) correspondent aux différents profils verticaux (e.g., s, q_T). Ce réseau est également composé de deux étages. Le premier étage correspond à un NN convolutif classique : une succession de couches Conv1D et de MaxPooling, réduisant la dimension à une faible valeur q (e.g., $q = 6$) après une couche Flatten finale. Comme dans l'architecture du NN précédent, le deuxième étage correspond à un petit réseau Dense qui prend en entrée la sortie du premier étage ainsi que les variables scalaires concaténées.

- **NN avec des couches récurrentes.** L'utilisation de couches récurrentes (e.g., LSTM) paraît adaptée à la forme de nos données. La convection profonde correspond à l'élévation d'une masse d'air de la surface[1] au sommet de la troposphère. Suivre l'ascendance d'une telle particule d'air revient à suivre son altitude. On peut donc voir une équivalence entre les axes vertical et temporel. Les couches récurrentes sont prévus pour le traitement de séries temporelles : dans le cas de la convection profonde, on peut donc faire porter la mémoire non pas sur une série temporelle, mais sur l'axe verticale. Ainsi, le premier étage de ce NN correspond à un bloc récurrent, qui prend en entrée les profils verticaux en entrée du modèle, pour condenser l'information à q unités seulement (e.g., $q = 4$). Comme dans les deux modèles précédemment présentés, ces sorties seront ensuite concaténées aux variables scalaires en entrée du modèle, et l'ensemble de ces données sera traité par un bloc Dense.

Bien que l'utilisation des architectures visant à exploiter l'information contenue dans les profils verticaux paraît prometteuse, les scores obtenus par l'ensemble des méthodes sont comparables (à l'exception du modèle utilisant des blocs convolutifs, pour lequel les scores obtenus sont moins bons). Ainsi, par parcimonie et simplicité d'implémentation dans ARPEGE-Climat, nous utiliserons dans la suite la première architecture présentée, composée d'une succession de couches Dense.

5.3.3 Le choix des paramètres de l'apprentissage

Avant l'apprentissage, les variables en entrée sont centrées et réduites. Cette normalisation est indispensable, car il y a des différences très importantes en ordre de

[1] Dans le cas particulier des orages d'atmosphère libre, les particules d'air se soulèvent à partir des niveaux au-dessus la surface. L'explication sur l'utilisation des couches récurrentes reste donc valable.

grandeur entre les variables (e.g., les valeurs prises par l'énergie statique sèche s sont de l'ordre de 10^5 alors que celles prises par l'humidité q_T sont de l'ordre de 10^{-6}). Nous avons essayé la normalisation décrite dans Brenowitz and Bretherton (2018), qui consiste à centrer et réduire les variables scalaires, et normaliser les profils verticaux en utilisant l'écart-type calculé sur l'ensemble du profil sur l'échantillon *train*. Cette normalisation a été rapidement abandonnée au profit de centrer-réduire les variables, car l'utilisation de ce dernier a permis d'obtenir de meilleurs résultats (*offline*).

La valeur de la plupart des hyperparamètres est optimisée manuellement (e.g., taux d'apprentissage, nombre de couches, nombre de neurones sur chaque couche), car leur valeur n'a que peu d'impact sur le résultat final. Le paramètre ayant le plus fortement influencé les scores obtenus était la taille du *batch* : les meilleurs résultats ont été obtenus avec un *batch* de taille 256. L'apprentissage s'est effectuée sur 35 *epochs* – cette valeur optimale a été optimisée manuellement. Le taux d'apprentissage initial est fixé à 10^{-3} ; cette valeur suit une décroissance exponentielle par paliers tous les 3 *epochs*. Pendant l'apprentissage, seuls les poids du NN associés aux fonctions perte les plus faibles sur l'échantillon de validation sont enregistrés.

La métrique que nous utilisons est l'erreur quadratique moyenne (*Mean Squared Error*, MSE) sur les flux émulées, en particulier pour sa simplicité d'utilisation. Nous avons également effectué un apprentissage visant à maximiser le score de R^2, mais les résultats obtenus ont été moins bons que ceux obtenus en utilisant la MSE. Afin d'éviter le surapprentissage, le "meilleur" NN est obtenu en enregistrant les poids et biais du modèle associé à la MSE la plus basse sur l'échantillon *test*. Les résultats obtenus avec ce "meilleur" modèle sont détaillés dans la section suivante.

5.3.4 Choix des variables en entrée du NN

Nous avons essayé plusieurs configurations dans le but de déterminer laquelle conduisait aux meilleurs scores. En fonction des variables dont on dispose dans l'échantillon d'apprentissage, les configurations testées sont décrites dans le Tableau 5.1.

Pour chacune de ces configurations, on calcule le score de R^2 sur l'échantillon de validation, séparément pour chaque profil de flux en sortie. Les résultats sont résumés dans le Tableau 5.1. Bien que les scores de R^2 soient assez proches pour l'ensemble des modèles évalués, on constate que le meilleur modèle prend en entrée les variables physiques du schéma de Tiedtke-Bechtold, ainsi que le profil vertical des ascendances w et la pression de surface P_S. Mais l'amélioration du modèle reste minime. Enlever le profil vertical de soulèvement de grande-échelle w déteriore le score obtenu. Enfin, l'ajout du masque terre-mer ne permet pas d'améliorer le modèle. Dans la suite, pour des raisons de parcimonie, nous utiliserons donc le modèle v1, dont la dimension en entrée est de 167.

nom NN	**3D inputs**	**2D inputs**	R^2 **d**$T(z)$	R^2 **d**$q(z)$	R^2 **d(precip)(z)**
v1	s_d, q_T, w	SHF,LHF	0,3157	0,3502	0,453
v2	s_d, q_T	SHF,LHF	0,2854	0,3128	0,3966
v3	s_d, q_T, w	SHF,LHF,P_s	0,3321	0,3638	0,4798
v4	s_d, q_T, w	SHF,LHF,P_s,LSM	0,3177	0,3410	0,4462
v5	s_d, q_T, w	SHF,LHF,LSM	0,3034	0,3291	0,4320
v6	s_d, RH, w	SHF,LHF	0,3204	0,3487	0,4531

Table 5.1: *Scores de R^2 calculés sur l'échantillon de validation, pour plusieurs configurations de variables en entrée du modèle NN. On constate seulement de faibles variations en ajoutant ou en retirant des variables.*

5.4 Validation *offline*

Avant d'être implémenté dans le modèle, l'émulateur NN du schéma de convection profonde doit être validé *offline*, en utilisant l'échantillon de validation décrit dans le paragraphe 5.3.1. Dans la suite, nous allons présenter quelques scores et beaucoup de figures.

5.4.1 Répartition spatiale de l'erreur

En modélisation météo-climatique, une des premières quantités à regarder pour l'évaluation d'un modèle est la *moyenne zonale* de certaines variables $\mathbf{x}$, pendant une simulation. Elle est définie comme la moyenne temporelle sur un cercle de latitude de $\mathbf{x}$:

$$\overline{\mathbf{x}}^{zm}(\text{lat}, z) = \frac{1}{N_t N_{\text{lon}}} \left(\sum_{t=1}^{N_t} \sum_{l=1}^{N_{\text{lon}}} \mathbf{x}(\text{lat,lon}, z, t) \right), \tag{5.1}$$

où N_t est le nombre de pas de temps dans la simulation et N_{lon} le nombre de point de grille sur une bande de latitude donnée. L'étude de la moyenne zonale permet de vérifier que les structures de circulation générale soient bien représentées. La circulation de grande échelle se met en place dans le but d'acheminer le surplus d'humidité et de chaleur des tropiques vers les latitudes plus élevées, via les *cellules de Hadley*. Ce processus est plus marqué dans l'hémisphère d'hiver, ce qui, dans le cas de notre échantillon de validation, correspond à l'hémisphère Nord (mois d'octobre), et correspond à un maximum des différents flux au niveau de l'équateur météorologique[2]. Sur la Figure 5.1, la moyenne zonale et verticale des différents flux en sortie du modèle de NN (d'énergie statique sèche dT, humidité dq_T et précipitations d(precip)) est représentée dans la colonne de gauche. On retrouve

[2]L'équateur météorologique correspond à l'endroit où les alizés convergent. Les alizés sont des vents de basse altitude, apportant une masse d'air froid et humide depuis les latitudes les plus élevées des tropiques vers l'équateur météorologique. Ces masses d'air froid et humides vont se réchauffer au niveau de l'équateur et donner naissance aux cellules convectives formant l'ITCZ. L'équateur météorologique se situe dans l'hémisphère d'hiver, entre 5°S et 5°N habituellement.

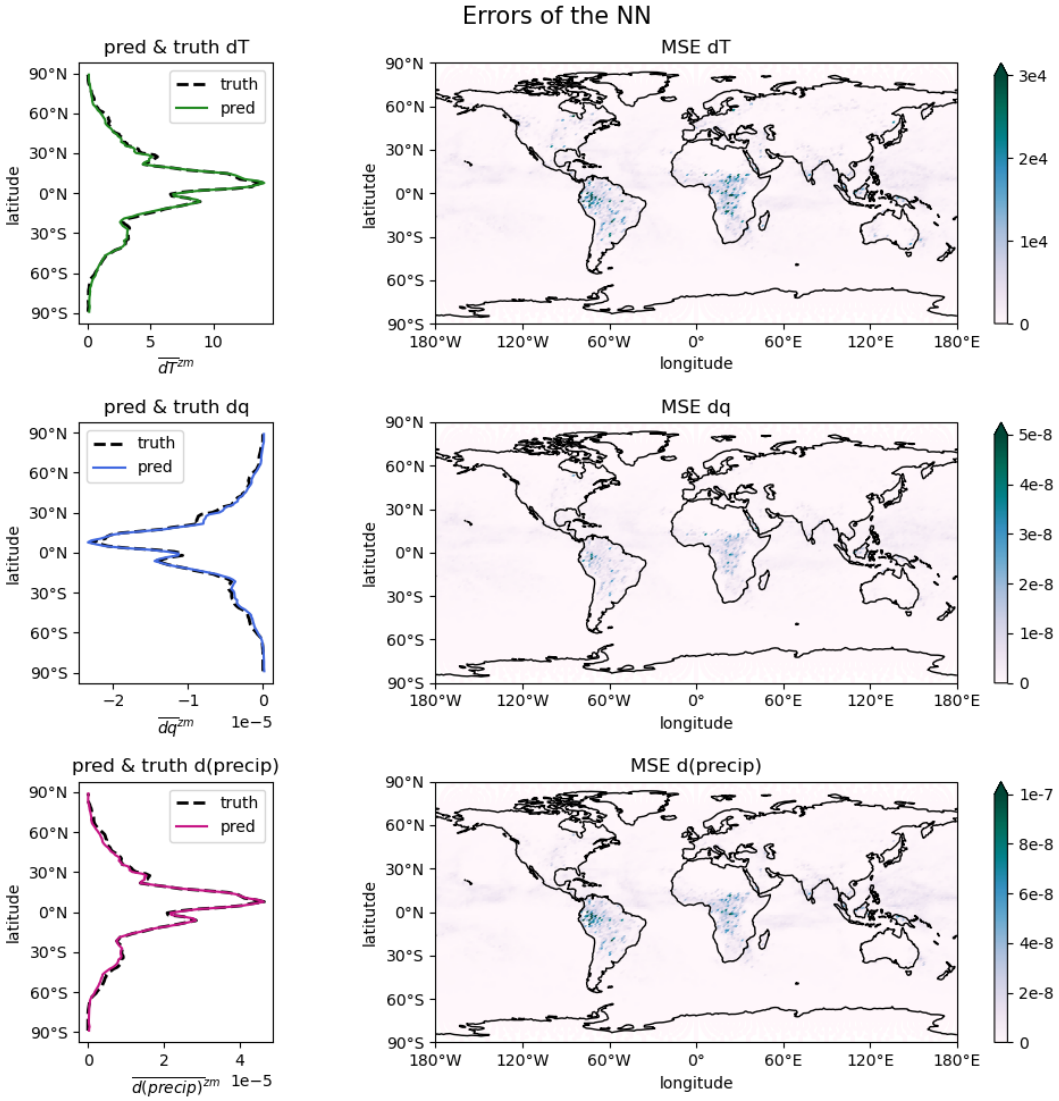


Figure 5.1: *Gauche : moyenne zonale et verticale des différents flux en sortie du NN (pred, en trait plein de couleur) et des flux dans l'échantillon de validation (truth, en tiretés noirs). Les valeurs représentées sont des moyennes sur le profil vertical. Droite : répartition spatiale de l'erreur quadratique moyenne (Mean Squared Error, MSE), calculée pour la prévision des différentes variables en sortie du NN, par rapport aux valeurs dans l'échantillon de validation.*

bien le maximum des différents flux situé au niveau de l'équateur, légèrement décalé dans l'hémisphère nord. On constate également une très bonne adéquation entre la moyenne zonale calculée sur l'échantillon de validation (colonne de gauche, tiretés noirs) et sur les prévisions du NN (colonne de gauche, traits pleins en couleurs).

L'écart le plus important entre les flux issues du schéma et du NN se trouve vers 10°S. Pour comprendre plus précisément à quel endroit le NN se trompe le plus, la colonne de droite de la Figure 5.1 représente les MSE calculées sur les profils verticaux à plusieurs endroits de l'espace. En revanche, entre 10°S et 25°S, on retrouve deux endroits où le NN n'a pas réussi à représenter les différents paramètres de manière satisfaisante. Plus précisément, il s'agit de la partie Nord-Ouest de l'Amérique du Sud ainsi que la partie centrale de l'Afrique. À ces endroits, l'intensité de l'ITCZ n'a pas été bien vue par le NN. Ce résultat est cohérent avec les scores de R^2 du modèle de NN utilisé, vus dans le Tableau 5.1 (respectivement, de 0.32, 0.35 et 0.45 pour dT, dq et d(precip)). Toutefois, pour les modèles de climat, avoir des erreurs faibles en moyenne zonale et temporelle est plus important que d'avoir des erreurs faibles au pas de temps. Pour la moyenne zonale de ces mêmes quantités, le score de R^2 est bien plus élevé (0.997, 0.995 et 0.997 pour dT, dq et d(precip) respectivement).

5.4.2 Coupe verticale de moyenne zonale

Il est également intéressant de vérifier que la structure verticale des prévisions issues du NN est cohérente avec celle de l'échantillon de validation. La Figure 5.2 représente les coupes verticales pour les différents flux calculés sur l'échantillon de validation. On constate un très bon accord global entre les flux issus de l'échantillon de validation (colonne de gauche) et celles calculées en utilisant l'émulateur NN (colonne de droite). Le NN parvient à représenter, avec une certaine finesse, les structures de grande échelle. Les valeurs maximales et minimales se situent à la bonne altitude, quelle que soit la variable.

