
Rapport de stage

**Apprentissage de caractéristiques hydrologiques
à partir de bases de données (in situ, modèles et satellitaires)**

Auteur :
MIKAEL EL OUAZZANI¹

Encadrants :
M. JÉRÔME MONNIER^{1 2}
M. KEVIN LARNIER^{1 2 3}

29 octobre 2020

1. INSA Toulouse, France
2. Institut de Mathématiques de Toulouse (IMT), France
3. CS Group, CS-SI, Space Business Unit, Toulouse, France

Table des matières

1	Contexte et objectif du stage	1
1.1	Présentation des entreprises	1
1.2	Contexte	1
1.3	Objectif	2
2	État de l'art	3
2.1	Le cycle de l'eau	3
2.2	État de l'art sur les mesures de la Terre par télédétection	4
2.3	Mesurer le débit des rivières	5
2.4	Comprendre les réseaux de neurones	9
3	Données et outils utilisés	16
3.1	Présentation et analyse statistique des jeux de données	16
3.2	Présentation de l'environnement de programmation	21
4	Méthodologie	23
4.1	Mise en place d'un ANN	23
4.2	Mise en place d'un LSTM	24
4.3	Évaluer efficacement les performances	24
5	Résultats et analyse	28
5.1	Résultats pour la modélisation via ANN	28
5.2	Résultats pour la modélisation via LSTM	36
5.3	Analyse des résultats	37
6	Conclusion	39
	Annexe A Notebook ANN	40
	Annexe B Notebook LSTM	45

Table des figures

1	Courbe de tarage (niveau de l'eau en fonction du débit). Les points représentent les mesures de hauteur d'eau et de débit, la courbe représente la courbe de tarage induite par ces mesures. [Braca, 2020]	6
3	Diagramme spatio-temporel des processus hydrologiques des eaux de surfaces et de la fenêtre d'observation de SWOT [Biancamaria et al., 2016]	7
4	Illustration de la régression linéaire simple par la méthode des moindres carrés	10
5	Illustration d'un neurone artificiel	10
6	Graphes de quelques fonctions sigmoïdes	11
7	Illustration d'un réseau de neurone artificiel constitué de 3 variables d'entrées (ou explicatives), de 2 couches cachées composées elles-mêmes de 4 neurones chacune, et d'une seule sortie (ou variable expliquée)	11
8	Illustration d'un RNN [Sherstinsky, 2020]	13
9	Illustration du déploiement d'un RNN. A représente un ANN, x_t l'observation au temps t d'une ou plusieurs variables explicatives et h_t la sortie de l'ANN associée à cette entrée. [Olah, 2020]	14
10	Graphe de la fonction logistique (sigmoid) et de sa dérivé.	14
11	Illustration d'une cellule LSTM [Sherstinsky, 2020]	15
12	Matrice des corrélations pour le jeu de données PEPSI	17
13	Comparaison des matrices de corrélation sur deux partition du jeu de données : amérique et asie	17
15	Biplot de l'ACP sur le jeu de données PEPSI	19
16	Matrice de corrélation sur le jeu de données HSWOT	20
17	Scree plot de l'ACP pour le jeu de données HSWOT	20
18	Contributions des variables aux dimensions 1 et 2 de l'ACP	21
19	Biplot de l'ACP sur le jeu de données HSWOT	21
20	Utilisation de Keras et de Pytorch dans les articles de la conférence NeurIPS [horace, 2020]	22
21	Illustration du processus <i>series to supervised</i>	25
22	Exemple d'affichage des résultats globaux	26
23	Exemple d'affichage des résultats pour une rivière	27
24	Résultats globaux de l'ANN <i>baseline</i> de prédiction du débit Q	28
25	Résultats de l'ANN <i>baseline</i> pour la rivière du Ganges	29
26	Résultats de l'ANN <i>baseline</i> pour la rivière de la Garonne	29
27	Comparaison entre les résultats sur le Ganges et la Garonne pour l'ANN <i>baseline</i>	29
28	Résultats globaux de l'ANN <i>climato</i> de prédiction du débit Q	29
29	Résultats de l'ANN sur Q <i>baseline</i> pour la rivière Iowa	30
30	Résultats de l'ANN sur Q <i>climato</i> pour la rivière Iowa	30
31	Comparaison entre les résultats du modèle ANN sur Q <i>baseline</i> et <i>climato</i> pour la rivière Iowa	30
32	Résultats de l'ANN sur Q <i>baseline</i> pour la rivière Cumberland	31
33	Résultats de l'ANN sur Q <i>climato</i> pour la rivière Cumberland	31
34	Comparaison entre les résultats du modèle ANN sur Q <i>baseline</i> et <i>climato</i> pour la rivière Cumberland	31
35	Résultats globaux de l'ANN <i>complet</i> de prédiction du débit Q	31

36	Résultats globaux de l'ANN <i>baseline</i>	33
37	Résultats globaux de l'ANN <i>complet</i>	33
38	Comparaison entre les résultats du modèle ANN sur \bar{A} <i>baseline</i> et <i>complet</i> pour les "petites rivières"	33
39	Résultats de l'ANN sur \bar{A} <i>complet</i> pour la rivière Ohio	34
40	Résultats de l'ANN sur \bar{A} <i>complet</i> pour la rivière Kushiyara	34
41	Démonstration des résultats du modèle ANN sur \bar{A} <i>complet</i> pour deux "petites rivières"	34
42	Résultats globaux de l'ANN <i>baseline</i>	35
43	Résultats globaux de l'ANN <i>complet</i>	35
44	Comparaison entre les résultats du modèle ANN sur \bar{A} <i>baseline</i> et <i>complet</i> pour les "grandes rivières"	35
45	Résultats de l'ANN sur \bar{A} <i>complet</i> pour la rivière Ganges	36
46	Résultats de l'ANN sur \bar{A} <i>complet</i> pour la rivière Ohio	36
47	Démonstration des résultats du modèle ANN sur \bar{A} <i>complet</i> pour deux "grandes rivières"	36
48	Résultats globaux du LSTM entraîné sur la Seine	37
49	Résultats globaux du LSTM entraîné sur la Ganges	37

Remerciements

Je souhaite remercier ici mon maître de stage, Jérôme Monnier, qui m'a donné l'opportunité de faire ce stage passionnant et qui m'a fait toucher du doigt, à mon échelle, les points clés d'un travail de recherche juste et efficace.

Merci aussi à Kevin Larnier. Ses conseils et son expérience m'ont permis de comprendre les problématiques principales liées au monde de la modélisation en hydrologie ainsi que d'être opérationnel rapidement pour ce stage.