5.4.3 Moyenne méridienne

La circulation méridienne se met en place pour redistribuer le surplus d'énergie (solaire) reçue sur une bande de latitude. Contrairement à la moyenne zonale, la moyenne méridienne d'une variable $\mathbf{x}$ issue d'une simulation sur N_t pas de temps est calculée donc en agrégeant les différentes latitudes, et non pas les longitudes :

$$\overline{\mathbf{x}}^{mm}(\text{lon}, z) = \frac{1}{N_t N_{\text{lat}}} \left(\sum_{t=1}^{N_t} \sum_{l=1}^{N_{\text{lat}}} \mathbf{x}(\text{lat,lon}, z, t) \right), \tag{5.2}$$

où N_{lat} est le nombre de latitudes dans la simulation de $\mathbf{x}$, pour une valeur de longitude donnée. Comme les transferts de chaleur et d'humidité se font essentiellement

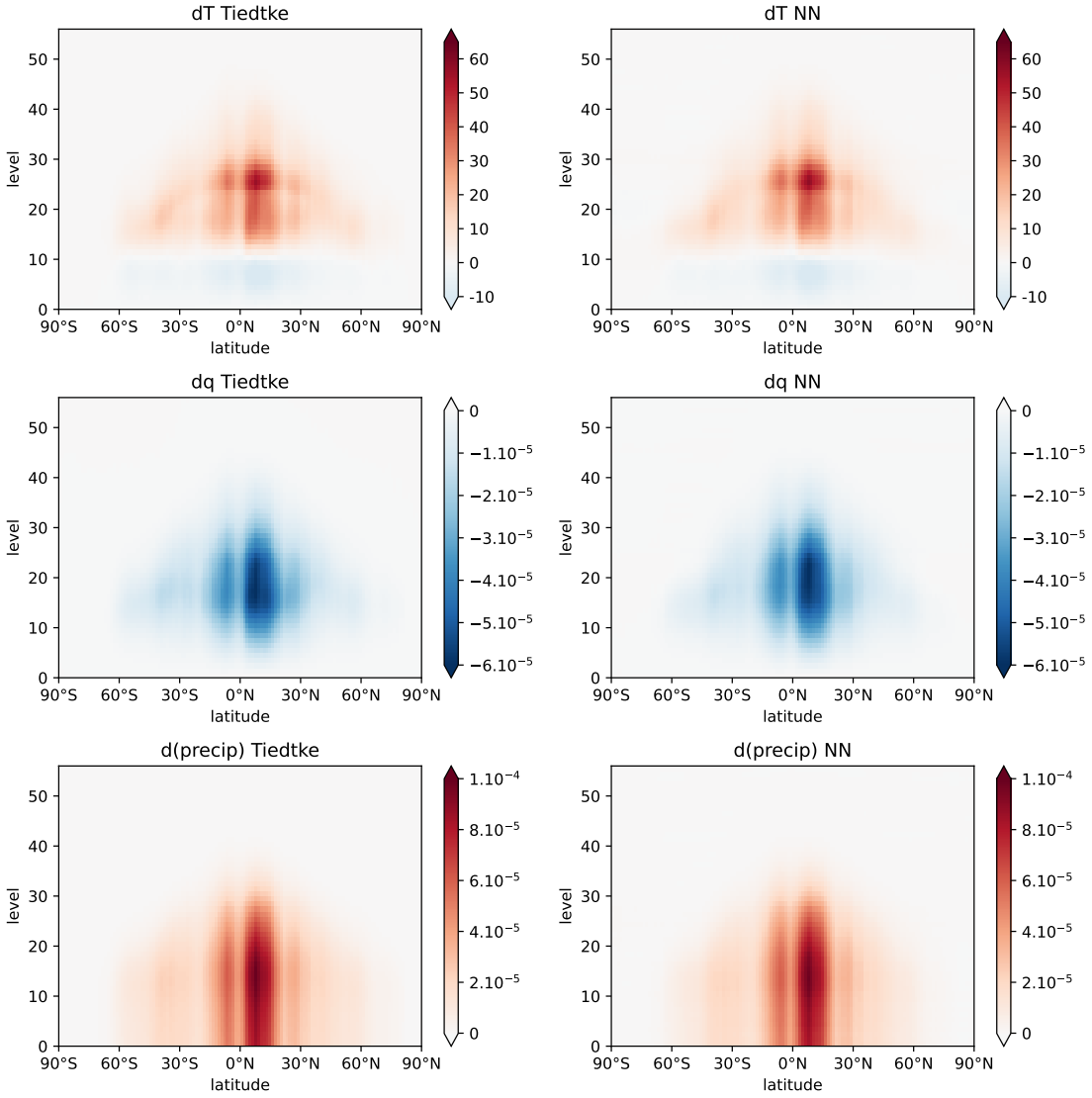


Figure 5.2: *Flux issus de l'échantillon de validation (gauche) et prédites sur l'échantillon de validation avec le NN (droite), en moyenne zonale. Les flux d'énergie statique (ligne du haut, dT) sont bien émulés par le NN. On retrouve en particulier un fort taux de réchauffement correspondant à l'ITCZ, vers 5 degrés Nord. La signature de l'ITCZ est également bien représenté par le NN pour le flux d'humidité dq associé. Enfin, les systèmes convectifs de l'ITCZ sont responsables de fortes précipitations, ce que l'on retrouve sur la dernière ligne de la figure. Les prévisions du NN montrent des champs légèrement plus lisses que ceux dans l'échantillon de validation.*

selon la latitude, les moyennes méridiennes sont interprétables uniquement par sous-régions, comme par exemple sur les tropiques. On va donc s'intéresser à la bande tropicale, qui s'étend de 30°S à 30°N. Il s'agit de la région où l'on constate les plus fortes erreurs entre le NN et le schéma à apprendre et également de l'endroit où le schéma de convection profonde est le plus actif. La moyenne méridienne d'un paramètre est habituellement plus bruitée que la moyenne zonale.

La Figure 5.3 représente les moyennes méridiennes, calculées sur la bande tropicale avec les prévisions du NN sur l'échantillon de validation ainsi que sur les flux disponibles dans ce même échantillon. Une nouvelle fois, et malgré la signature bruitée de la moyenne méridienne, on retrouve un très bon accord entre la variable cible de validation et les prévisions du NN.

5.5 Validation *online*

Dans la section précédente, nous nous sommes intéressés aux performances *offline* du NN, à travers les figures utilisant les données de l'échantillon de validation. Dans la suite, on s'intéresse à la validation *online*. Pour cela, la convection profonde apprise est implémentée dans ARPEGE-Climat, en remplacement du schéma de convection profonde initial. Les résultats issus d'une simulation réalisée avec ARPEGE-Climat et la physique apprise sont détaillés dans la suite.

5.5.1 Implémentation de la physique apprise dans ARPEGE-Climat

L'apprentissage de la convection profonde par le modèle de NN a été réalisé en Python, alors que ARPEGE-Climat est codé en Fortran. Pour implémenter le modèle de NN dans ARPEGE-Climat, nous avons utilisé un package développé par David Saint-Martin, permettant de traduire le modèle de NN dans un format lisible en Fortran. Tout d'abord, le package traduit le NN sous forme d'un fichier netCDF, où le NN est décrit sous forme de tableaux de poids et de biais, en y renseignant le type et la fonctions d'activation des différentes couches. Ensuite, la deuxième partie du package permet de lire les informations inscrites dans le fichier netCDF et de réaliser les opérations matricielles correspondant au *predict* du NN.

5.5.2 Réalisation de la simulation

On réalise une simulation de validation d'une durée de 1 an, en utilisant les forçages de l'année 2000, où le schéma de convection profonde a été partiellement remplacé par le NN. Les profils verticaux de flux de l'énergie statique sèche, d'humidité totale et de précipitations sont issues du NN. Les profils verticaux de flux de quantité de mouvement engendrés par des processus de convection profonde (e.g., les courants de densité) sont issus du schéma de Tiedtke-Bechtold. Pour faciliter ce premier essai, les orages d'atmosphère libre (OAL), correspondant à une convection débutant à la moyenne troposphère, sont désactivés ; ils ne représentent qu'une faible part

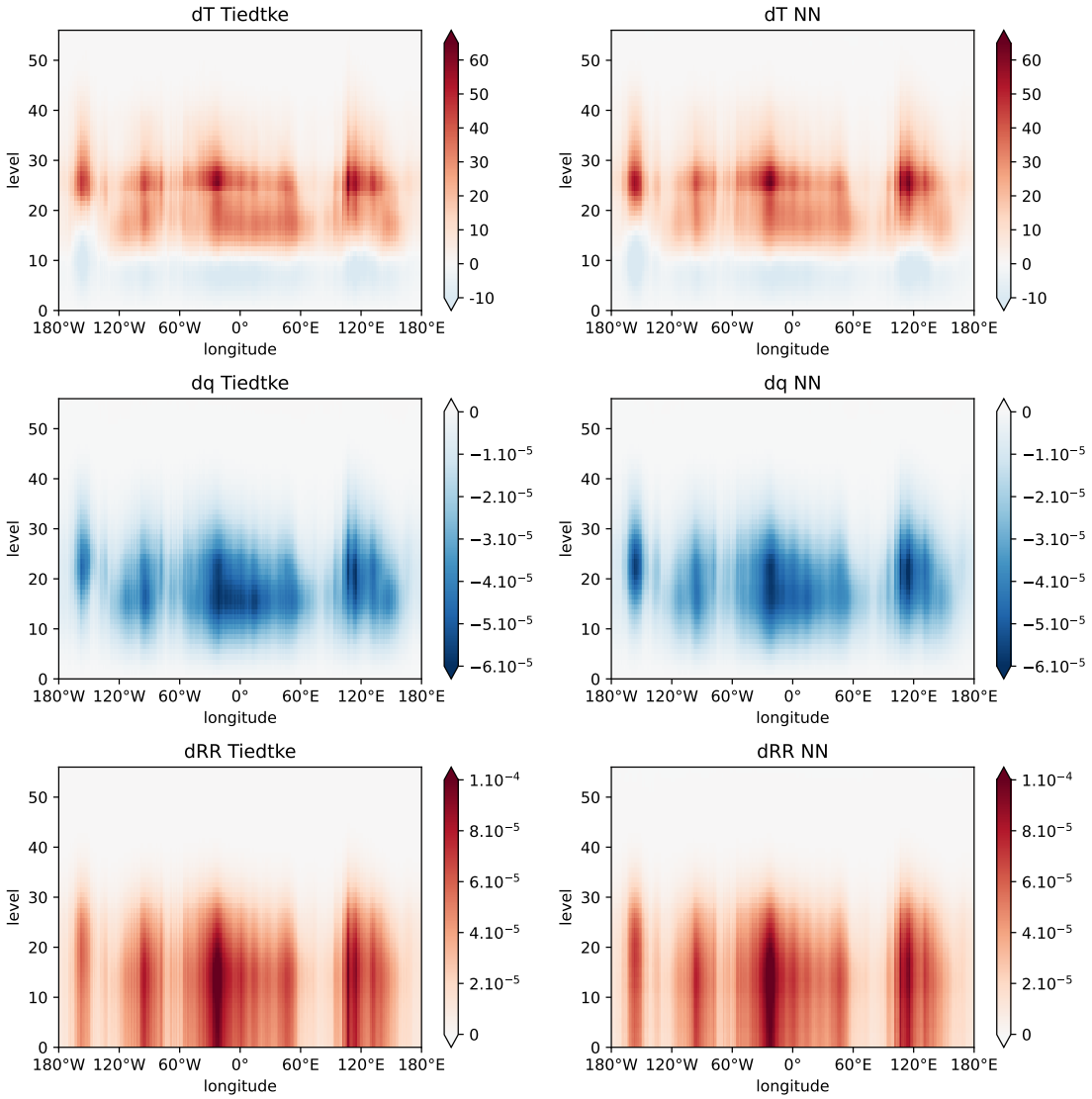


Figure 5.3: *Moyenne méridienne pour les flux en sortie du modèle NN, calculées sur les tropiques, sur l'échantillon de validation. La colonne de gauche représente, pour chaque paramètre, la variable cible. De haut en bas : dT, dq_T,$d(precip)$. La colonne de droite représente l'estimation NN pour les mêmes variables. Globalement, on retrouve un très bon accord entre la variable cible et les prévisions du NN sur l'échantillon de validation. La signature des différentes flux semble bien représentée par le NN.*

des orages observés.

Le premier résultat obtenu est la stabilité d'ARPEGE-Climat avec la physique apprise. La simulation de quelques années a été réalisée sans explosion (numérique) du modèle. Nous avons vu précédemment que la stabilité d'un modèle avec une physique apprise est loin d'être assurée. Dans notre cas, la stabilité est favorisée par le fait que l'on ne remplace pas la totalité, mais seulement une partie de la physique du GCM.

Pour l'analyse plus détaillée des résultats, on utilise au CNRM des *atlas*. Les atlas correspondent à un ensemble de figures réalisées après chaque changement effectué dans le modèle. Ces figures représentent des paramètres clés, permettant de vérifier rapidement la présence de certaines structures de grande échelle bien connues, comme l'ITCZ ou encore la présence de nuages stratiformes sur le bord ouest des continents. Les atlas permettent de comparer les résultats d'une simulation à la climatologie, définie comme la moyenne sur un certain nombre d'années d'observations ou de réanalyses. Pour les champs auxquels nous allons nous intéresser dans la suite, la climatologie est composée d'observations, dont la source sera précisée.

Précipitations annuelles

La présence de l'ITCZ est la première chose à vérifier. Celle-ci est particulièrement visible sur le champ de précipitations. Sur la Figure 5.4, on constate d'ores et déjà que l'ITCZ est présente (et est double). L'intensité de cet amas de convection est plus importante sur la simulation réalisée avec la convection profonde apprise que sur la climatologie. On peut également remarquer que la simulation avec la physique apprise a permis de représenter d'autres structures de précipitation, comme par exemple ceux se produisant au Nord de l'Amérique du Sud, ou bien à l'Est de l'Amérique du Nord. La structure spatiale des amas convectifs est globalement bien représentée dans la simulation avec la physique apprise, malgré quelques différences par rapport à la climatologie, notamment au nord de l'Océan Indien. La climatologie a été tracée à l'aide du jeu de données *Global Precipitation Climatology Project* (GPCP) (Adler et al., 2003).

Il est également intéressant de représenter la moyenne zonale et verticale des précipitations issues de plusieurs simulations. Celles-ci peuvent ensuite être comparées à la climatologie GPCP. Ces valeurs moyennes sont représentées sur la Figure 5.5.

Tout d'abord, on réalise une simulation ARPEGE-Climat où la convection profonde a été désactivée (bleu). On constate des écarts importants entre la moyenne zonale et verticale calculée sur la simulation et la climatologie (tiretés, noir), notamment au niveau de l'ITCZ. L'utilisation d'un schéma de convection profonde, comme le schéma de Tiedtke-Bechtold (rouge), permet une meilleure répartition de

l'humidité atmosphérique sur le globe, ce qui permet une représentation plus juste des précipitations selon les latitudes.La simulation ARPEGE-Climat avec la convection profonde apprise (vert) est plus proche de la climatologie. Les simulations réalisées avec la convection profonde apprise et le schéma de Tiedtke-Bechtold sont également assez proches, malgré des différences au-dessus des latitudes moyennes (entre 30 et 60 degrés nord et sud). À titre de comparaison, nous avons également représenté la moyenne zonale et verticale issue d'une simulation utilisation un schéma de convection différent. Dans sa version précédente, ARPEGE-Climat utilise le schéma PCMT pour représenter la convection profonde sous-maille (Guérémy, 2011) (jaune). La moyenne zonale et méridienne des précipitations, calculées sur les simulations réalisées avec les trois schémas de convection profonde, est similaire. La physique apprise permet donc de simuler une moyenne zonale et verticale assez réaliste.

Température de surface

De la même manière que pour les précipitations, on peut également s'intéresser à la température de surface issue d'une simulation et calculée sur les données climatologiques (Figure 5.6). Une nouvelle fois, on constate un bon accord entre la simulation et les données climatologiques, notamment en terme de moyenne zonale. Les plus gros écarts sont observés au niveau du Groënland, où les températures sont très faibles dans la simulation, ainsi qu'au niveau des reliefs (Hiamalaya notamment).

La climatologie a été tracée à l'aide du jeu de données *Climate Research Unit* CRU TS version 3.2 (Harris et al., 2014).

Couverture nuageuse

La couverture nuageuse est également un paramètre fortement dépendant de la convection profonde. Elle est exprimée en pourcentage couvert, pour chaque étage (bas, moyen et haut). La présence de nuages convectifs contribue à augmenter la couverture nuageuse, à chaque étage. Sur le champ de couverture nuageuse totale, on constate de plus gros écarts entre la climatologie et la simulation par rapport aux deux champs précédents (Figure 5.7). En particulier, on constate que le schéma est responsable de trop de nuages au-dessus des océans, ainsi qu'à certains endroits des surfaces continentales (Afrique Centrale, Nord de l'Amérique du Sud). La nébulosité totale au-dessus des régions polaires est également trop importante. L'étude de la moyenne zonale permet également de mettre en évidence des écarts plus forts entre la climatologie et la simulation.

L'analyse plus en détail des figures de l'atlas a permis comprendre que le schéma de convection profonde appris produit trop de nuages à l'étage haut (Figure 5.8). Sur les données climatologiques, la couverture nuageuse est généralement faible à l'étage haut (20-30%), alors qu'elle est très importante dans la simulation (>50-60%). Cela

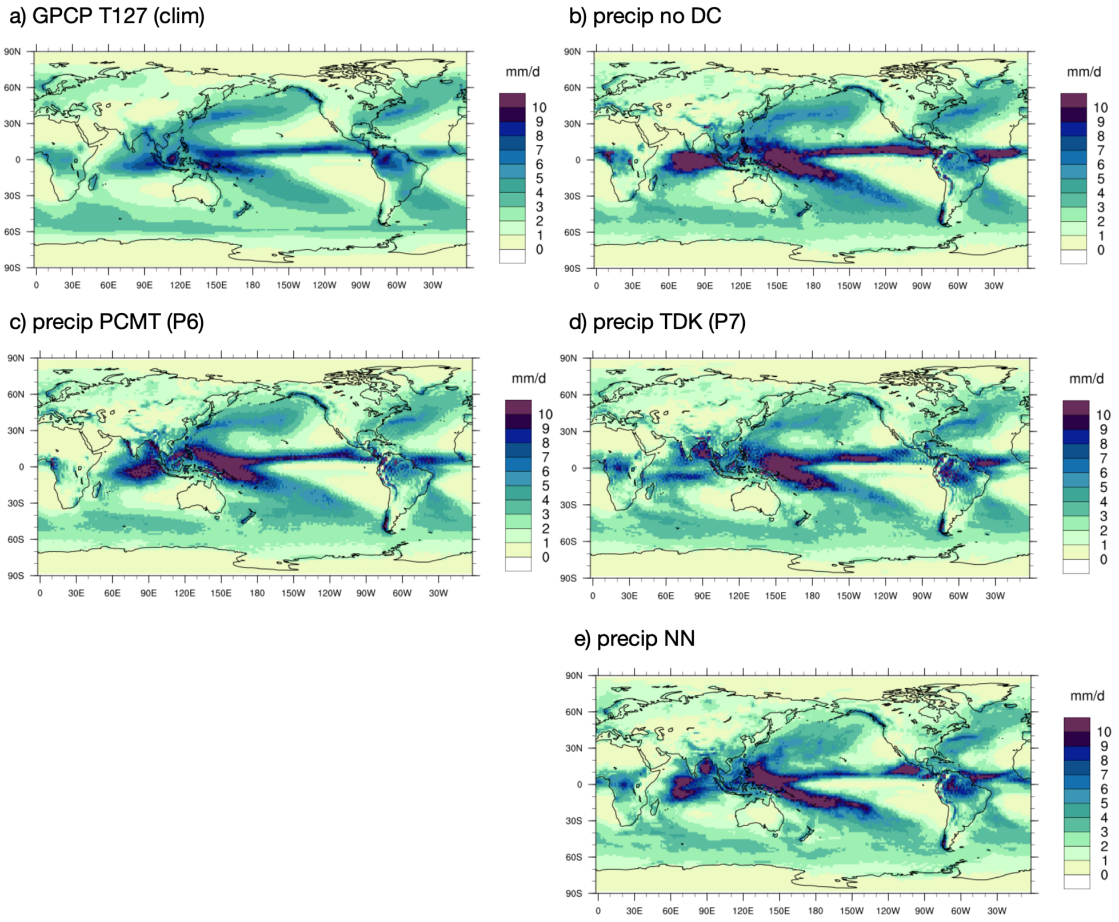


Figure 5.4: *Moyennes annuelles des précipitation calculées à partir de plusieurs jeux de données. Les figures sont issues de l'atlas. a) Climatologie des précipitations annuelles, représentées à partir du jeu de données CRUTS, version 3.2. b) Moyenne annuelle des précipitations, calculées à partir d'une simulation ARPEGE-Climat de 5 ans sans schéma de convection profonde. c) Moyenne annuelle des précipitations, calculées à partir d'une simulation ARPEGE-Climat avec le schéma de convection profonde de la version précédente. d) Moyenne annuelle des précipitations, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec le schéma de convection profonde actuelle (Tiedtke-Bechtold). e) Moyenne annuelle des précipitations, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec la schéma de convection profonde appris par le NN.*

signifie que les nuages convectifs issus du schéma de convection profonde appris ont tendance à s'étaler au niveau de la tropopause. Les nuages à l'étage haut modifient de manière significative l'albédo, ce qui peut avoir des conséquences directes sur d'autres champs. Mais une couverture nuageuse exagérée à l'étage haut est déjà un biais notable dans les simulations ARPEGE-Climat avec les schémas de convection profonde PCMT et Tiedtke-Bechtold.

La climatologie de la couverture nuageuse et de la couverture nuageuse à l'étage haut a été tracée à l'aide du jeu de données *International Satellite Cloud Climate*

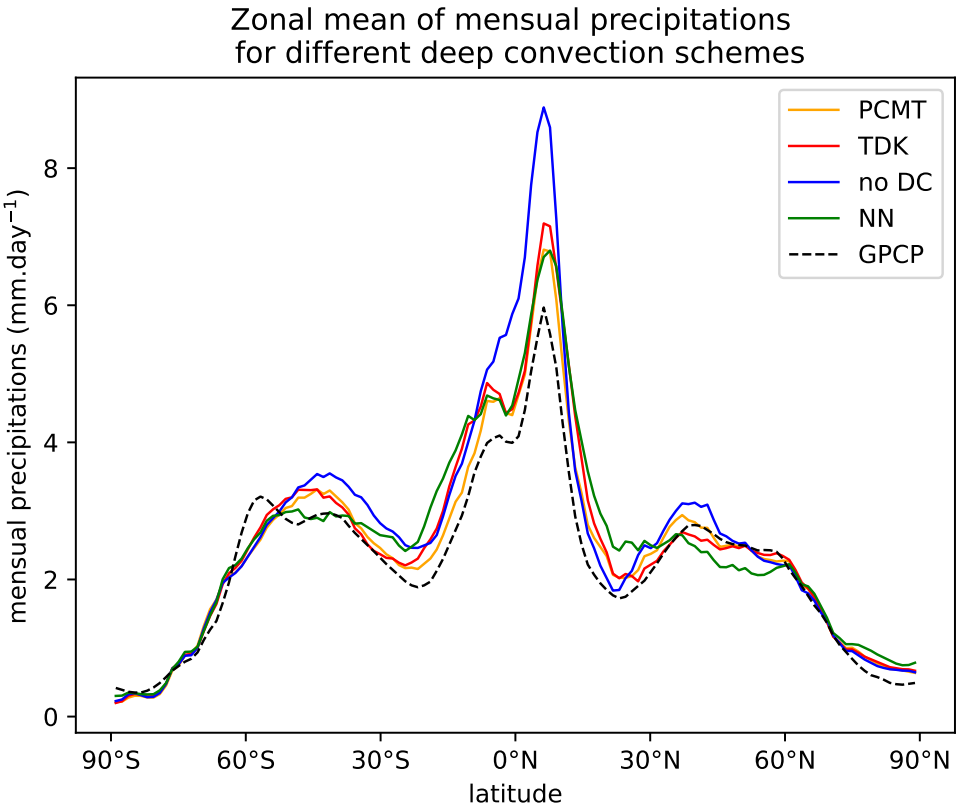


Figure 5.5: *Moyenne zonale des précipitations, calculées sur plusieurs jeux de données. Les données issues de simulation sont calculées sur un an, en utilisant les forçages de l'an 2000. Tiretés noirs : la climatologie est calculée à partir des données GPCP. Bleu : précipitations issues d'une simulation ARPEGE-Climat sans schéma de convection profonde. Nous avons également utilisé plusieurs schémas de convection profonde pour réaliser les simulations ARPEGE-Climat, et calculer la moyenne zonale des précipitations sur le jeu de données ainsi obtenu. Jaune : schéma de convection profonde de la version précédente d'ARPEGE-Climat, PCMT. Rouge : schéma de Tiedtke-Bechtold pour la paramétrisation de la convection profonde. Vert : schéma de convection profonde apprise par le NN.*

Project (ISCCP) (Young et al., 2018).

5.6 Analyse de sensibilité

Nous avons vu que l'un des problèmes majeurs liés à l'utilisation de techniques d'IA est que l'on n'en comprend pas directement le fonctionnement. On peut alors douter de la cohérence physique du schéma appris. Une manière de confirmer si les relations physiques entre différentes variables ont bien été apprises est de de vérifier que les deux réagissent de façon proche à des perturbations de petite échelle. Cela peut être évalué grâce à une *analyse de sensibilité*, permettant d'estimer l'influence d'une

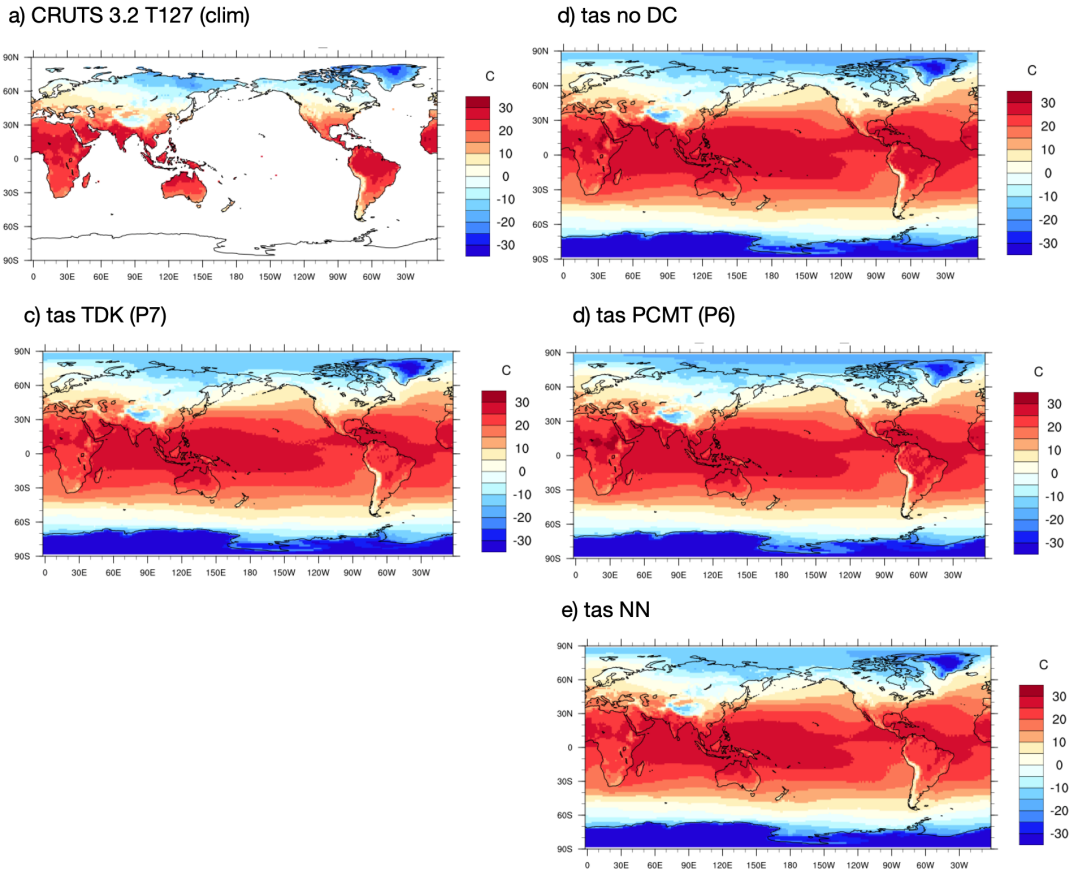


Figure 5.6: *Moyennes annuelles de la température de surface (TAS) calculées à partir de plusieurs jeux de données. Les figures sont issues de l'atlas. a) Climatologie des températures de surface annuelles, représentées à partir du jeu de données CRU TS version 3.2. b) Moyenne annuelle des températures de surface, calculées à partir d'une simulation ARPEGE-Climat de 5 ans sans schéma de convection profonde. c) Moyenne annuelle des températures de surface, calculées à partir d'une simulation ARPEGE-Climat avec le schéma de convection profonde de la version précédente. d) Moyenne annuelle des températures de surface, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec le schéma de convection profonde actuelle (Tiedtke-Bechtold). e) Moyenne annuelle des températures de surface, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec la schéma de convection profonde appris par le NN.*

variable en entrée donnée sur les variables en sortie. Dans la suite, nous allons nous intéresser à la sensibilité du schéma de Tiedtke-Bechtold et de celui de l'émulateur NN.