1 Contexte et objectif du stage

1.1 Présentation des entreprises

Institut National des Sciences Appliquées (INSA) Toulouse L'INSA Toulouse est une des 205 écoles d'ingénieurs françaises à délivrer un diplôme d'ingénieur. C'est un établissement public, membre des écoles dites "fondatrices" du groupe INSA. 8 spécialités sont représentées, dont la spécialité "Mathématiques Appliquées (MA)", traitant majoritairement des problématiques liées à la modélisation statistique, analytique, physique, l'optimisation, les méthodes numériques de résolution et le traitement du signal. De nombreux enseignants-chercheurs présents au département mathématiques appliquées sont membres de l'Institut de Mathématiques de Toulouse (IMT), un laboratoire de recherche qui fédère la communauté mathématique de la région toulousaine en France. Cet environnement scientifique fait du département un environnement propice à la recherche : De 2017 à 2019, entre 10 à 15% des élèves diplômés ont décidé de poursuivre une thèse [INSA, 2020]. La recherche à l'IMT est organisée en six équipes principales, avec quelques chevauchements :

- Analyse
- Dynamique et géométrie complexe
- Équations aux dérivées partielles
- Géométrie topologie algèbre
- Probabilités
- Statistiques et optimisation

L'IMT est l'un des plus grands centres de recherche français en mathématiques, et ses activités scientifiques couvrent presque tous les domaines des mathématiques. Il compte environ 200 chercheurs permanents et 100 doctorants (en 2020), appartenant à différentes institutions de l'Université de Toulouse et du CNRS. On notera aussi que l'IMT est en charge du Prix Fermat et de la publication des Annales de la Faculté des Sciences de Toulouse. Ses bâtiments sont situés sur le campus de l'Université Paul Sabatier [IMT, 2020].

CS Group CS Group est une société de services en ingénierie informatique et intégrateurs de systèmes critiques intelligents et cyberprotégés. Membre de l'indice *CAC small* de la bourse de Paris et affichant un chiffre d'affaires de 230 millions d'euros en 2019, CS Group intervient sur différents grands secteurs économiques, comme la défense, l'aéronautique, le spatial et l'énergie. La majorité de ses revenus sont liés aux activités relatives à la défense et l'espace et, de surplus, en France. Le groupe intervient sur toute la chaîne de valeur, du conseil à l'intégration et la maintenance, en passant par la conception et le développement. Reconnue pour son expertise, la branche "espace" du groupe est amenée à travailler avec des clients importants comme le Centre national d'études spatiales (CNES) ou l'Agence spatiale européenne (ESA).

Les solutions de CS pour les systèmes spatiaux et les applications spatiales, au sol ou embarquées, sont au cœur des programmes civils et militaires. Les services proposés sont très complets :

- Contrôle commande
- Programmation de mission
- Dynamique du vol
- Simulateurs de satellites
- Simulateurs d'attitude
- Bancs de tests
- Stations sols

Cette expertise complète sur toute la chaîne de valeur a notamment permis au groupe d'être présent dans la réalisation de la plupart des grands programmes spatiaux européens depuis plus de 30 ans.

En plus des systèmes et des services spatiaux, CS Group propose aussi de la géo-information et traitement d'images satellite, mais aussi de la surveillance spatiale [CSGroup, 2020].

1.2 Contexte

Comprendre et anticiper le cycle de l'eau, plus précisément le débit des rivières, représente des enjeux critiques pour une multitude de secteurs : agriculture, urbanisation, prévention et réaction aux phénomènes naturels (inondations liées aux crues par exemple) [Edwards et al., 2015]. Cependant, estimer globalement le débit des rivières est pratiquement impossible à cause du manque de connaissances sur la totalité des interactions intervenant dans le cycle de l'eau, mais aussi, et surtout à cause du manque de mesure *in situ*. En effet, on constate un manque de stations de mesures, particulièrement dans les pays peu développés, ainsi que le déclin global du nombre de stations depuis quelques décennies, pour des raisons aussi bien financières que politiques.

Parallèlement à ce déclin, l'altimétrie satellitaire militaire et civile s'est développée au travers d'une multitude de missions pendant les 30 dernières années, amenant avec elle l'espérance de contrer le problème des stations de mesures. Même

si l'altimétrie satellitaire a permis de grandes avancées en hydrologie (voir [McCabe et al., 2017]), il a fallu attendre jusqu'en 2007 pour qu'une mission soit dédiée à la question de l'estimation globale du débit. Aujourd'hui encore, la variable débit reste l'une des variables hydrologiques les plus complexes à estimer quand on ne se situe pas dans un scénario optimal. En effet les différents modèles existants manquent de robustesse, qu'ils soient basés sur des approches physiques classiques, ou statistiques [Durand et al., 2016].

De plus, malgré les apports de l'altimétrie satellitaire, on note que certains problèmes apparaissent. La résolution spatio-temporelle des satellites détermine la nature des phénomènes possible à étudier. Une trop faible résolution spatiale ne permet pas une étude globale à l'échelle de la rivière, et une trop faible résolution temporelle ne permet pas l'étude de phénomènes hydrologiques rapides (comme les crues par exemple).

1.3 Objectif

La future mission Surface Water and Ocean Topography (NASA/CNES, 2022) a pour ambition de fournir (entre autres choses) une carte globale des débits des rivières de plus de 100m de large avec un satellite à orbite basse qui mesurera la largeur, la hauteur et la pente de la surface de l'eau des rivières avec une résolution spatio-temporelle sans précédent [Durand et al., 2016]. Dans la suite de ce rapport, on fait référence à ces mesures avec la paraphrase "mesures types SWOT". C'est dans ce contexte, catalyseur pour la recherche sur les modèles d'estimation du débit, que se place ce stage. On tente ici de rendre compte du potentiel d'une approche nouvelle, permettant d'améliorer les modèles existants d'estimation du débit.

Dans le cas du modèle auquel on vient essayer d'ajouter une contribution (modèle HiVDI [Larnier et al., 2020]), mais aussi au sein de plusieurs autres modèles, on constate une forte sensibilité à l'estimation de départ de débit qu'on leur fournit [Durand et al., 2016]. Le but de ce stage est donc d'implémenter une méthode basée sur des réseaux de neurones utilisant des données de simulations semblables aux futures données SWOT pour 'entraîner' un modèle à estimer le débit. On espère pouvoir utiliser cette valeur de débit comme une bonne approximation de départ pour le modèle principal (voir [Larnier et al., 2020]).

2 État de l'art

2.1 Le cycle de l'eau

Le cycle de l'eau est le concept au coeur de l'hydrologie. Il permet de comprendre les interactions entre l'eau et son environnement. L'unité de base en hydrologie pour étudier ces interactions est le bassin versant. Un bassin versant est défini comme une zone de terre dans laquelle toutes les précipitations entrantes s'écoulent au même endroit – vers la même masse d'eau ou la même dépression topographique – en raison de sa topographie. On décrit brièvement ses principaux composants du cycle de l'eau ci-dessous, en se basant sur la synthèse réalisée par [Edwards et al., 2015].

1. **Précipitation** : C'est l'entrée d'eau dans le bassin versant. Elle peut prendre la forme de pluie, de neige, ou de grêle. C'est un phénomène très variable dans le temps et l'espace. On note par exemple l'effet du soulèvement orographique qui intervient lorsqu'une masse d'air approche d'un obstacle, l'obligeant à gagner en altitude pour le franchir. Ce faisant, la masse d'air se condense, sous l'effet de la baisse du gradient de pression et de température, causant des précipitations. Ce phénomène peut être l'origine du développement de deux écosystèmes tout à fait différents entre un versant et l'autre d'un relief. Dans les zones forestières, on observe d'autres phénomènes sources de variabilité spatiale et temporelle dont le plus important est l'interception. En effet lorsque la forêt est dense, une grande partie des précipitations se retrouvent d'abord sur les feuilles des arbres. Dans le cas où la précipitation est peu intense relativement à la résistance mécanique des feuilles il est possible que l'eau s'évapore directement sans avoir atteint le sol, on parle alors de perte par interception. La proportion d'eau parvenant à s'écouler jusqu'au sol par rapport à celle qui est évaporée directement depuis les feuilles dépend de beaucoup de facteurs. On note par exemple les facteurs climatiques comme le type et l'intensité des précipitations, la puissance du vent, mais aussi le type d'arbres présents dans la forêt.
2. **L'évaporation** constitue l'une des principales sources de perte d'eau du bassin versant. L'eau peut s'évaporer de toutes surfaces qu'on retrouve dans un bassin versant : plantes, sol, eau, mais aussi routes et bâtiments. La quantité d'eau perdue par évaporation est directement liée à l'énergie solaire fournie au milieu.
3. **Transpiration** : Elle correspond à la perte d'eau des plantes depuis leurs stomates. C'est la plus grande source de perte d'eau chez les plantes. Elle est principalement contrôlée par les apports en eau extérieurs à la plante. Lorsque le milieu devient aride, les stomates se ferment, inversement lorsque le milieu s'humidifie. Étant lié au processus de photosynthèse, le taux de transpiration d'une plante est, comme pour l'évaporation, directement lié à l'énergie solaire fournie. On regroupe souvent l'évaporation et la transpiration sous le terme évapotranspiration.
4. **Rétention** : Les sols sont constitués de pores de différentes tailles. Les eaux qui pénètrent les sols viennent alimenter ces pores. Les plus petits, appelés micro pores, alimentent les racines des plantes par capillarité (on parle de potentiel hydrique). Au contraire, les plus gros pores (meso-, macro-pores) peuvent contenir une quantité d'eau plus grande où les phénomènes de capillarité sont largement contrebalancés par la gravité. L'eau qu'ils contiennent approvisionne donc les eaux souterraines.
5. Les **eaux souterraines** correspondent aux régions souterraines saturées en eau. On appelle aquifère les roches qui contiennent ces eaux dans leurs fractures et niveau piézométrique la séparation entre l'aquifère et les sols non saturés en eau. On fait la distinction entre aquifères locaux, alimentés par des précipitations récentes, peu profondes, et les aquifères régionaux, bien plus grands et profonds, contenant des eaux très vieilles (dizaines de milliers d'années).
6. Le **ruissèlement**, ou débit est la quantité d'eau qui s'écoule par unité de temps dans les cours d'eau de différentes tailles. Il est constitué de deux contributions principales : l'étiage, qui est la partie du débit issue des eaux souterraines lorsque le niveau piézométrique de ces dernières rencontre le lit du cours d'eau, et le "stormflow", correspondant à la contribution des précipitations et de la fonte des neiges au débit.

On regroupe ces quantités dans un bilan hydrique qui décrit le cycle de l'eau.

$$Q = P - ET \pm \Delta S \pm \Delta GW \quad (1)$$

- Q : Débit (ruissèlement)
- P : Précipitation
- ET : Évapotranspiration
- S : Humidité du sol (rétention)
- GW : Stock d'eau souterraine

On peut aussi considérer le cycle de l'eau sur une année hydrique complète. Une année hydrique commence au moment le plus sec ou le plus humide de l'année selon la convention choisie. En se plaçant sur une telle période, on peut considérer que les variations de stocks d'eau souterrains contrebalancent les variations de rétention d'eau du sol. L'Équation (1) devient alors :

$$Q = P - ET \quad (2)$$

2.2 État de l'art sur les mesures de la Terre par télédétection

2.2.1 Les prémisses de l'utilisation de la télédétection en sciences

Les premières tentatives de cartographie et de surveillance de la surface de la Terre ont été menées à partir de ballons-sondes, puis d'avions à voilure fixe et d'avions de reconnaissance à haute altitude tels que l'avion-espion U-2 utilisé par les États-Unis. Les répercussions géopolitiques en combinaison avec le rejet de la proposition d'Eisenhower de l'initiative "Ciel Ouvert" ont précipité un changement tactique vers la détection spatiale et, en fin de compte, vers la course à l'espace. Bien que les premiers systèmes satellitaires furent axés sur la reconnaissance militaire, l'utilité des capteurs spatiaux pour la météorologie et l'étude du climat a été rapidement reconnue [Nordberg, 1965]. On exploita notamment ce potentiel avec la mission Landsat 1 (NASA, 1972) qui fut la première mission spatiale de télédétection destinée à des fins civiles. Depuis cette mission, d'exceptionnelles avancées ont été réalisées autant en aval avec les performances des capteurs utilisés qu'en amont avec le traitement des données. D'un point de vue hydrologique, la télédétection a su apporter une dynamique nouvelle à l'étude des phénomènes qui constituent le cycle de l'eau, mais il subsiste néanmoins un fossé entre nos capacités de mesures et nos capacités à analyser ces dernières [McCabe et al., 2017].