5.6.1 Les fonctions de réponse linéaire

Il existe plusieurs méthodes permettant de réaliser une analyse de sensibilité. Par simplicité, nous utiliserons les *fonctions de réponse linéaire* (*linear response functions*, LRF), technique également utilisée dans Brenowitz and Bretherton (2019). La

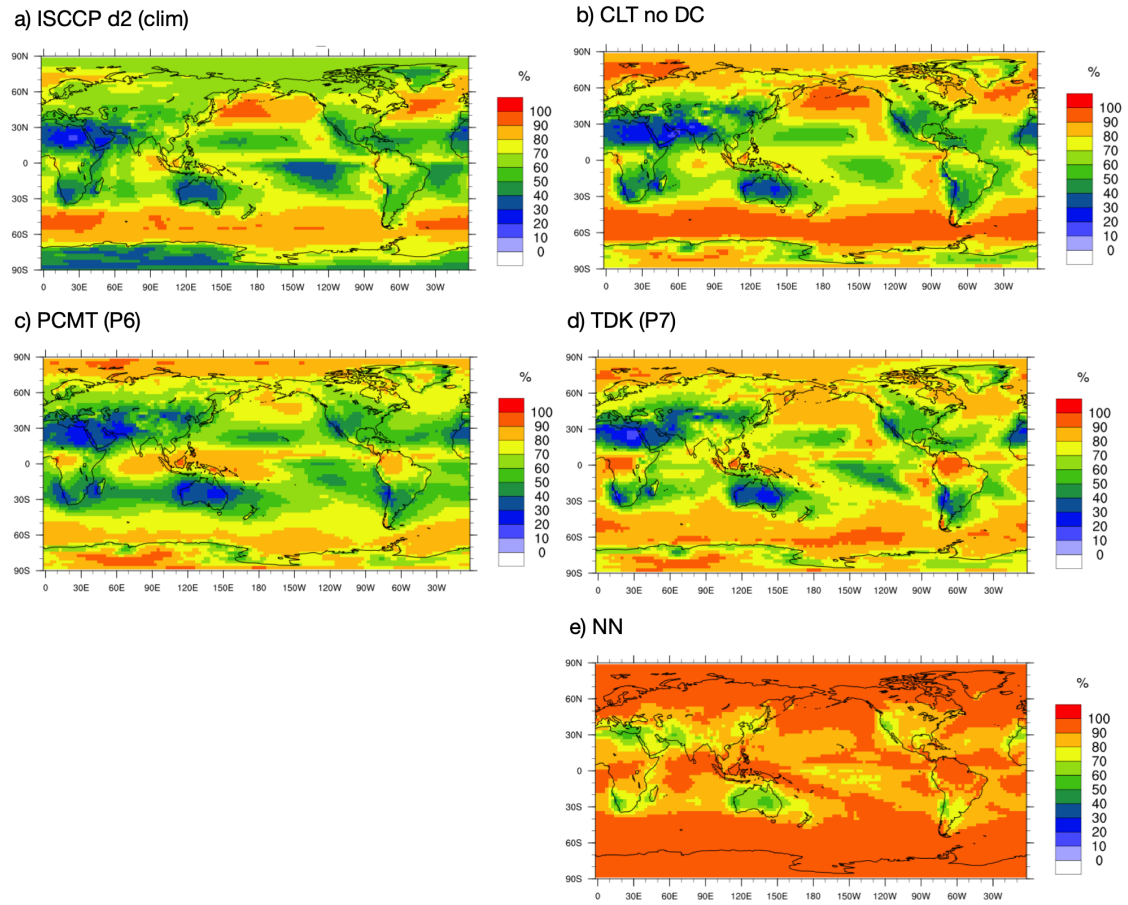


Figure 5.7: *Moyennes annuelles de couverture nuageuse (CLT) calculées à partir de plusieurs jeux de données. Les figures sont issues de l'atlas. a) Climatologie des couvertures nuageuses annuelles, représentées à partir du jeu de données ISCCP d2. b) Moyenne annuelle des couvertures nuageuses, calculées à partir d'une simulation ARPEGE-Climat de 5 ans sans schéma de convection profonde. c) Moyenne annuelle des couvertures nuageuses, calculées à partir d'une simulation ARPEGE-Climat avec le schéma de convection profonde de la version précédente. d) Moyenne annuelle des couvertures nuageuses, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec le schéma de convection profonde actuelle (Tiedtke-Bechtold). e) Moyenne annuelle des couvertures nuageuses, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec la schéma de convection profonde appris par le NN.*

LRF consiste à estimer les dérivées d'une fonction f, en fonction de ses différentes variables en entrée. Le principe de l'algorithme permettant d'estimer la LRF est décrit dans l'Algorithme 5.1. L'avantage des LRF est qu'elles peuvent également être calculées pour le schéma de Tiedtke-Bechtold, dans ARPEGE-Climat.

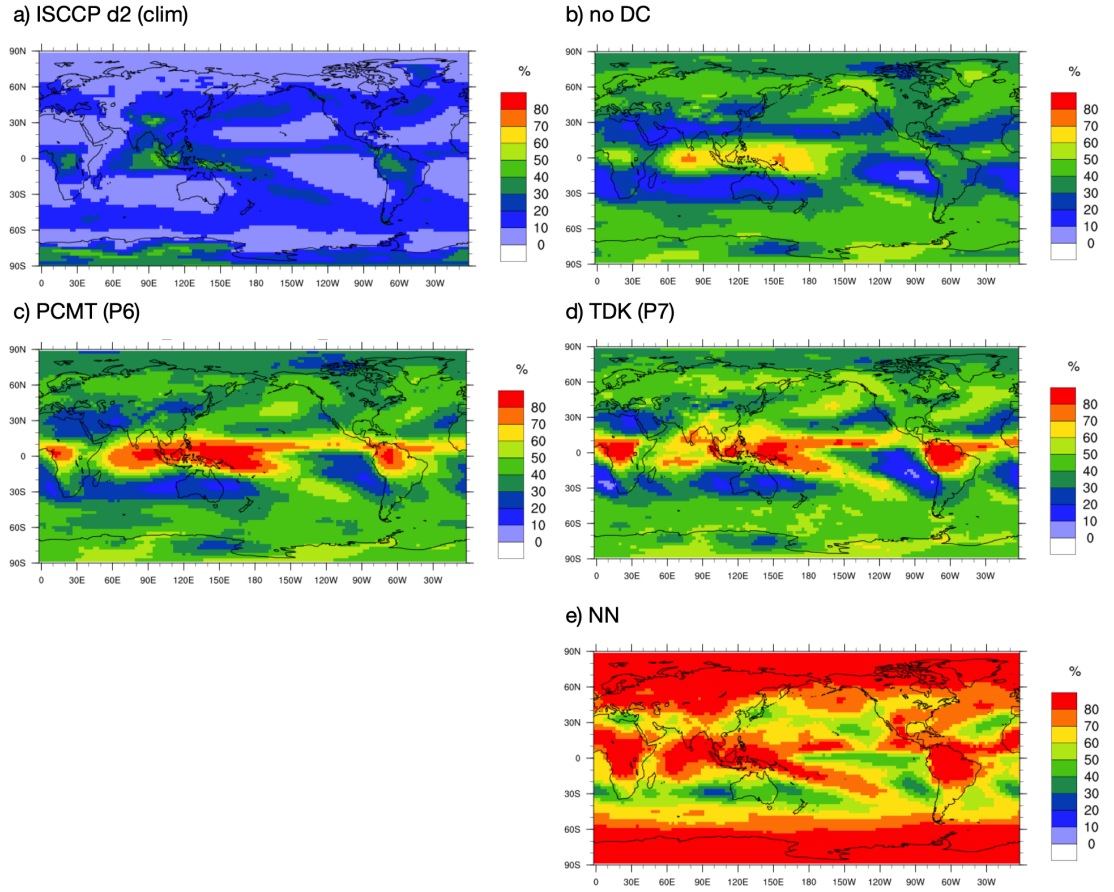


Figure 5.8: *Moyennes annuelles de couverture nuageuse à l'étage haut (CLTH) calculées à partir de plusieurs jeux de données. Les figures sont issues de l'atlas. a) Climatologie des couvertures nuageuses annuelles à l'étage haut, représentées à partir du jeu de données ISCCP d2. b) Moyenne annuelle des couvertures nuageuses à l'étage haut, calculées à partir d'une simulation ARPEGE-Climat de 5 ans sans schéma de convection profonde. c) Moyenne annuelle des couvertures nuageuses à l'étage haut, calculées à partir d'une simulation ARPEGE-Climat avec le schéma de convection profonde de la version précédente. d) Moyenne annuelle des couvertures nuageuses à l'étage haut, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec le schéma de convection profonde actuelle (Tiedtke-Bechtold). e) Moyenne annuelle des couvertures nuageuses à l'étage haut, calculées à partir d'une simulation ARPEGE-Climat de 5 ans avec la schéma de convection profonde appris par le NN.*

5.6.2 Analyse de sensibilité du schéma de Tiedtke-Bechtold

Les variables (pronostiques) en entrée du schéma de Tiedtke-Bechtold sont : les profils verticaux d'énergie statique sèche s et d'humidité totale q_T, ainsi que les flux de surface de chaleur latente LHF et sensible SHF. Les variables en sortie du schéma sont les profils verticaux de flux convectives d'énergie statique (dT), d'humidité totale (dq) et de précipitations (d(precip)). Le schéma renvoie également des flux pour les profils verticaux de quantité de mouvement, qui seront ignorés dans la suite (car

Algorithme 5.1 : linear response function (LRF)

Initialisation. Soit $f : \mathbb{R}^p \to \mathbb{R}^q$ la fonction dont on souhaite estimer la sensibilité sur une collection $\{\mathbf{x}_i\}_{1 \leq i \leq N}$. On note $\delta\mathbf{x} = (\delta x^1, \delta x^2, ..., \delta x^p)$ une (petite) perturbation de $\mathbf{x}$.

for $j = 1, ..., p$,do;
  for $i = 1, .., N$, do;
  1. Perturber x^l : $\mathbf{x}_i^{\text{pert}} = (x_i^1, x_i^2, ..., x_i^j + \delta x^j, ..., x_i^p)$.
  2. Estimer la sensibilité ∇f_i^j : $\nabla f_i^j \approx \dfrac{f(\mathbf{x}_i) - f(\mathbf{x}_i^{\text{pert}})}{\delta x^j}$.
  end do.

Estimer ∇f^j : $\nabla f^j \approx \dfrac{1}{N} \sum_{i=1}^{N} \nabla f_i^j$.

end do.

La valeur finale de la sensibilité de f est $\nabla f = (\nabla f^1 \ \nabla f^2 \ ... \nabla f^p)$.

pas appris par le NN).

Pour le calcul des LRF du schéma de Tiedtke-Bechtold, nous avons utilisé les perturbations suivantes :

1. Énergie statique sèche s : $\delta s = c_p \times \delta T$, avec $c_p = 1004$ J.kg^{-1}.K^{-1} la capacité thermique massique isobare de l'air et $\delta T = 1$ K.

2. Humidité q_T : $\delta q_T = q_T$ tel que $\frac{P}{P_{sat}(T+\delta T)} = \frac{P}{P_{sat}(T)}$ avec $\delta T = 1$ K, P la pression et $P_{sat}(T)$ la pression de vapeur saturante à T.

3. Flux de surface LHF et SHF : 110% de la valeur moyenne.

Dans son implémentation dans ARPEGE-Climat, la LRF du schéma de Tiedtke-Bechtold met en évidence une forte réaction du schéma aux variables décrivant les basses-couches, en particulier l'humidité q_T (Figure 5.9). Cela n'est pas surprenant pour un schéma de convection profonde : l'humidité atmosphérique, une des sources d'alimentation des systèmes convectifs, est essentiellement contenue dans la couche limite atmosphérique (de la surface jusqu'à environ 750 hPa ou 3000m d'altitude) et les quelques niveaux au-dessus.

Si l'émulateur NN est associé à une carte de sensibilité proche de celle présentée pour le schéma de Tiedtke-Bechtold, cela nous permet de renforcer la confiance que l'on accorde au NN. En d'autres termes, retrouver des LRF proches pour le NN et le schéma permettrait de mettre en évidence que le NN a appris les relations

physiques que nous souhaitions lui faire apprendre. Néanmoins, si la LRF du schéma de Tiedtke-Bechtold et de l'émulateur NN sont (très) différentes, cela ne signifie pas nécessairement que le NN ne sera pas performant une fois implémenté dans le GCM ; si le NN n'est pas assez performant, les LRF peuvent en outre nous orienter sur la création d'un nouvel échantillon d'apprentissage (e.g., utilisation d'autres normes, exclusion des variables peu utilisés par le NN).

5.6.3 Analyse de sensibilité du NN

Les LRF peuvent également être calculées pour l'émulateur NN, de la même manière que pour le schéma de Tiedtke-Bechtold. Les résultats du calcul des LRF sur l'émulateur NN sont représentées sur la Figure 5.10.

Tout d'abord, on constate que la sensibilité de l'émulateur NN est moins importante que celle du schéma d'origine, d'un facteur 10 environ. En effet, les données issues du schéma de Tiedtke-Bechtold sont assez bruitées, ce que le réseau de neurones apprend sous une forme lissée. Si l'on s'intéresse de manière plus qualitative aux structures qui apparaissent sur les LRF, on constate plusieurs choses. Tout d'abord, on retrouve le même type de dépendance des différents flux en sortie à l'humidité entre l'émulateur et le schéma de Tiedtke-Bechtold. Dans les deux cas, l'humidité contenue dans les basses couches seulement est influente sur les sorties. Comme nous l'avons précédemment évoqué, l'humidité atmosphérique est essentiellement contenue dans les basses couches, ainsi, la réponse de l'émulateur paraît physiquement cohérente. De plus, le signe des LRF est identique pour le schéma de Tiedtke-Bechtold et l'émulateur NN : l'humidité affecte de manière positive les précipitations, négativement les flux d'humidité, et positvement les flux d'énergie statique sèche au-delà de 850 hPa. L'influence de l'humidité sur les flux d'énergie statique sèche est légèrement négative dans les plus basses couches de l'atmosphère pour les deux schémas de convection profonde.

Néanmoins, l'influence des profils verticaux de température sur les différents flux est moins bien apprise par le NN. Dans les basses couches, on retrouve la même forme d'influence de la température sur les profils verticaux de flux sur les LRF calculées pour l'émulateur, de manière très peu marqué. Mais surtout, on voit apparaître des zones de forte influence (relative) de la température sur les différents flux en moyenne troposphère. Le signe de ces paramètres paraît cohérent pour les flux d'énergie statique sèche et d'humidité totale. Le signe des LRF est inversé dans cette zone pour les flux de précipitations.

Une deuxième différence notable dans les LRF calculées sur les deux schémas apparaît sur les flux de surface, en entrée des différents schémas. Les deux colonnes représentent, de gauche à droite, les flux de surface de chaleur latente (LHF) et de chaleur sensible (SHF). Le NN est davantage sensible à ces quantités que le schéma d'origine, notamment aux flux de chaleur sensible de surface. Qualitativement, on peut remarquer que la sensibilité du NN à la SHF est proche de celle du schéma de Tiedtke-Bechtold à $s(z)$ dans les basses couches. Par exemple, $s(z)$ dans les