2.2.2 Les obstacles dans le processus de recherche liés à la télédétection

1. **Les défis de la récupération et de l'interprétation des mesures par satellites** : Un des problèmes principaux de la télédétection appliquée à la géophysique est la limitation de ne travailler qu'avec des mesures de radiations rétro propagées depuis la surface de la Terre. En effet, des modèles d'extraction complexes, avec leurs diverses simplifications, paramétrisations et solutions non uniques, sont presque toujours utilisés pour transformer la mesure satellitaire en une variable d'intérêt spécifique. Pour certaines variables, cette conversion est assez simple (par exemple, l'indice de végétation par différence normalisée, NDVI), tandis que pour d'autres, le modèle d'extraction peut avoir des hypothèses sous-jacentes ou nécessiter des données auxiliaires qui contribuent de manière significative à l'erreur d'extraction (par exemple, l'humidité du sol, l'évaporation ou encore le débit).
2. **L'homogénéité et l'harmonisation des données** La temporalité élevée des cycles de certains phénomènes hydrologiques (pouvant être de l'ordre de la décennie) et des phénomènes climatiques (de l'ordre du centenaire) exigent, pour une étude facilitée, de disposer de données continues et homogènes. Cependant, le fonctionnement des différents acteurs spatiaux ayant participé à des programmes de mesures satellites rend cette tâche complexe. On dénombre effectivement trop peu de missions longues, principalement pour des raisons de budget, mais aussi une variabilité non négligeable dans les technologies de télédétections utilisées. En conséquence, les jeux de données à longue temporalité sont obtenus en fusionnant des jeux de données issues de différents capteurs sur différentes périodes.
3. **Les contraintes liées à l'ingénierie satellitaire** : On distingue 2 types d'orbites lorsque l'on traite d'altimétrie spatiale. La première est atteinte aux alentours de 36 000 km d'altitude au niveau de l'équateur, c'est l'orbite géostationnaire ou "Geostationary Orbit" (GEO). Les satellites placés en orbite à cette altitude peuvent effectuer des mesures avec une très bonne résolution temporelle, mais sont cependant très limités en termes de couverture spatiale. Le second type d'orbite, atteinte aux alentours de 700 km est appelée orbite basse ou "Low Earth Orbit" (LEO). Un satellite opérant à cette orbite effectue des cycles de l'ordre de plusieurs jours au cours desquels il couvre la totalité ou une très grande partie de la surface de la Terre. Ce faisant, il permet d'effectuer des mesures avec une couverture quasi optimale, mais au prix d'une résolution temporelle de l'ordre de la durée de ses cycles. En météorologie ou en hydrologie, la grande variabilité temporelle et spatiale des phénomènes ne permet pas d'utiliser un seul type d'orbite pour tous les étudier.
4. **Le déclin des infrastructures de mesures (in situ)** : Les mesures in situ représentent la clé de voûte de tout processus d'étude des variables géophysiques par télédétection. Malheureusement, on constate une forte disparité d'un au niveau de la qualité de mesures et couverture spatiale entre les pays développés et ceux en voie de développement. De plus, depuis les années 80, on observe un déclin du nombre de stations de mesure, même dans les pays développés, ce qui pose problème pour la vision au long terme nécessaire à une étude approfondie des phénomènes hydrologiques et climatiques.

2.2.3 Les challenges spécifiques à l'hydrologie

Les obstacles cités précédemment sont, dans leur essence, communs à toutes les disciplines de télédétection terrestre. Dans cette partie, on s'attarde aux spécificités de quelques-unes des variables d'intérêt du cycle de l'eau.

1. **Précipitation** : C'est l'un des phénomènes ayant la plus grande variabilité spatio-temporelle (de l'ordre de l'heure à la journée en temps, de la centaine de mètres à la centaine de kilomètres en espace). Un satellite à orbite basse ou géostationnaire seul ne suffit pas à rendre compte précisément de cette variabilité. De plus, les variantes telles que la bruine ou la chute de neige sont très complexes à mesurer à cause de phénomènes de rétrodiffusion ou de

"backscatter" [Gaona et al., 2016]. Actuellement, les stations de mesures *in situ* représentent la meilleure source de donnée à disposition malgré la faible couverture spatiale dans les pays peu développés.

2. **Évaporation** : La récupération de cette variable repose sur des processus empiriques nécessitant beaucoup d'informations auxiliaires. L'estimation mondiale de l'évaporation par satellite est actuellement trop peu précise pour être utilisée dans des applications critiques. Cependant, associée à ces informations auxiliaires (température à la surface, mesures de radiation ...) il est possible de tirer des représentations de suffisamment bonne résolution.
3. **Humidité du sol** : Cette donnée est calculable à partir d'algorithmes se basant sur des mesures de rétro propagation des micro-ondes sur le sol. Grâce à la mission Soil Moisture and Ocean Salinity (SMOS, ESA, 2009) la communauté scientifique dispose maintenant de jeux de données globales et l'on sait maintenant que la partie limitante de l'étude de l'humidité du sol se situe plutôt du côté modélisation et algorithmique. La recherche est donc principalement tournée vers l'amélioration des algorithmes inverse [Mladenova et al., 2014].
4. **Neige** : L'étude des dimensions de la surface du manteau neigeux est un domaine mature de la recherche depuis près de 2 décennies. Les autres variables, plus utiles pour l'hydrologie, telles que la profondeur de neige ou l'équivalent en eau du manteau neigeux sont quant à elles complexes à mesurer dû à leur forte variabilité saisonnière, particulièrement dans les régions montagneuses. Un des problèmes majeurs pour la mesure étant la bande de fréquence utilisée par les satellites, sensible à saturation et aux phénomènes de rétrodifussion.
5. **Débit** : Variable hydrologique la plus critique étant donné les implications qu'aurait une mesure précise et continue de cette dernière, que nous énumérons dans la section suivante. Les algorithmes inverses sur lesquels reposent nos estimations actuelles sont limités par la précision des données utilisées et par extension des satellites dont elles sont issues. Bien que la recherche et le développement de ces algorithmes soient encourageants, le déclin du nombre de stations de mesure complexifie grandement la tâche. On notera aussi que, comme pour les précipitations, certains phénomènes comme les crues ont une variabilité temporelle élevée ce qui rend une étude globale de ces dernières impossible pour un satellite seul.

Les autres variables non citées ici telles que la vapeur d'eau ou la qualité de l'eau sont aussi sujettes aux limitations spatio-temporelles inhérentes aux satellites d'observation, et ne représentent pas un apport significatif dans la modélisation du débit, variable qui nous intéresse ici.

2.3 Mesurer le débit des rivières

Comprendre et anticiper l'évolution du débit des rivières comporte de forts enjeux :

- Prévoir les changements climatiques
- Quantifier les ressources disponibles à la consommation.
- Quantifier les ressources pour l'agriculture et le développement urbain.
- Anticiper et réduire les risques liés aux crues et inondations

2.3.1 Les mesures *in situ*

Étant donné que le débit est le produit de la surface et de la vitesse d'écoulement, la mesure *in situ* du débit des rivières exige des mesures spatialement explicites du profil de vitesse verticale. Bien que les mesures *in situ* du débit peuvent permettre une grande précision, elles sont longues et peu pratiques pour un *monitorage* continu. Ce *monitorage* est souvent réalisé à l'aide de jauge fluviales qui s'appuient sur des mesures périodiques et simultanées du *niveau du fleuve* (hauteur au-dessus d'une donnée arbitraire) et du débit pour établir une "courbe de tarage" ou "rating curve" à partir d'une simplification de la formule de Manning-Strickler (voir Equation 3). Une fois cette courbe définie, la mesure du niveau de la rivière est effectuée (généralement) par un transducteur de pression, ce qui permet de prédire presque continuellement le débit de la rivière, voir Figure 1. Il a été constaté que l'erreur globale affectant les observations de débit des rivières se situait entre 20% et 30%, avec un niveau de confiance de 95% [Di Baldassarre and Montanari, 2009].

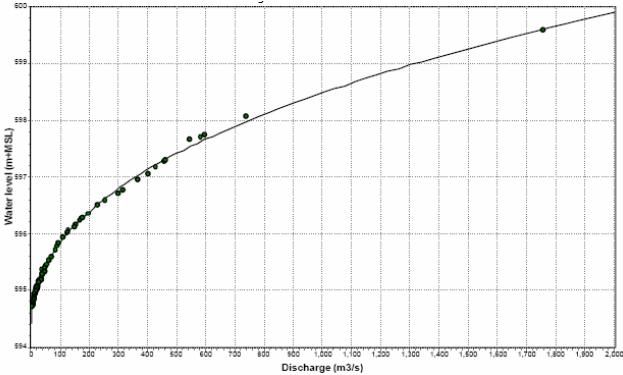


FIGURE 1: Courbe de tarage (niveau de l'eau en fonction du débit). Les points représentent les mesures de hauteur d'eau et de débit, la courbe représente la courbe de tarage induite par ces mesures. [Braca, 2020]

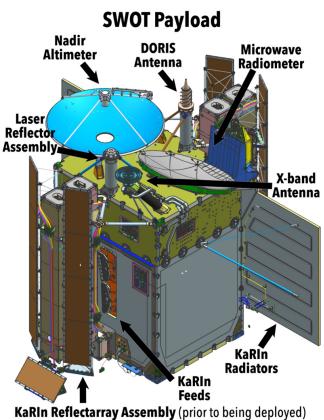
2.3.2 La mission SWOT (NASA/CNES, 2022)

Comme expliqué dans la section précédente, la mesure du débit des rivières à l'échelle mondiale est sujette à complications. Le potentiel du spatial pour répondre à la problématique du déclin des stations de mesures et donc de la couverture spatiale est évident, mais aucune mission pour répondre directement à cette problématique jusqu'en 2007. C'est cette année que la NASA officialise son nouveau programme spatial Surface Water and Ocean Topography (SWOT) où viendront plus tard se greffer plusieurs autres agences spatiales, notamment le Centre National d'Etudes Spatiales (CNES). [Biancamaria et al., 2016]

La mission SWOT a un objectif double :

1. Océanographie : Améliorer les modèles climatiques en mesurant les mouvements de l'océan à fine échelle (20 km).
2. Hydrologie : Améliorer la quantification des réserves d'eau douce et planifier les risques naturels tels que les inondations et les sécheresses à l'échelle mondiale en mesurant le débit des rivières.

Ce qui nous intéresse ici est uniquement l'aspect hydrologique de la mission, mais il est important de noter que cette dernière sera la première étude à échelle mondiale des eaux de surface de la Terre. Le lanceur, construit par Thales Alenia Space, branche spatiale de l'entreprise française Thales, sera équipé de nombreux outils innovants. On s'intéresse notamment au nouveau radar à synthèse d'ouverture (SAR) "KaRin" développé par le laboratoire de recherche de la NASA Jet Propulsion Laboratory (JPL). La particularité de ce radar est que la bande de fréquence à laquelle il opère (bande "Ka" : 35.75 GHz) permet une résolution spatiale de l'ordre des 100m, permettant une étude globale des masses d'eau comme les rivières. Plus précisément, le SAR de SWOT sera capable de fournir la hauteur, la largeur et la pente des rivières d'au moins 100m de large avec une précision verticale d'environ 1cm par kilomètre carré sur une bande de 120km. Cela constitue une avancée considérable par rapport aux altimètres précédents utilisés pour les applications hydrologiques qui rapportent seulement des hauteurs en 1-D. [Biancamaria et al., 2016]



(a) Diagramme des composants de la charge utile de SWOT [NASA, 2020]



(b) Vaisseau SWOT [NASA, 2020]

La plateforme aura une période d'orbite 21 jours, ce qui signifie qu'elle produira des observations de toutes les surfaces continentales tous les 21 jours, pendant 3 ans. Avec de tels paramètres d'orbite, seules les questions relatives au cycle

global de l'eau pourront trouver une réponse, là où il sera impossible en revanche d'observer la dynamique temporelle des phénomènes locaux comme les crues ou les précipitations (voir la Figure 3).

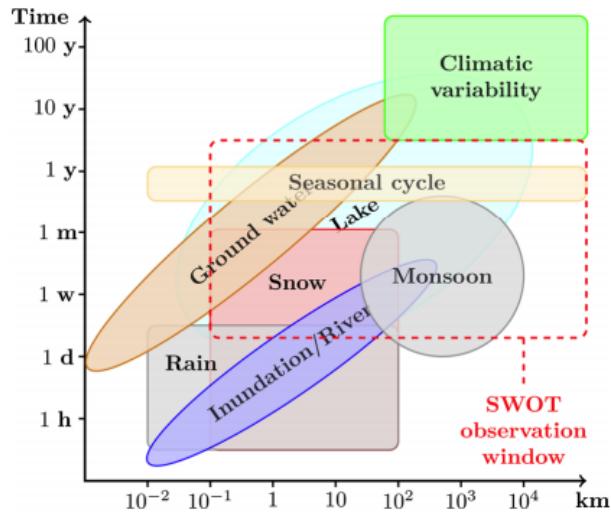


FIGURE 3: Diagramme spatio-temporel des processus hydrologiques des eaux de surfaces et de la fenêtre d'observation de SWOT [Biancamaria et al., 2016]

Ce futur apport de données globales et de bonne résolution est très intéressant pour la communauté scientifique. Bien qu'on ne s'attende pas à obtenir des estimations du débit plus précises que les mesures *in situ*, cette mission permettra de disposer de données continues dans l'espace, c'est à dire même dans les zones historiquement non jaugées. Il se pose néanmoins la question du calibrage des algorithmes inverse. En effet, les méthodes existantes d'estimation du débit reposent pour la plupart sur des données *a priori* des rivières étudiées, pour contraindre les calculs [Durand et al., 2016]. Une grande partie du challenge repose donc sur l'estimation du débit dans ces zones historiquement non jaugées et pour lesquelles les algorithmes ne peuvent pas être calibrés.

2.3.3 Le problème de l'estimation du débit

L'objectif des chercheurs de la *science team* est d'implémenter une méthode robuste pour estimer le débit $Q(x, t)$ de rivières aux caractéristiques hydriques très variées et parfois situées dans des zones où on ne dispose pas de mesures *in situ* de hauteur de pente et de largeur de l'eau *in situ* pour contraindre les calculs. On distingue 3 scénarios différents, que l'on classe par ordre croissant de difficulté à traiter :

1. Rivières non jaugées : Aucune mesure n'est disponible à part les mesures d'altimétrie spatiale.
2. Rivières partiellement jaugées : On dispose de quelques mesures sur une partie de la rivière. Ces valeurs peuvent être issues de bases de données antérieures ou bien de sorties de modèles hydrologiques calibrés.
3. Rivières jaugées : On dispose d'un profil de bathymétrie.