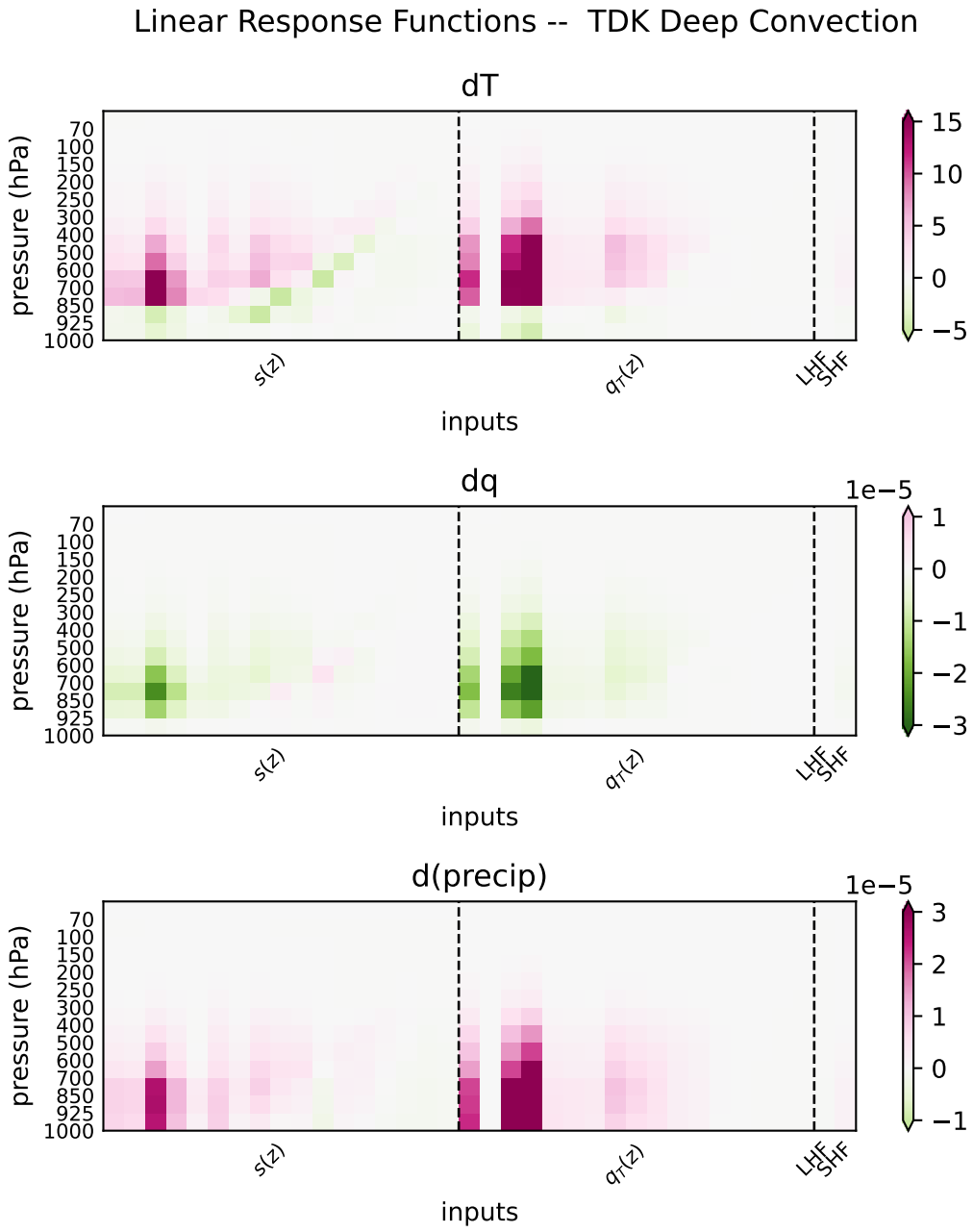


Figure 5.9: *Linear Response Function du schéma de Tiedtke-Bechtold, implémenté dans ARPEGE-Climat. Les différentes variables en entrée du schéma sont les profils verticaux de température T et d'humidité totale q_T et les flux de surface de chaleur latente (LHF) et sensible (SHF). Le schéma donne les profils verticaux de flux d'énergie statique sèche convectif dT, d'humidification convective dq et d'humidité précipitante convective d(precip). On constate que, principalement, les variables décrivant l'état thermodynamique de la couche limite sont celles qui conditionnent le plus fort le schéma.*

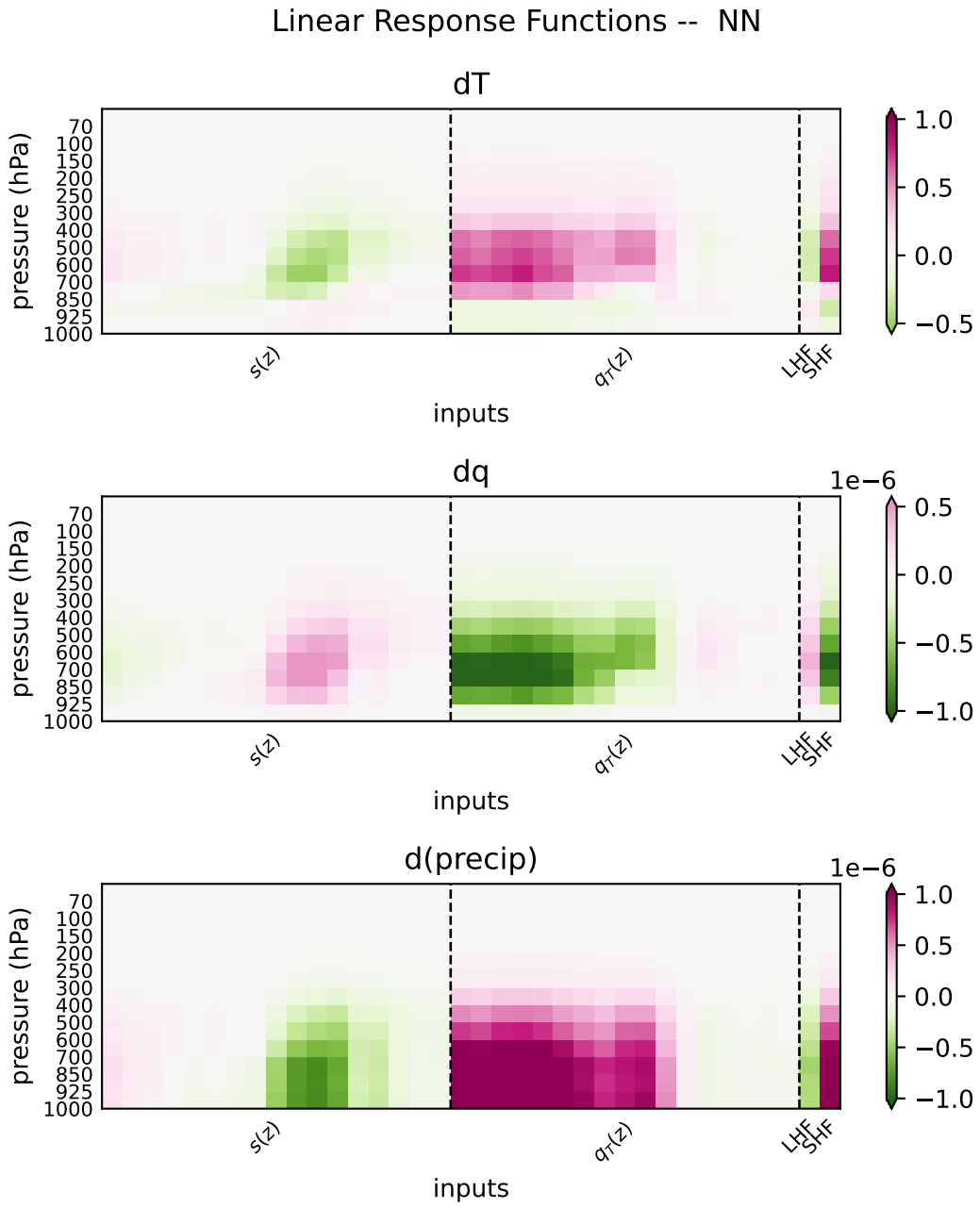


Figure 5.10: *Linear Response Function (LRF) calculé sur le schéma de Tiedtke-Bechtold appris par un NN. Les différentes variables en entrée du schéma sont les profils verticaux de température T et d'humidité totale q_T et les flux de surface de chaleur latente (LHF) et sensible (SHF). Pour chaque variable représentée sur l'axe des abscisses, les niveaux verticaux sont croissants vers la drotie (i.e., la description des basses couches se situe à gauche). La réponse linéaire du schéma appris à la température T et à l'humidité q_T est assez proche de celle du schéma de Tiedtke-Bechtold dans les basses couches.*

basses couches a une influence négative sur dT jusqu'à 850 hPa, puis une (forte) influence positive entre 850 et 500 hPa. Le NN réagit de façon similaire à la SHF : les variations de celle-ci ont une influence négative sur dT jusqu'à 850 hPa et une influence positive sur dT entre 700 et 500 hPa. Or, la chaleur sensible de surface et l'énergie statique dans les basses couches jouent des rôles similaires et leurs variations sont fortement corrélées. Le NN a donc possiblement appris une dépendance à la SHF seulement, plutôt qu'à $s(z)$ dans les basses couches. Cela peut être vu comme un "raccourci" qui peut être physiquement cohérent.

5.7 Conclusion du chapitre

Dans cette section, nous avons utilisé un NN pour émuler le schéma de convection profonde utilisé dans la version la plus récente d'ARPEGE-Climat (schéma de Tiedtke-Bechtold). Lors de la validation *offline*, nous avons montré que l'émulateur parvient à représenter la plupart des structures de grande échelle (e.g., les cellules de Hadley, l'ITCZ).

Le premier résultat est l'implémentation, avec succès, du schéma de convection profonde appris par un NN, dans ARPEGE-Climat. La validation *online*, durant laquelle le schéma appris a été branché dans le modèle ARPEGE-Climat en remplacement du schéma de Tiedtke-Bechtold, est stable : nous avons réalisé une simulation d'une durée de quelques années sans que le modèle résultant diverge. Cette stabilité est favorisée par le fait que nous ayons remplacé seulement une partie des paramétrisations physiques d'ARPEGE-Climat.

Le second résultat concerne la performance *online* de la physique apprise. On constate un bon accord global entre une simulation réalisée avec le schéma d'origine et la simulation avec la physique apprise, notamment pour le champ des précipitations. Néanmoins, la représentation des nuages, notamment à l'étage haut, n'est pas satisfaisante. Bien que la couverture nuageuse soit à l'origine d'un biais non négligeable pour les simulations réalisées avec les deux paramétrisations physiques présentées, l'utilisation de la physique apprise conduit à un biais encore plus prononcé.

Enfin, nous avons réalisé une analyse de sensibilité sur le schéma d'origine et le schéma appris par le NN. Pour cela, nous avons calculé les réponses linéaires (LRF). L'analyse des LRF du schéma de Tiedtke-Bechtold et de l'émulateur NN montrent un bon accord global, à l'exception de la réponse à des perturbations de l'énergie statique sèche dans les basses couches. Mais nous avons vu que le NN semble avoir appris une autre relation physiquement cohérente à la place de celle avec l'énergie statique sèche dans les basses couches.

L'analyse de sensibilité peut fournir des informations précieuses pour l'interprétation des NN, en particulier lorsque l'on peut comparer les prévisions à une référence. Dans notre cas, la référence était le schéma de Tiedtke-Bechtold, qu'il est possible de perturber pour calculer les réponses linéaires. Lorsque l'on apprend une paramétrisation à partir de données agrégées, avoir une référence pour l'interprétation de la sensi-

bilité du NN est plus difficile, car nous ne disposons pas directement d'une fonction de référence dont on peut perturber facilement les variables en entrée.

L'apprentissage du schéma de Tiedtke-Bechtold constitue un pas supplémentaire entre nos études réalisées sur des modèles jouets et l'apprentissage d'une paramétrisation physique à partir d'un jeu de données haute-résolution pour un GCM. Notre but n'était pas l'amélioration ou l'accélération du schéma à apprendre, mais de réaliser des essais préliminaires dans un contexte plus général que celui des modèles jouets (e.g., tester l'implémentation du schéma appris dans ARPEGE-Climat, utiliser un jeu de données plus volumineux).

Le modèle étant stable, cette nouvelle expérience ne pourra pas être utilisée comme cadre pour l'utilisation de la méthode de stabilité, décrite dans le Chapitre 3. En revanche, nous pourrons essayer l'utilisation de la méthode de calibration détaillée dans le Chapitre 4 dans un cadre plus réaliste.

Conclusion et perspectives

Depuis très récemment, la puissance de calcul disponible permet de réaliser des simulations de quelques mois avec des modèles atmosphériques à une résolution de l'ordre du kilomètre. À cette résolution, une partie des processus de fine échelle pour les modèles de circulation générale (GCM) peut être explicitement résolue, ce qui permet de réduire les incertitudes introduites dans le modèle par l'utilisation de paramétrisations. Les données issues de telles simulations peuvent être utilisées pour améliorer la précision de la représentation des processus de fine échelle dans les GCM. Pour cela, on peut utiliser des algorithmes de machine learning (ML) qui apprennent, à partir des données haute-résolution, la représentation de certains ou de la totalité des processus sous-maille (e.g., Brenowitz and Bretherton, 2018; Gentine et al., 2018; Rasp et al., 2018). Avec l'utilisation de la physique apprise, les premières études montrent une réduction de certains biais bien connus dans les modèles de climat, comme par exemple la double ITCZ (e.g., Oueslati and Bellon, 2015). Néanmoins, plusieurs problèmes limitent l'utilisation des physiques apprises dans les GCM. L'analyse de certains d'entre eux fait l'objet de travaux durant cette thèse.

Le premier problème concerne la stabilité du GCM une fois que la physique apprise a été branchée (Brenowitz and Bretherton, 2019; Brenowitz et al., 2020a). Pour comprendre l'origine des instabilités observées, une idée séduisante consiste à les reproduire dans des modèles jouets. Toutefois, Rasp (2020) montre que les modèles de Lorenz sont trop simples pour reproduire les instabilités que l'on observe en branchant la physique apprise dans un GCM. Nous avons donc développé un modèle jouet à partir du modèle Lorenz'63 (Lorenz, 1963), permettant l'étude des problèmes de stabilité (Balogh et al., 2021). Ce nouveau modèle, appelé "embedded" Lorezn'63 (eL63), est obtenu en plongeant les attracteurs tridimensionnels du modèle Lorenz'63 dans un espace de dimension $d > 3$. Lorsqu'un réseau de neurones (NN) apprend la "physique" à partir d'une série temporelle, appelée orbite, d'eL63, on constate que le modèle dynamique avec la physique apprise diverge, et ce d'autant plus que la dimension d augmente. Cette instabilité peut s'expliquer par l'échantillon d'apprentissage qui est *dégénéré*. En effet, la structure des attracteurs d'eL63 est localement quasi-plane. Une fois la physique NN branchée, toute (petite) erreur conduit la variable d'état dans des régions en dehors du plan des attracteurs, dans des régions qui n'ont pas été visitées pendant l'apprentissage et où la physique apprise n'est pas généralisable.

Pour résoudre le problème d'instabilité, nous avons proposé de réaliser un deuxième apprentissage, avec des données générées dans une région englobant les attracteurs. L'échantillon ainsi obtenu permet de lever la dégénérescence observée dans l'échantillon d'apprentissage précédent, et l'utilisation de la physique apprise conduit à un système dynamique stable.

Le deuxième problème concerne la performance de la physique apprise, une fois que celle-ci a été branchée dans un GCM. Disposer d'une physique apprise stable une fois branchée dans un GCM est essentiel, mais loin d'être suffisant pour garantir la performance du modèle obtenu, mesurée par sa capacité à simuler une climatologie proche de celle observée. En particulier, l'utilisation d'une paramétrisation (physique ou apprise) peut conduire à des biais dans le climat moyen. Dans le cas des paramétrisations physiques traditionnelles, le climat moyen du modèle est réglé en calibrant la valeur des paramètres au moyen desquels elles sont définies. Du fait du très grand nombre de paramètres en jeu, réaliser une telle calibration est, dans la pratique, impossible pour les physiques apprises. Dès lors, nous n'avons aucune garantie qu'une physique apprise performante sur un échantillon de validation (*offline*) sera performante une fois branchée dans un modèle de climat (*online*).

Nous avons proposé une méthode permettant d'obtenir une physique apprise dont la valeur de certains paramètres peut être optimisée *online* (Balogh et al., 2022). Cette méthode consiste à identifier des paramètres $\boldsymbol{\theta}$ pilotant la physique à apprendre et à échantillonner ces paramètres en plus de la variable d'état pour créer les données d'apprentissage. La physique sera ainsi apprise en fonction de la variable d'état, mais aussi en fonction de $\boldsymbol{\theta}$. La valeur de $\boldsymbol{\theta}$ pourra être optimisée dans un second temps en fonction d'une métrique *online* m, une fois que la physique a été branchée dans le système dynamique. Typiquement, la métrique m pourra pénaliser les écarts par rapport aux caractéristiques statistiques de la climatologie (e.g., moyenne, écart-type), calculées sur une longue série temporelle générée en utilisant la physique apprise. L'application de cette méthode nous a permis d'obtenir de très bons résultats dans le cas de la calibration de modèles jouets.