Le scénario le plus délicat à traiter est celui où l'on ne dispose pas de données *in situ*. Délicat, car le problème inverse que l'on peut vouloir résoudre est, de fait, mal posé (voir Equation 3). Effectivement en théorie, retrouver le débit d'une rivière nécessite la connaissance de variables non observables directement, comme le coefficient de Strickler $K(t, x)$ ou bien la bathymétrie $b(x)$. C'est pourquoi au sein de la Science Team, différents chercheurs expérimentent d'autres approches qui pourront venir compléter la base physique des modèles d'écoulements classique comme Saint-Venant et Manning Strickler. Certaines reposent sur la modélisation statistique, comme celle proposée par [Kouraev et al., 2004]. D'autres utilisent l'assimilation variationnelle de données en prenant en compte les différents scénarios, comme le modèle HiVDI proposé par [Larnier et al., 2020] et auquel on tente de contribuer au travers les résultats de ce stage.

Rappelons une des formulations de l'équation empirique Manning-Strickler décrivant la vitesse moyenne d'un fluide s'écoulant en surface libre :

$$V = K_s R_h^{2/3} S^{1/2} \quad (3)$$

où

- V : Vitesse moyenne de la section transversale ($m.s^{-1}$)
- K_s : Le coefficient de Strickler (inverse du coefficient n de Manning)
- R_h : Le rayon hydraulique (m)
- S : La pente hydraulique (cm/km)

Le modèle HiVDI, présenté dans [Larnier et al., 2020] peut se décomposer (grossièrement) comme suit :

1. Période d'apprentissage (à itérer sur différentes périodes hydrologiques) : le but est de fournir le meilleur *prior* où estimation du triplet (Q, K, A) à l'algorithme d'assimilation variationnelle de données. Pour le *prior* sur Q on utilise le Q_{MAF} (Mean Annual Flow) obtenu via la base de données a priori [Andreadis et al., 2013] ou via le Water Balance Model [Wisser et al., 2010]. On procède de même pour la valeur du *prior* sur K . Pour la bathymétrie b , on distingue 3 scénarios :

- (a) Rivières non mesurées : On inverse l'équation 4 (autre forme de l'équation de Manning Strickler) en utilisant les *prior* sur Q et K .

$$(K^{-1}Q)^{3/5} = (A_0 + \delta A) W^{-2/5} S^{3/10} \quad (4)$$

Avec K le coefficient de Strickler, Q le débit, A la section, W la largeur de la surface d'eau et S la pente de la surface d'eau.

- (b) Rivière partiellement mesurée : On inverse le système algébrique suivant, dit "Low Froude" (on néglige l'inertie, $Fr < 1$) :

$$c \cdot K^{3/5} A_0 + d \cdot K^{3/5} = Q^{3/5} \quad (5)$$

avec $c = W^{-\frac{2}{5}} S^{\frac{3}{10}}$ et $d = c\delta A$

- (c) Rivière déjà mesurée : On dispose d'observations de la bathymétrie, on calcule le profil de bathymétrie complet avec l'expression "Low Froude Bathymetry" suivante :

$$b + (Z - b) \cdot \mathcal{O}^{-1} \cdot \mathcal{O} = Z \quad (6)$$

Avec b la bathymétrie et $\mathcal{O} = (W\sqrt{S})^{-\frac{3}{5}}$

Un module d'assimilation de donnée fournie des estimations du triplet (Q, K, A) à partir de données d'observations de type SWOT ainsi que des *prior* du triplet calculé précédemment.

2. Ces estimations sont ensuite utilisées par un algorithme (appelé "Low complexity") basé sur l'équation de Manning-Strickler avec l'hypothèse d'un courant calme (on néglige l'inertie, $Fr < 1$) pour calculer la valeur de A et K réelle.
3. Temps réel : On utilise ces valeurs de A et K réelles pour, à partir des futures observations SWOT, calculer le débit Q réel, en utilisant l'algorithme "Low complexity".

2.3.4 Comparaison des modèles d'estimation du débit

Dans ses travaux, [Durand et al., 2016] propose une comparaison de 5 algorithmes d'estimation du débit différents, ainsi que les pistes d'amélioration envisageables. Le jeu de données utilisé pour estimer la performance des différents algorithmes est composé du même type de mesures que celles que fournira SWOT. Ces données ont été générées par un modèle hydraulique contraint par des mesures *in situ* de bathymétrie et de débit, le tout pour 19 rivières aux caractéristiques hydriques variées. On notera de plus que la fréquence de mesure simulée est journalière et que le bruit ajouté est quasi nul. Ces données sont donc idéalisées, mais permettent néanmoins une comparaison équitable des différents algorithmes.

Les 5 algorithmes en question sont :

- At-many-stations hydraulic geometry (AMHG)
- GaMo
- MetroMan
- Mean Flow and geomorphology
- Mean flow constant Roughness (MCFR)

Sur les 5 algorithmes présentés dans cette étude, 4 sont basés sur une variante de l'équation (3), le dernier utilise un algorithme génétique d'optimisation.

Les résultats de l'étude de [Durand et al., 2016] suggèrent qu'une synergie entre les différents modèles permettrait d'améliorer la robustesse et la précision déjà très prometteurs de l'ensemble des algorithmes. Indépendamment les algorithmes sont relativement efficaces dans certains cas de figure, on constate une erreur quadratique relative (RRMSE) de 35% par au moins un des 5 algorithmes sur 14/16 rivières à géométrie simple et sans obstacle. Les performances d'estimations sont cependant très mauvaises (jusqu'à plus de 100% d'erreur quadratique relative) pour les rivières présentant des caractéristiques hydrologiques complexes comme du tressage, ou des obstacles faisant barrage. Il reste important de noter que dans le cadre de cette étude aucune donnée auxiliaire n'a été exploitée, or en pratique le gain potentiel à utiliser ces dernières pour calibrer et/ou contraindre son algorithme est très significatif [Larnier et al., 2020] [Lin et al., 2019].

2.3.5 L'opportunité offerte par réseaux de neurones

Les réseaux de neurones sont des outils faisant partie des nombreuses méthodes d'apprentissage supervisé, permettant de résoudre des problèmes d'optimisation très complexes, sous réserve de disposer de suffisamment de données pour les "entraîner". Dans la section suivante, on décrit plus en détail leur fonctionnement.