Les deux études précédemment citées ont été menées sur des modèles jouets. Pour réaliser des expériences dans un cadre plus réaliste que celui des modèles de Lorenz, on utilise des NN pour l'émulation du schéma de convection profonde de la version la plus récente d'ARPEGE-Climat, appelé schéma de Tiedtke (Tiedtke, 1993). Nous avons utilisé un NN implémentant uniquement des couches Dense pour émuler le schéma, à partir de données issues d'un an de simulation ARPEGE-Climat.

La validation *offline* a permis de montrer que le NN parvient à représenter la plupart des structures de grande échelle, comme par exemple l'ITCZ ou encore les cellules de Hadley. Le schéma appris a ensuite pu être utilisé dans ARPEGE-Climat, en remplacement de la paramétrisation physique de la convection profonde. Nous avons réalisé une simulation d'une durée de plusieurs années ; malgré l'utilisation du schéma appris par le NN, nous n'avons pas observé de divergence pendant toute la durée de la simulation. De plus, pour la plupart des variables, les résultats issus

de la simulation avec la physique apprise reste assez proche de la climatologie. La nébulosité, et particulièrement la nébulosité à l'étage haut (*cirrus*), n'est toutefois pas bien simulée.

Enfin, nous avons comparé les fonctions de réponse linéaire (*Linear Response Function*, LRF) du schéma de Tiedtke et le schéma appris par le NN. L'étude des LRF permet de comprendre la sensibilité des différentes variables en sortie à celles en entrée. Les deux cartes de LRF sont globalement proches, sauf pour l'énergie statique sèche dans les basses couches. Le NN semble ainsi avoir appris une grande partie des dépendances qui apparaissent dans le schéma de Tiedtke.

Durant la thèse, de nombreuses questions ont été soulevées et restent encore sans réponse. Tout d'abord, nous avons montré dans Balogh et al. (2021) que l'échantillon d'apprentissage constituée à partir d'une orbite peut être dégénéré., i.e., ne pas contenir une diversité de cas suffisante pour garantir une bonne généralisation. La plupart des paramétrisations physiques apprises actuelles utilisent des séries temporelles comme échantillon d'apprentissage, (i.e., des orbites). Dès lors, il n'y a aucune garantie sur la non-dégénérescence de l'échantillon d'apprentissage. Ce point pourrait être étudié dans le cas d'un GCM. Néanmoins, la réalisation de l'échantillonnage autour d'une orbite sera plus difficile que dans le cadre du modèle eL63. La difficulté principale réside dans la manière d'échantillonner la variable d'état dans le cas d'un GCM. En effet, les variables d'état sont *liées entre elles*, notamment dans les profils verticaux. Contrairement aux GCMs, le modèle eL63 ne décrit pas de structure verticale. Un échantillonnage aléatoire des variables décrivant l'état thermodynamique de l'atmosphère pourrait aboutir à des champs physiquement incohérents.

La méthode de calibration que nous avons développée et testée à l'aide des modèles jouets de Lorenz pourrait également être utilisée dans le cas de l'apprentissage d'une paramétrisation pour un GCM. Cela permettrait de régler la climatologie du modèle, de la même manière que pour les paramétrisations physiques. Il y a plusieurs difficultés à surmonter pour appliquer cette méthode. Tout d'abord, il est nécessaire de sélectionner les paramètres $\boldsymbol{\theta}$ qui seront inclus dans l'échantillon d'apprentissage. Ensuite, réaliser une simulation en perturbant la valeur des différents paramètres $\boldsymbol{\theta}$ va nécessairement augmenter le coût numérique et la complexité de la réalisation de la simulation. Le nombre de variables en entrée de la physique apprise sera nécessairement plus élevé que dans le cas où les paramètres $\boldsymbol{\theta}$ ne sont pas pris en compte. Toutefois, nous pensons que l'application de la méthode décrite pour L96 dans le cas d'un GCM peut se faire à faible coût numérique de la réalisation de la simulation.

Enfin, il est important de renforcer la fiabilité que nous accordons aux paramétrisations apprises. Apprendre la dépendance de la physique à certains paramètres $\boldsymbol{\theta}$, que l'on peut interpréter physiquement, est un premier pas vers l'interprétabilité. Une deuxième méthode permettant de renforcer le crédit que nous accordons aux physiques

apprises est l'implémentation de certaines équations de la physique, comme par exemple les lois de conservation (Beucler et al., 2021a). Néanmoins, bien que la physique apprise soit performante, nous ne pouvons garantir le fondement physique des relations apprises par l'algorithme de ML. L'utilisation de techniques d'interprétation des NN (e.g., McGovern et al., 2019; Toms et al., 2020), comme par exemple l'analyse de sensibilité (e.g., calcul des fonctions de réponse linéaire), nous permet de mieux comprendre le fonctionnement de l'émulateur. L'application d'une méthode d'analyse de sensibilité nous a permis de montrer que certaines relations physiques apparaissant entre variables en entrée et en sortie du schéma sont bien appris par l'émulateur NN.

Il convient de remarquer que les sensibilités du schéma de Tiedtke et de l'émulateur NN ont pu être comparés, car les variables en entrée du schéma de Tiedtke sont relativement faciles à perturber. Les réponses linéaires sont ainsi faciles à estimer. Dans le cas de l'apprentissage de la physique à partir de données haute-résolution, le calcul des réponses linéaires ne peut pas être directement effectuée pour vérifier si les dépendances qui apparaissent dans la physique apprise sont proches de celles de la physique à émuler. Bien que les LRF soient faciles à calculer pour la physique apprise, la formulation du problème (i.e., apprentissage depuis des données agrégées issues d'une simulation haute-résolution) ne permet pas une application directe de la méthode pour déterminer les sensibilités de référence. Ce problème est abordé par l'analyse de sensibilité, dont il faudra s'inspirer par la suite pour mieux comprendre le fonctionnement de la physique apprise.

Dans un futur proche, nous envisageons d'implémenter la méthode de calibration développée sur les modèles jouets, dans le cadre du schéma de Tiedtke. Les schémas de convection profonde disposent, en effet, de plusieurs paramètres que l'on pourrait inclure dans l'échantillon d'apprentissage. L'apprentissage et la calibration *online* du schéma de Tiedtke constitue un cadre plus réaliste pour cette étude que celle des modèles jouet.

Actuellement, les techniques d'IA connaissent des progrès fulgurants. Afin de les utiliser au mieux, il faudra suivre, avec attention, les progrès, y compris dans des domaines différents de l'apprentissage des paramétrisations dans des modèles de climat. On pourra s'inspirer des nouvelles méthodes et approches pour construire des paramétrisations physiques à la fois performantes et faciles à interpréter.

Bibliographie

R. F. Adler, G. J. Huffman, A. Chang, R. Ferraro, P.-P. Xie, J. Janowiak, B. Rudolf, U. Schneider, S. Curtis, D. Bolvin, A. Gruber, J. Susskind, P. Arkin, and E. Nelkin. The version-2 global precipitation climatology project (gpcp) monthly precipitation analysis (1979–present). *Journal of Hydrometeorology*, 4(6):1147 – 1167, 2003. doi: 10.1175/1525-7541(2003)004⟨1147:TVGPCP⟩2.0.CO;2. URL https://journals.ametsoc.org/view/journals/hydr/4/6/1525-7541_2003_004_1147_tvgpcp_2_0_co_2.xml.

I. AR5. *Climate Change 2013 : the Physical Science Basis*. Cambridge University Press, 2014. doi: https://doi.org/10.1017/CBO9781107415324.

H. M. Arnold, I. M. Moroz, and T. N. Palmer. Stochastic parametrizations and model uncertainty in the lorenz '96 system. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1991): 20110479, 2013. doi: 10.1098/rsta.2011.0479. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2011.0479.

B. Balogh, D. Saint-Martin, and A. Ribes. A Toy Model to Investigate Stability of AI-Based Dynamical Systems. *Geophysical Research Letters*, 48(8), Apr. 2021. doi: 10.1029/2020GL092133. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL092133.

B. Balogh, D. Saint-Martin, and A. Ribes. How to Calibrate a Dynamical System With Neural Network Based Physics? *Geophysical Research Letters*, 49 (8):e2022GL097872, 2022. ISSN 1944-8007. doi: 10.1029/2022GL097872. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2022GL097872. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2022GL097872.

P. Bechtold. Atmospheric moist convection. 2017. URL https://www.ecmwf.int/node/16953. ¡p¿ revised: 2019, 2020¡/p¿.

T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.*, 126:098302, Mar 2021a. doi: 10.1103/PhysRevLett.126.098302. URL https://link.aps.org/doi/10.1103/PhysRevLett.126.098302.

T. Beucler, M. Pritchard, J. Yuval, A. Gupta, L. Peng, S. Rasp, F. Ahmed, P. A. O'Gorman, J. D. Neelin, N. J. Lutsko, and P. Gentine. Climate-Invariant Machine

Learning. *arXiv:2112.08440 [physics]*, Dec. 2021b. URL http://arxiv.org/abs/2112.08440. arXiv: 2112.08440 version: 1.

L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, Aug. 1996. ISSN 1573-0565. doi: 10.1023/A:1018054314350. URL https://doi.org/10.1023/A:1018054314350.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Routledge, 1984. doi: Routledge.https://doi.org/10.1201/9781315139470.

N. D. Brenowitz and C. S. Bretherton. Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, 45(12):6289–6298, June 2018. ISSN 00948276. doi: 10.1029/2018GL078510. URL http://doi.wiley.com/10.1029/2018GL078510.

N. D. Brenowitz and C. S. Bretherton. Spatially extended tests of a neural network parametrization trained by coarse-graining. *Journal of Advances in Modeling Earth Systems*, 11(8):2728–2744, 2019. ISSN 1942-2466. doi: 10.1029/2019MS001711. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001711.

N. D. Brenowitz, T. Beucler, M. Pritchard, and C. S. Bretherton. Interpreting and stabilizing machine-learning parametrizations of convection. *Journal of the Atmospheric Sciences*, pages 1–55, 2020a. doi: 10.1175/JAS-D-20-0082.1. URL https://journals.ametsoc.org/jas/article/doi/10.1175/JAS-D-20-0082.1/354940/Interpreting-and-Stabilizing-Machine-learning.

N. D. Brenowitz, B. Henn, J. McGibbon, S. K. Clark, A. Kwa, W. A. Perkins, O. Watt-Meyer, and C. S. Bretherton. Machine learning climate model dynamics: offline versus online performance. *arXiv:2011.03081 [physics]*, Nov. 2020b. URL http://arxiv.org/abs/2011.03081.

C. S. Bretherton, B. Henn, A. Kwa, N. D. Brenowitz, O. Watt-Meyer, J. McGibbon, W. A. Perkins, S. K. Clark, and L. Harris. Correcting Coarse-Grid Weather and Climate Models by Machine Learning From Global Storm-Resolving Simulations. *Journal of Advances in Modeling Earth Systems*, 14(2):e2021MS002794, 2022. ISSN 1942-2466. doi: 10.1029/2021MS002794. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002794. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002794.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, Apr. 2016. ISSN 0027-8424,

1091-6490. doi: 10.1073/pnas.1517384113. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1517384113.

F. Caillez and J. Pagès. *Introduction à l'analyse des données*. smash Paris, 1976.

K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, Nov. 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1906995116.

M. Chantry, S. Hatfield, P. Dueben, I. Polichtchouk, and T. Palmer. Machine learning emulation of gravity wave drag in numerical weather forecasting. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002477, 2021. doi: https://doi.org/10.1029/2021MS002477. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002477. e2021MS002477 2021MS002477.

A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian. Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, July 2020. ISSN 1607-7946. doi: 10.5194/npg-27-373-2020. URL https://npg.copernicus.org/articles/27/373/2020/.

F. Chevallier, F. Chéruy, N. A. Scott, and A. Chédin. A neural network approach for a fast and accurate computation of a longwave radiative budget. *Journal of Applied Meteorology*, 37:1385–1397, 1998.

F. Chollet. *Deep Learning with Python*. Manning, Nov. 2017. ISBN 9781617294433.

S. K. Clark, N. D. Brenowitz, B. Henn, A. Kwa, J. McGibbon, W. A. Perkins, O. Watt-Meyer, C. S. Bretherton, and L. M. Harris. Correcting a 200 km Resolution Climate Model in Multiple Climates by Machine Learning From 25 km Resolution Simulations. *Journal of Advances in Modeling Earth Systems*, 14(9):e2022MS003219, 2022. ISSN 1942-2466. doi: 10.1029/2022MS003219. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003219. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003219.

E. Cleary, A. Garbuno-Inigo, S. Lan, T. Schneider, and A. M. Stuart. Calibrate, emulate, sample. *Journal of Computational Physics*, 424:109716, Jan. 2021. ISSN 00219991. doi: 10.1016/j.jcp.2020.109716. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999120304903.

F. Couvreux, F. Hourdin, D. Williamson, R. Roehrig, V. Volodina, N. Villefranque, C. Rio, O. Audouin, J. Salter, E. Bazile, F. Brient, F. Favot, R. Honnert, M.-P. Lefebvre, J.-B. Madeleine, Q. Rodier, and W. Xu. Process-based climate model development harnessing machine learning: I. a calibration tool for parameterization improvement. *Journal of Advances in Modeling Earth Systems*, 13(3):

e2020MS002217, 2021. doi: https://doi.org/10.1029/2020MS002217. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002217. e2020MS002217 2020MS002217.

N. Cressie. *Statistics for Spatial Data*. Wiley Online Library, 1992. ISBN 978-1-119-11461-1.

P. D. Dueben and P. Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11 (10):3999–4009, 2018. doi: 10.5194/gmd-11-3999-2018. URL https://gmd.copernicus.org/articles/11/3999/2018/.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37, Mar. 1996. doi: 10.1609/aimag.v17i3.1230. URL https://ojs.aaai.org/index.php/aimagazine/article/view/1230.

R. Fletcher. *Practical Methods of Optimization*. Wiley Online Library, 2013. ISBN 978-1-118-72318-0.

D. J. Gagne, H. M. Christensen, A. C. Subramanian, and A. H. Monahan. Machine learning for stochastic parameterization: generative adversarial networks in the Lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, 12(3): e2019MS001896, 2020. ISSN 1942-2466. doi: 10.1029/2019MS001896. URL http://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001896.

P. Gentine, M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, June 2018. ISSN 00948276. doi: 10.1029/2018GL078202. URL http://doi.wiley.com/10.1029/2018GL078202.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. *arXiv:1906.01563 [cs]*, Sept. 2019. URL http://arxiv.org/abs/1906.01563.

J. Guérémy. A continuous buoyancy based convection scheme: one-and three-dimensional validation. *Tellus A: Dynamic Meteorology and Oceanography*, 63 (4):687–706, 2011. doi: 10.1111/j.1600-0870.2011.00521.x. URL https://doi.org/10.1111/j.1600-0870.2011.00521.x.

Y.-G. Ham, J.-H. Kim, and J.-J. Luo. Deep learning for multi-year ENSO forecasts. *Nature*, 573(7775):568–572, Sept. 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1559-7. URL https://doi.org/10.1038/s41586-019-1559-7.

I. Harris, P. Jones, T. Osborn, and D. Lister. Updated high-resolution grids of monthly climatic observations – the cru ts3.10 dataset. *International Journal of Climatology*, 34(3):623–642, 2014. doi: https://doi.org/10.1002/joc.3711. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/joc.3711.

D. O. Hebb. *The organization of behavior; a neuropsychological theory*. The organization of behavior; a neuropsychological theory. Wiley, Oxford, England, 1949. Pages: xix, 335.

K. Hornik, M. B. Stinchcombe, and H. L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

F. Hourdin, T. Mauritsen, A. Gettelman, J.-C. Golaz, V. Balaji, Q. Duan, D. Folini, D. Ji, D. Klocke, Y. Qian, F. Rauser, C. Rio, L. Tomassini, M. Watanabe, and D. Williamson. The Art and Science of Climate Model Tuning. *Bulletin of the American Meteorological Society*, 98(3):589–602, Mar. 2017. ISSN 0003-0007, 1520-0477. doi: 10.1175/BAMS-D-15-00135.1. URL `http://journals.ametsoc.org/doi/10.1175/BAMS-D-15-00135.1`.

IPCC. *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2022.

C. Jakob. Accelerating progress in global atmospheric model development through improved parameterizations: Challenges, opportunities, and strategies. *Bulletin of the American Meteorological Society*, 91(7):869 – 876, 2010. doi: 10.1175/2009BAMS2898.1. URL `https://journals.ametsoc.org/view/journals/bams/91/7/2009bams2898_1.xml`.

E. Kim and R. Brunner. Star-galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 464, 08 2016. doi: 10.1093/mnras/stw2672.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. Technical Report arXiv:1412.6980, arXiv, Jan. 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv:1412.6980 [cs] type: article.

R. Kline. Cybernetics, Automata Studies, and the Dartmouth Conference on Artificial Intelligence. *IEEE Annals of the History of Computing*, 33(4):5–16, Apr. 2011. ISSN 1934-1547. doi: 10.1109/MAHC.2010.44. Conference Name: IEEE Annals of the History of Computing.

V. Krasnopolsky, M. Fox-Rabinovitz, and A. Belochitski. Decadal climate simulations using accurate and fast neural network emulation of full, longwave and shortwave, radiation. *Monthly Weather Review - MON WEATHER REV*, 136, 10 2008. doi: 10.1175/2008MWR2385.1.

V. M. Krasnopolsky, M. S. Fox-Rabinovitz, and D. V. Chalikov. New Approach to Calculation of Atmospheric Model Physics: Accurate and Fast Neural Network Emulation of Longwave Radiation in a Climate Model. *Monthly Weather Review*, 133(5):1370–1383, May 2005. ISSN 1520-0493, 0027-0644. doi: 10.1175/MWR2923.1. URL `https://journals.ametsoc.org/view/journals/mwre/133/5/`

mwr2923.1.xml. Publisher: American Meteorological Society Section: Monthly Weather Review.

V. M. Krasnopolsky, M. S. Fox-Rabinovitz, Y. T. Hou, S. J. Lord, and A. A. Belochitski. Accurate and Fast Neural Network Emulations of Model Radiation for the NCEP Coupled Climate Forecast System: Climate Simulations and Seasonal Predictions. *Monthly Weather Review*, 138(5):1822–1842, May 2010. ISSN 1520-0493, 0027-0644. doi: 10.1175/2009MWR3149.1. URL https://journals.ametsoc.org/view/journals/mwre/138/5/2009mwr3149.1.xml. Publisher: American Meteorological Society Section: Monthly Weather Review.

V. M. Krasnopolsky, M. S. Fox-Rabinovitz, and A. A. Belochitski. Using Ensemble of Neural Networks to Learn Stochastic Convection Parameterizations for Climate and Numerical Weather Prediction Models from Data Simulated by a Cloud Resolving Model. *Advances in Artificial Neural Systems*, 2013: e485913, May 2013. ISSN 1687-7594. doi: 10.1155/2013/485913. URL https://www.hindawi.com/journals/aans/2013/485913/. Publisher: Hindawi.

Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. In F. F. Soulié and J. Hérault, editors, *Neurocomputing*, pages 303–318, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. ISBN 978-3-642-76153-9.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2.

E. N. Lorenz. Predictability: a problem partly solved. *Proc. ECMWF Seminar on Predictability*, Vol. I, Reading, United Kingdom, ECMWF:1–18, 1996.

J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Commun. ACM*, 3(4):184–195, apr 1960. ISSN 0001-0782. doi: 10.1145/367177.367199. URL https://doi.org/10.1145/367177.367199.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec. 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

A. McGovern, R. Lagerquist, D. J. Gagne, G. E. Jergensen, K. L. Elmore, C. R. Homeyer, and T. Smith. Making the black box more transparent: Understanding the physical implications of machine learning. *Bulletin of the American Meteorological Society*, 100(11):2175 – 2199, 2019. doi: 10.1175/BAMS-D-18-0195.1. URL https://journals.ametsoc.org/view/journals/bams/100/11/bams-d-18-0195.1.xml.

M. D. McKay. Latin Hypercube Sampling as a tool in uncertainty analysis of computer models. *Proceedings of the 24th Conference on Winter Simulation*, pages 557–564, 1992. doi: https://doi.org/10.1145/167293.167637.

B. Medeiros and B. Stevens. Revealing differences in gcm representations of low clouds. *Climate dynamics*, 36(1-2):385–399, 2011. doi: 10.1007/s00382-009-0694-5.

B. Medeiros, B. Stevens, I. M. Held, M. Zhao, D. L. Williamson, J. G. Olson, and C. S. Bretherton. Aquaplanets, climate sensitivity, and low clouds. *Journal of Climate*, 21(19):4974 – 4991, 2008. doi: 10.1175/2008JCLI1995.1. URL `https://journals.ametsoc.org/view/journals/clim/21/19/2008jcli1995.1.xml`.

S. Moorthi and M. J. Suarez. Relaxed arakawa-schubert. a parameterization of moist convection for general circulation models. *Monthly Weather Review*, 120(6):978 – 1002, 1992. doi: 10.1175/1520-0493(1992)120⟨0978:RASAPO⟩2.0.CO;2. URL `https://journals.ametsoc.org/view/journals/mwre/120/6/1520-0493_1992_120_0978_rasapo_2_0_co_2.xml`.

A. Mounier, L. Raynaud, L. Rottner, M. Plu, P. Arbogast, M. Kreitz, L. Mignan, and B. Touzé. Detection of bow echoes in kilometer-scale forecasts using a convolutional neural network. *Artificial Intelligence for the Earth Systems*, pages 1 – 66, 2022. doi: 10.1175/AIES-D-21-0010.1. URL `https://journals.ametsoc.org/view/journals/aies/aop/AIES-D-21-0010.1/AIES-D-21-0010.1.xml`.

Z. E. Musielak and D. E. Musielak. High-dimensional chaos in dissipative and driven dynamical systems. *International Journal of Bifurcation and Chaos*, 19(09):2823–2869, Sept. 2009. ISSN 0218-1274, 1793-6551. doi: 10.1142/S0218127409024517. URL `https://www.worldscientific.com/doi/abs/10.1142/S0218127409024517`.

A. Newell, J. Shaw, and H. Simon. Empirical explorations of the logic theory machine: A case study in heuristic. *Proceedings of the Western Joint Computer Conference, Los Angeles*, page 218–230, 1957.

P. A. O'Gorman and J. G. Dwyer. Using machine learning to parameterize moist convection: potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, 10(10):2548–2563, 2018. ISSN 1942-2466. doi: 10.1029/2018MS001351. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001351`.

B. Oueslati and G. Bellon. The double itcz bias in cmip5 models: interaction between sst, large-scale circulation and precipitation. *Climate Dynamics*, 44:585–607, 02 2015. doi: 10.1007/s00382-015-2468-6.

T. N. Palmer. A nonlinear dynamical perspective on model error: A proposal for non-local stochastic-dynamic parametrization in weather and climate prediction

models. *Quarterly Journal of the Royal Meteorological Society*, 127(572):279–304, 2001. doi: https://doi.org/10.1002/qj.49712757202. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.49712757202.

J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, Dec. 2017. ISSN 1054-1500, 1089-7682. doi: 10.1063/1.5010300. URL http://aip.scitation.org/doi/10.1063/1.5010300.

A. Rakhlin, A. A. Shvets, V. I. Iglovikov, and A. A. Kalinin. Deep convolutional neural networks for breast cancer histology image analysis. *bioRxiv*, 2018.

S. Rasp. Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0). *Geoscientific Model Development*, 13(5):2185–2196, May 2020. ISSN 1991-9603. doi: 10.5194/gmd-13-2185-2020. URL https://gmd.copernicus.org/articles/13/2185/2020/.

S. Rasp, M. S. Pritchard, and P. Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115 (39):9684–9689, Sept. 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1810286115. URL http://www.pnas.org/content/115/39/9684.

R. Roehrig, I. Beau, D. Saint-Martin, A. Alias, B. Decharme, J.-F. Guérémy, A. Voldoire, A. Y. Abdel-Lathif, E. Bazile, S. Belamari, S. Blein, D. Bouniol, Y. Bouteloup, J. Cattiaux, F. Chauvin, M. Chevallier, J. Colin, H. Douville, P. Marquet, M. Michou, P. Nabat, T. Oudar, P. Peyrillé, J.-M. Piriou, D. Salas y Mélia, R. Séférian, and S. Sénési. The cnrm global atmosphere model arpege-climat 6.3: Description and evaluation. *Journal of Advances in Modeling Earth Systems*, 12(7):e2020MS002075, 2020. doi: https://doi.org/10.1029/2020MS002075. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002075. e2020MS002075 10.1029/2020MS002075.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471. doi: 10.1037/h0042519. Place: US Publisher: American Psychological Association.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0. Number: 6088 Publisher: Nature Publishing Group.

S. Scher and G. Messori. Generalization properties of feed-forward neural networks trained on Lorenz systems. *Nonlinear Processes in Geophysics*, 26(4):381–399,

Nov. 2019. ISSN 1607-7946. doi: 10.5194/npg-26-381-2019. URL https://npg.copernicus.org/articles/26/381/2019/.

G. A. Schmidt, D. Bader, L. J. Donner, G. S. Elsaesser, J.-C. Golaz, C. Hannay, A. Molod, R. B. Neale, and S. Saha. Practice and philosophy of climate model tuning across six US modeling centers. *Geosci. Model Dev.*, page 17, 2017.

T. Schneider, S. Lan, A. Stuart, and J. Teixeira. Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations: EARTH SYSTEM MODELING 2.0. *Geophysical Research Letters*, 44(24):12,396–12,417, Dec. 2017. ISSN 00948276. doi: 10.1002/2017GL076101. URL http://doi.wiley.com/10.1002/2017GL076101.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

B. Stevens and S. Bony. What are climate models missing? *Science*, 340(6136): 1053–1054, 2013. doi: 10.1126/science.1237554. URL https://www.science.org/doi/abs/10.1126/science.1237554.

B. Stevens, M. Satoh, L. Auger, J. Biercamp, C. S. Bretherton, X. Chen, P. Düben, F. Judt, M. Khairoutdinov, D. Klocke, C. Kodama, L. Kornblueh, S.-J. Lin, P. Neumann, W. M. Putman, N. Röber, R. Shibuya, B. Vanniere, P. L. Vidale, N. Wedi, and L. Zhou. DYAMOND: the DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains. *Progress in Earth and Planetary Science*, 6(1):61, Dec. 2019. ISSN 2197-4284. doi: 10.1186/s40645-019-0304-z. URL https://progearthplanetsci.springeropen.com/articles/10.1186/s40645-019-0304-z.

B. Tian and X. Dong. The double-itcz bias in cmip3, cmip5, and cmip6 models based on annual mean precipitation. *Geophysical Research Letters*, 47(8): e2020GL087232, 2020. doi: https://doi.org/10.1029/2020GL087232. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL087232. e2020GL087232 2020GL087232.

M. Tiedtke. Representation of clouds in large-scale models. *Monthly Weather Review*, 121(11):3040 – 3061, 1993. doi: 10.1175/1520-0493(1993)121⟨3040:ROCILS⟩2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/121/11/1520-0493_1993_121_3040_rocils_2_0_co_2.xml.

B. A. Toms, E. A. Barnes, and I. Ebert-Uphoff. Physically interpretable neural networks for the geosciences: Applications to earth system variability. *Journal of Advances in Modeling Earth Systems*, 12(9):e2019MS002002, 2020. doi: https://doi.org/10.1029/2019MS002002. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS002002. e2019MS002002 10.1029/2019MS002002.

A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, Oct. 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

J. Turkey. *Exploratory Data Analysis*. Reading, Mass. : Addison-Wesley Pub. Co., 1977.

V. Vapnik. The Nature of Statistical Learning Theory. In *Statistics for Engineering and Information Science*, volume 8, pages 1–15. Jan. 1998. ISBN 978-1-4419-3160-3. doi: 10.1007/978-1-4757-3264-1_1. Journal Abbreviation: Statistics for Engineering and Information Science.

A. Voldoire, D. Saint-Martin, S. Sénési, B. Decharme, A. Alias, M. Chevallier, J. Colin, J.-F. Guérémy, M. Michou, M.-P. Moine, P. Nabat, R. Roehrig, D. Salas y Mélia, R. Séférian, S. Valcke, I. Beau, S. Belamari, S. Berthet, C. Cassou, J. Cattiaux, J. Deshayes, H. Douville, C. Ethé, L. Franchistéguy, O. Geoffroy, C. Lévy, G. Madec, Y. Meurdesoif, R. Msadek, A. Ribes, E. Sanchez-Gomez, L. Terray, and R. Waldman. Evaluation of cmip6 deck experiments with cnrm-cm6-1. *Journal of Advances in Modeling Earth Systems*, 11(7):2177–2213, 2019. doi: https://doi.org/10.1029/2019MS001683. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001683.

X. Wang, Y. Han, W. Xue, G. Yang, and G. J. Zhang. Stable climate simulations using a realistic general circulation model with neural network parameterizations for atmospheric moist physics and radiation processes. *Geoscientific Model Development*, 15(9):3923–3940, May 2022. ISSN 1991-959X. doi: 10.5194/gmd-15-3923-2022. URL https://gmd.copernicus.org/articles/15/3923/2022/. Publisher: Copernicus GmbH.

P. A. G. Watson. Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction. *Journal of Advances in Modeling Earth Systems*, 11(5):1402–1417, 2019. ISSN 1942-2466. doi: 10.1029/2018MS001597. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001597.

J. A. Weyn, D. R. Durran, and R. Caruana. Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, 11(8):2680–2693, 2019. doi: https://doi.org/10.1029/2019MS001705. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001705.

J. A. Weyn, D. R. Durran, and R. Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9), Sep 2020. ISSN 1942-2466. doi: 10.1029/2020ms002109. URL http://dx.doi.org/10.1029/2020MS002109.

A. H. Young, K. R. Knapp, A. Inamdar, W. Hankins, and W. B. Rossow. The International Satellite Cloud Climatology Project H-Series climate data record product. *Earth System Science Data*, 10(1):583–593, 2018. doi: 10.5194/essd-10-583-2018. URL `https://essd.copernicus.org/articles/10/583/2018/`.

J. Yuval and P. A. O'Gorman. Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, 11(1):3295, July 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17142-3. URL `https://doi.org/10.1038/s41467-020-17142-3`.

J. Yuval, P. A. O'Gorman, and C. N. Hill. Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, 48(6): e2020GL091363, 2021. doi: https://doi.org/10.1029/2020GL091363. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL091363`. e2020GL091363 2020GL091363.