L'usage de réseaux de neurones dans l'estimation du débit a d'abord prouvé son potentiel à travers le travail de [Beck et al., 2015] qui a produit une carte globale des débits à l'échelle du bassin versant. Le modèle développé lors de cette étude se base sur des mesures de type SWOT ainsi que des variables climatiques et quelques variables liées au sol. Les résultats suggèrent une forte corrélation entre le débit et les caractéristiques climatologiques du bassin versant étudié. L'auteur met cependant le lecteur en garde, car les caractéristiques ne présentant pas ou peu de corrélation avec le débit ne sont pas pour autant à éliminer des modèles. En effet, les variables liées au sol n'ont pas montré de forte corrélation avec le débit, cependant on sait depuis plusieurs décennies que ces variables exercent un contrôle sur le débit et ses caractéristiques (e.g., Farvolden 1963 ; Boorman et al. 1995 ; Bruijnzeel 2004). Ces résultats nous indiquent une composante clé à prendre en compte avec l'utilisation des réseaux de neurones qui est que la physique sous-jacente peut être contredite par le modèle sans que l'on ne puisse déterminer pourquoi directement.

Plus récemment, c'est surtout à des fins de calibration et de correction de biais que viennent se greffer les réseaux de neurones dans les modèles d'estimation déjà existants, et ce avec de bons résultats. Les travaux de [Lin et al., 2019] montrent, par exemple, la pertinence des réseaux de neurones pour contraindre un modèle, en s'appuyant sur des bases de données de mesures de débits. Ici on recourt à une plus grande variété de variables : topographiques et géologiques. Une telle calibration prend tout son sens pour réduire le biais d'estimation à l'échelle fine que représente la rivière. En effet, on constate un biais de $\pm 20\%$ sur 35% des 14000 mesures in situ, et un score mensuel de "Kling-Gupta Efficiency" supérieur à 0.6 avec le modèle VIC, ce qui constitue très bons résultats.

Note : L'efficacité de Kling-Gupta est un indicateur de performance pour les modèles hydrologiques par rapport aux données observées. Il vaut 1 quand le modèle modélise parfaitement les données observées et $-\infty$ le cas échéant. Le reste des prédictions du modèle (64%) présentent un biais de $\pm 50\%$ et un Kling-Gupta Efficiency supérieur à 0.2.

C'est dans ce cadre que l'on tente d'exploiter les réseaux de neurones pour améliorer le modèle HiVDI.

2.4 Comprendre les réseaux de neurones

2.4.1 Raison d'être et fondements théoriques de l'apprentissage supervisé

Donner la faculté aux ordinateurs "d'apprendre" à partir de données constitue un champ d'études important de l'informatique et des mathématiques, c'est ce qu'on appelle l'apprentissage automatique ou "machine learning". Cette discipline nouvelle, mais dont les fondements trouvent place au 19e siècle n'a pendant longtemps pas pu être exploitée à son plein potentiel, mais suscite depuis quelques décennies un fort engouement avec l'essor du big data et l'explosion de la puissance de calcul des ordinateurs. Les types de problèmes que l'on peut rencontrer en apprentissage automatique sont très variés : traduction, classification d'images, reconnaissance vocale, prédiction sur des séries temporelles, etc. Dans le cas le plus courant de l'apprentissage supervisé, que l'on étudiera ici, le *système* utilise des exemples connus ou "étiquetés" pour apprendre, ce qui veut dire qu'on sait ce que l'on veut obtenir à partir des données. De façon générale, l'apprentissage supervisé peut se résumer en deux étapes clés :

1. La phase d'apprentissage : On utilise des exemples étiquetés de couples entrés/sortis pour produire un modèle.
2. La phase de test : On utilise de nouvelles données et on examine si le modèle obtenu à la suite de la phase d'apprentissage fournit une estimation correcte à partir de ces dernières.

Prenons l'exemple de la régression linéaire, qui est une méthode basique d'apprentissage supervisé. On considère un couple d'entrées/sorties $(x_i, y_i)_{1 \leq i \leq N}$ qui sont respectivement des réalisations de la variable explicative X et de la variable cible Y . On fait l'hypothèse que la relation entre la variable cible Y et la variable explicative X est linéaire et l'on construit le modèle linéaire dit "simple" : $\forall i, y_i = f(x_i) + \epsilon_i$, où f est une fonction affine et ϵ_i représente l'erreur d'approximation (voir Figure 4). Pour déterminer les coefficients de la droite affine f on utilise l'estimateur des moindres carrés que l'on applique uniquement sur les couples $(x_i, y_i)_{1 \leq i \leq n}$ où $n < N$ (on parle alors de partition d'entraînement des données), c'est la phase d'apprentissage. Le principe général qu'on retrouve en phase d'apprentissage, c'est la minimisation d'une erreur d'approximation. Dans le cas de la régression linéaire simple, on sait résoudre explicitement ce problème d'optimisation, mais dans la plupart des cas, par exemple avec les réseaux de neurones, minimiser cette erreur fait appel à des algorithmes itératifs. On peut ensuite, à partir d'autres réalisations $(x_i, y_i)_{n \leq i \leq N}$ (on parle ici de partition de test des données), produire des prédictions $\hat{y}_i = f(x_i)$ et examiner la valeur du résidu $\hat{\epsilon} = y_i - \hat{y}_i$, c'est la phase de test.

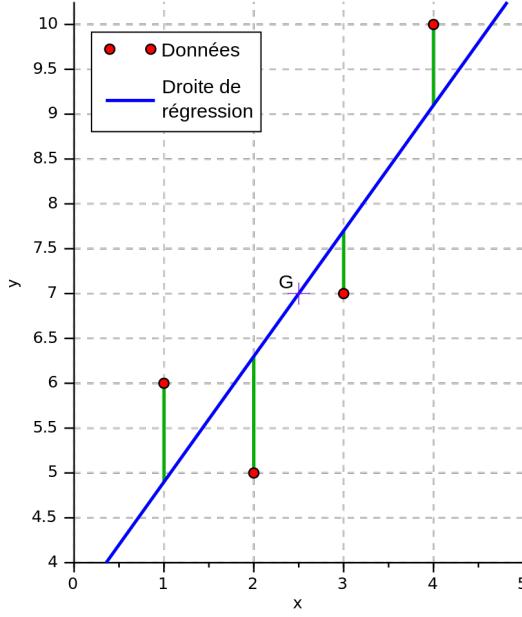


FIGURE 4: Illustration de la régression linéaire simple par la méthode des moindres carrés

Les méthodes classiques d'apprentissage supervisé comme la régression linéaire ainsi que ses variantes sont efficaces dans le cas où l'on modélise des phénomènes peu complexes. Essayer de modéliser des phénomènes fortement non linéaires n'est pas chose facile. C'est dans ce cas de figure que prennent leur sens les réseaux de neurones, introduit pour la première fois par [LeCun et al., 1989] et dont les fondements seront théorisés par [Hornik, 1991]. Dans la section suivante, nous expliquons les tenants et les aboutissants de cette approche et nous expliquons en quoi elles surpassent les méthodes conventionnelles d'apprentissage supervisé.