Annexes

Annexe A

Les équations primitives

Les équations primitives sont rappelées ci-dessous, en coordonnées pression.

A.1 Notations

Soit $\mathbf{x}$ une variable décrivant l'état thermodynamique de l'atmosphère. Dans la suite, on note :

$$\frac{\mathrm{D}\mathbf{x}}{\mathrm{D}t} = \frac{\partial \mathbf{x}}{\partial t} + u\frac{\partial \mathbf{x}}{\partial x} + v\frac{\partial \mathbf{x}}{\partial y} + \omega\frac{\partial \mathbf{x}}{\partial p},$$

avec u, v, respectivement, les composantes zonale et méridienne du vent en m/s, ω sa composante verticale en hPa, p la pression, x, y les coordonnées en longitude et latitude.

A.2 Dynamique

$$\frac{\mathrm{D}u}{\mathrm{D}t} - fv = -\frac{\partial \phi}{\partial x} + F_x \tag{A.1}$$

$$\frac{\mathrm{D}v}{\mathrm{D}t} + fu = -\frac{\partial \phi}{\partial y} + F_y, \tag{A.2}$$

avec ϕ le géopotentiel et F_x, F_y les composantes des forces de friction selon la longitude et la latitude respectivement.

A.3 Hydrostatique

$$\frac{\partial \Phi}{\partial P} = -\frac{1}{\rho} \tag{A.3}$$

A.4 Continuité

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial \omega}{\partial p} = 0. \tag{A.4}$$

A.5 Conservation de l'énergie

$$\frac{\mathrm{D}T}{\mathrm{D}t} = \frac{RT}{c_P P}\omega + \frac{1}{c_P}\left(\frac{\mathrm{d}Q}{\mathrm{d}t}\right) \tag{A.5}$$

Annexe B

Les forêts aléatoires (*Random Forests)*

B.1 Les arbres de décision

B.1.1 Principe

Les arbres de décision ont été formalisés dans Breiman et al. (1984), sous le nom *d'arbres de classification et de régression* (*classification and regression tree*, CART). Rapidement devenus populaires grâce à leur facilité d'utilisation et d'interprétation, les CART serviront de base pour la construction de forêts aléatoires.

Soit $\{\mathbf{x}_i, \mathbf{y}_i\}_{1 \leq i \leq N}$ un échantillon de N variables, avec $\mathbf{x} = (x_1, x_2, ..., x_p)$ et $\mathbf{y} = (y_1, y_2, ..., y_m)$. On cherche à construire un modèle statistique permettant d'estimer $\mathbf{y}_i$ en fonction de $\mathbf{x}_i$. Un CART est composé de *nœuds* κ, correspondant à une variable x_{p_0} et un critère de division D_κ. Dans la pratique, un critère de division correspond à une valeur seuil sur la variable x_{p_0}. Un CART se construit en itérant les critères de division à partir du *nœud initial*, jusqu'à ce qu'il n'y ait plus de division admissible (Fig. B.1). Lorsqu'il n'y a plus de critère de division admissible, cela signifie que le nœud est homogène ou bien que le nombre d'individus affectés au nœud est inférieur à une valeur minimale préalablement fixée. Les derniers nœuds forment les *feuilles* du CART. Dans le cas d'une régression, la valeur affectée à une feuille correspond à la moyenne des individus associées à elle. Dans le cas d'un CART de classification, il existe plusieurs choix pour la valeur $\widehat{\mathbf{y}}$ associée à chaque feuille, en fonction de l'objectif à atteindre (e.g., médiane, classe la plus représentée).

Une fois le CART ajusté, il peut être utilisé pour déterminer la valeur d'un $\mathbf{x}_0$. Celui-ci est d'abord affecté à une feuille, et la prévision du CART correspond à la valeur associée à la feuille où $\mathbf{x}_0$ a été affecté.

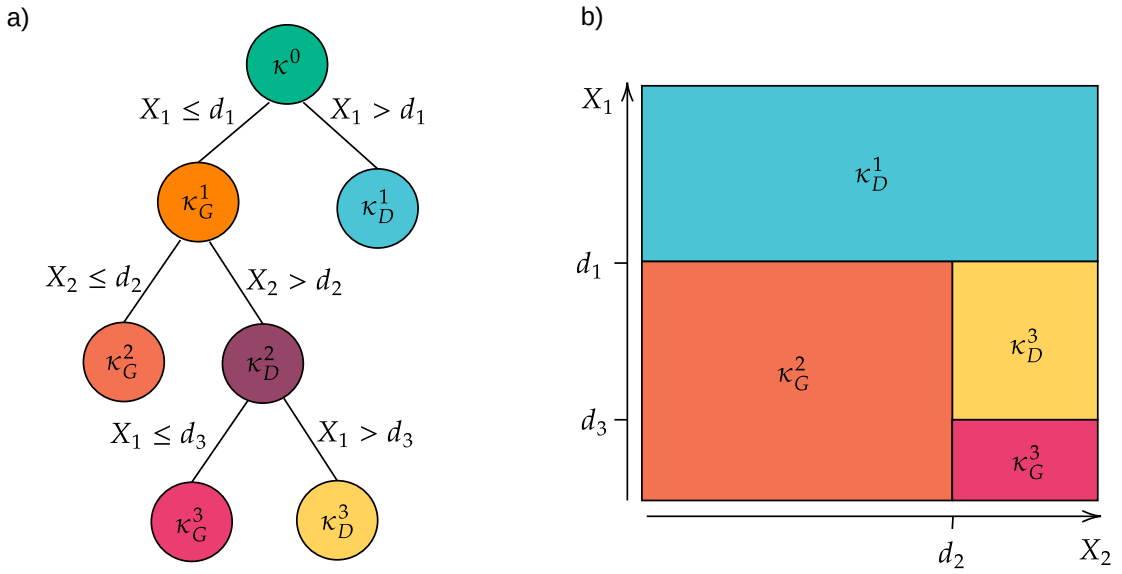


Figure B.1: *Exemple d'arbre de décision $\widehat{f}(X_1, X_2)$. a) Représentation de l'arbre. b) Pavage dyadique associé.*

B.1.2 Définition des nœuds

Dans la suite, nous allons nous intéresser au cas de la régression uniquement. L'hétérogénéité du nœud κ, notée D_κ, se définit alors par :

$$D_\kappa = \frac{1}{|\kappa|} \sum_{i \in \kappa} (y_i - \overline{y}_\kappa)^2, \tag{B.1}$$

avec $|\kappa|$ l'effectif du nœud κ. L'Eq. B.1 correspond à l'estimateur de la variance en κ. À partir d'un nœud, on détermine deux nœuds fils, κ_G (gauche) et κ_D (droite), en cherchant la division permettant de *maximiser* l'hétérogénéité entre κ_G et κ_D et de *minimiser* l'hétérogénéité au sein de chacun des deux nœuds κ_D et κ_G. Ce critère de division est itéré jusqu'à ce que l'on n'ait plus de division admissible. On obtient ainsi l'arbre *maximal*.

L'arbre *optimal* est obtenu à partir de l'arbre maximal en l'élaguant. Pour cela, on ajoute une pénalité sur le nombre de feuilles K_A. La pénalité se présente sous forme d'un coefficient γ, permettant de quantifier la complexité C de l'arbre A :

$$C(A) = \sum_{\kappa \in A} D_\kappa + \gamma K_A. \tag{B.2}$$

Dans la pratique, la valeur de γ est optimisée par validation croisée en V segements.

B.2 Agrégation d'arbres

B.2.1 Bagging

Avec un arbre A unique, bien qu'il soit élagué, la variance observée au niveau de chaque feuille peut être importante, ce qui peut augmenter les erreurs de prévision. En effet, la variance associée à chaque feuille $\kappa_A^{e,i}$ est estimée (sans biais) par

$$s^2 = \frac{1}{n-1} \sum_{\mathbf{y} \in \kappa_A^{e,i}} (\mathbf{y} - \overline{\mathbf{y}}^{e,i})^2 \tag{B.3}$$

avec n le nombre d'individus affectés à la feuille $\kappa_A^{e,i}$, de moyenne $\overline{\mathbf{y}}^{e,i}$. Pour un échantillon donné, la variance des feuilles peut être réduite en agrégeant plusieurs arbres, chacun ayant été ajusté à un sous-échantillon bootstrap différent, issu de l'échantillon d'apprentissage total disponible. Cet algorithme est appelé *bagging*, contraction de *bootstrap* et de *aggregating* (Breiman, 1996). L'algorithme est décrit dans l'encadré ci-dessous.

Il existe de nombreux paramètres à optimiser dans l'algorithme, notamment le nombre N_A d'arbres, mais aussi l'ensemble des paramètres à optimiser pour les CART, comme par exemple le nombre minimal d'individus sur chaque feuille ou encore la profondeur maximale des arbres. Dans la pratique, la valeur des paramètres permettant d'obtenir les meilleurs résultats est déterminée par validation croisée en V segments. On constate que les modèles performants sont souvent composés d'un grand nombre d'arbres dont la profondeur est limitée.

Algorithme : Bagging

Soit $[\mathbf{x}] = \{\mathbf{x}_n, \mathbf{y}_n\}_{1 \leq n \leq N}$ un échantillon.

for $i = 1, ..., N_A$, do;
  Tirer un échantillon bootstrap $[\mathbf{x}]_i$
  Estimer un CART $\widehat{f}_i$ sur cet échantillon.
end

Calculer la moyenne agrégée : $\widehat{f} = \frac{1}{N_A} \sum_{i=1}^{N_A} \widehat{f}_i$.

B.2.2 Forêts aléatoires

Le bagging a également ses défauts. En particulier, la seule différence entre deux CART agrégés du modèle est l'échantillon d'apprentissage bootstrap ayant été utilisée pour l'apprentissage. Les arbres agrégés peuvent alors présenter une forte auto-corrélation entre elles.

Pour résoudre ce problème, Breiman (2001) propose une amélioration de l'algorithme

de bagging. Une fois l'échantillon bootstrap tiré, un deuxième tirage aléatoire a lieu, pour sélectionner un sous-ensemble de q prédicteurs disponibles parmi p. Les CART apprennent alors à partir du sous-échantillon bootstrap, en utilisant seulement une partie des variables en entrée du modèle. L'encadré à la fin du paragraphe résume l'algorithme de manière schématique.

Comme pour le bagging, la valeur de nombreux paramètres peut être optimisé, notamment le nombre q de variables à tirer aléatoirement pour l'apprentissage des CART.

Algorithme : Random Forests

Soit $[\mathbf{x}] = \{\mathbf{x}_n, \mathbf{y}_n\}_{1 \leq n \leq N}$ un échantillon.

for $i = 1, ..., N_A$, do;
Tirer un échantillon bootstrap $[\mathbf{x}]_i$
Tirer un sous-ensemble de q prédicteurs parmi p.
Estimer un CART $\widehat{f}_i$ sur l'échantillon d'apprentissage, en utilisant seulement les q prédicteurs aléatoirement tirés.
end

Calculer la moyenne agrégée : $\widehat{f} = \frac{1}{N_A} \sum_{i=1}^{N_A} \widehat{f}_i$.

Annexe C

Démonstration des équations de rétropropagation de l'erreur

C.1 Démonstration de l'équation 1.4

Proof. Soit $1 \leq l \leq L$ le numéro d'une couche cachée d'un MLP. On s'intéresse à $\delta^{(l)} = (\delta_1^{(l)}, \delta_2^{(l)}, ..., \delta_{n_l}^{(l)})$ où n_l est le nombre de neurones sur la couche l. Pour tout $1 \leq i \leq n_l$, on peut écrire :

$$\begin{aligned}\delta_i^{(l)} &= \frac{\partial \mathcal{L}}{\partial h_i^{(l)}} \\ &= \sum_{k=1}^{n_l} \frac{\partial \mathcal{L}}{\partial h_k^{(l+1)}} \frac{\partial h_k^{(l+1)}}{\partial h_i^{(l)}} \\ &= \sum_{k=1}^{n_l} \delta_k^{(l+1)} \frac{\partial h_k^{(l+1)}}{\partial h_i^{(l)}}.\end{aligned}$$

Or, par définition de $\mathbf{h}$, on peut écrire, pour $1 \leq k \leq n_{l+1}$:

$$z_k^{(l+1)} = \sum_{i=1}^{n_l} w_{k,i}^{(l+1)} s_i^{(l)} + b_i^{(l)},$$

d'où, en différentiant :

$$\frac{\partial h_k^{(l+1)}}{\partial h_i(l)} = w_{k,i}^{(l+1)} a'(h_k^{(l)}).$$

Finalement, on obtient l'équation 1.4, en mettant les équations précédentes sous forme vectorielle :

$$\delta^{(l)} = \mathbf{w}^{(l+1)} \delta^{(l+1)} a'(\mathbf{h}^{(l)}).$$

□

C.2 Démonstration de l'équation 1.5

Proof. Soit $1 \leq l \leq L$ le numéro d'une couche cachée et $1 \leq i \leq n_l$ l'indice d'un neurone sur la couche l. Pour ce neurone, on différentie la fonction de coût $\mathcal{L}$ par rapport à un poids $w_{i,k}^{(l)}$, en utilisation l'équation 1.4 :

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{i,k}^{(l)}} &= \frac{\partial \mathcal{L}}{\partial h_i^{(l)}} \frac{\partial h_i^{(l)}}{\partial w_{i,k}^{(l)}} \\ &= \delta_i^{(l)} \frac{\partial h_i^{(l)}}{\partial w_{i,k}^{(l)}}.\end{aligned}$$

Or, par définition,

$$h_i^{(l)} = \sum_{k=1}^{n_l} w_{i,k}^{(l)} s_k^{(l-1)} b_i^{(l)}.$$

Ainsi, l'équation précédente peut se réécrire

$$\frac{\partial \mathcal{L}}{\partial w_{i,k}^{(l)}} = \delta_i^{(l)} a_k^{(l-1)},$$

ce qui, sous forme vectorielle, donne :

$$\frac{\partial \mathcal{L}}{\partial w^{(l)}} = \delta^{(l)} (a^{(l-1)})^T.$$

□

C.3 Démonstration de l'équation 1.6

Proof. Soit $1 \leq l \leq L$ le numéro d'une couche cachée. En différentiant la fonction de coût $\mathcal{L}$ par rapport au biais de la couche l, $\mathbf{b}^{(l)}$, on obtient, compte tenu de l'équation 1.4 :

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \\ &= \delta^{(l)} \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}}.\end{aligned}$$

Or, par définition

$$\mathbf{z}^{(l)} = \mathbf{w}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}.$$

En différentiant par rapport à $\mathbf{b}^{(l)}$, on obtient

$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = 1.$$

En combinant les deux équations précédemment obtenues, on retrouve l'équation 1.6 :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \times 1 = \delta^{(l)}.$$

□

TITRE EN ANGLAIS : Using Artificial Intelligence in a numerical climate model

---

RÉSUMÉ EN ANGLAIS

In a numerical climate model, fine-scale processes are represented by parameterizations. A new method to develop parameterizations consists in using Artificial Intelligence techniques, by learning the representation of these processes from data coming from high resolution simulations. But a number of problems remain to be solved before the learned parameterizations can be used in a climate model.

The objective of this thesis is to study some problems preventing the use of AI parameterizations in a climate model: numerical stability and online performance. These problems are studied through two studies on toy models. The results obtained are then placed in perspective, through the learning of a parameterization implemented in a climate model.