2.4.2 Structure d'un réseau de neurones et exemple de l'ANN

Un réseau de neurones artificiels est une application, non linéaire par rapport à ses paramètres θ qui associe à une entrée x une sortie $y = f(x, \theta)$. Par souci de simplicité, nous supposons que y est unidimensionnel, mais il pourrait aussi être multidimensionnel. Nous précisons la forme de la fonction f par la suite. La particularité des réseaux de neurones est que la méthode d'apprentissage sous-jacente, appelée la rétropropagation du gradient, permet de trouver facilement un minimum local de la fonction à minimiser si cette dernière n'est pas convexe.

Le neurone artificiel et la fonction d'activation La brique élémentaire du réseau de neurones est le neurone artificiel (voir Figure 5). Inspiré du fonctionnement des neurones biologique il représente une fonction f_j de l'entrée $x = (x_1, \dots, x_d)$ pondérée par un vecteur de poids $w = (w_1, \dots, w_d)$ auquel on ajoute un biais b_j soit :

$$y_j = f_j(x) = \phi(\langle w_j, x \rangle + b_j)$$

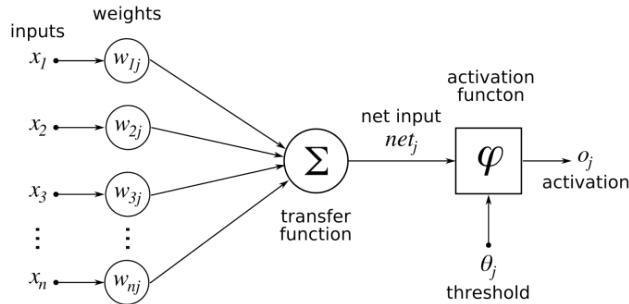


FIGURE 5: Illustration d'un neurone artificiel

On appelle Φ "fonction d'activation". Elle peut prendre différentes formes selon le type de problème traité (régression ou classification). Les exemples les plus courants de fonctions d'activation sont les fonctions sigmoïdes (en forme de S)

dont la plus populaire est la fonction logistique que l'on écrit :

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

Toute fonction sigmoïde peut s'exprimer par translation et/ou dilatation d'une autre fonction sigmoïde. Par exemple pour la fonction tangente hyperbolique :

$$\frac{1}{1 + e^{-x}} = \frac{1}{2} + \frac{1}{2} \tanh\left(\frac{x}{2}\right).$$

Cette classe de fonction est intéressante notamment, car très facilement dérivable. On trouve en Figure 6 le graphe de quelques fonctions sigmoïdes.

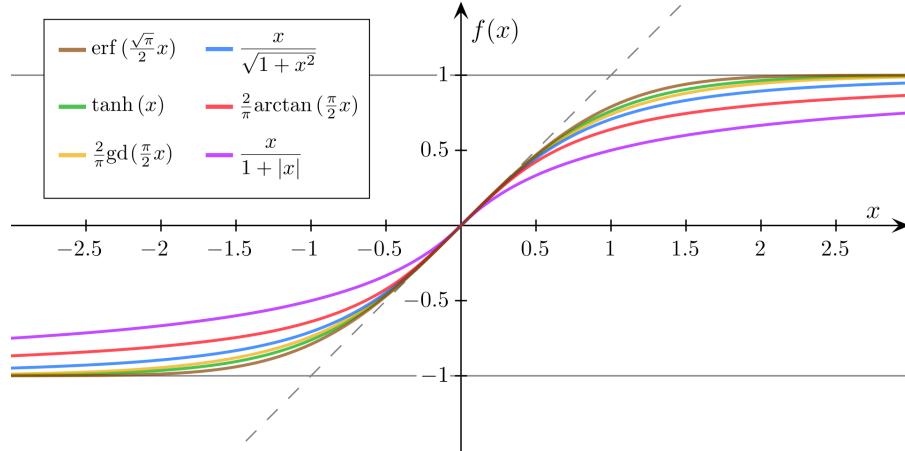


FIGURE 6: Graphes de quelques fonctions sigmoïdes

Le réseau de neurones Un réseau de neurones est une structure composée de plusieurs couches dites "cachées". Dans le cas du réseau de neurones artificiel (ou classique), chacune de ces couches est composée de neurones artificiels dont la sortie devient l'entrée de chacun des neurones artificiels de la couche cachée suivante (voir Figure 7). On appelle la première couche du réseau "couche d'entrée" et la dernière "couche de sortie". Dans d'autres structures comme les réseaux de neurones récurrents que nous verrons par la suite, la sortie d'un neurone peut être une entrée d'un neurone de la même couche ou encore d'un neurone de la couche précédente.

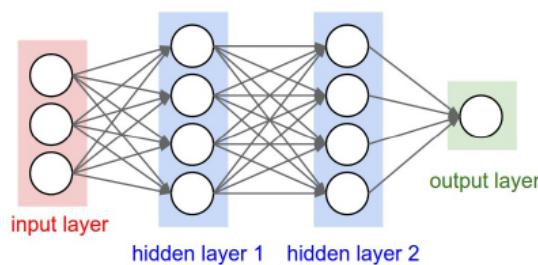


FIGURE 7: Illustration d'un réseau de neurone artificiel constitué de 3 variables d'entrées (ou explicatives), de 2 couches cachées composées elles-mêmes de 4 neurones chacune, et d'une seule sortie (ou variable expliquée)

Les paramètres d'un réseau de neurones sont son nombre de couches cachées, le nombre de neurones de chaque couche, ainsi que les fonctions d'activations choisies. Il est important de noter que la fonction d'activation de la couche de sortie est généralement différente de celle utilisée dans les couches cachées. Par exemple pour le cas de la classification on choisit une fonction de telle sorte que les neurones de la couche de sortie, représentant les différentes classes, aient une valeur comprise entre 0 et 1. Cette valeur représente alors la probabilité que la classe soit celle à laquelle appartient l'individu dont les données d'entrées sont issues. Dans le cas qui nous intéresse de la régression on n'applique pas de fonction d'activation sur la couche de sortie (fonction identité).

Le théorème universel d'approximation Le théorème donnant tout son intérêt aux réseaux de neurones est le théorème universel d'approximation de Hornik [Hornik, 1991]. Ce résultat prouve que toute fonction bornée, régulière de $\mathbb{R}^d \mapsto \mathbb{R}$ peut être approximé à ϵ près par un réseau de neurones contenant une couche cachée et un nombre fini de neurones utilisant la même fonction d'activation, avec un neurone sur la couche de sortie. D'abord prouvé par Cybenko dans le cas où la fonction d'activation est une sigmoïde [Cybenko, 1989]. Formellement le théorème s'écrit :

Théorème 1 Soit Φ une fonction d'activation bornée, continue et non décroissante. Soit K_d un compact de \mathbb{R}^d et $\mathcal{C}(K_d)$ l'ensemble des fonctions continues sur K_d . Soit $f \in \mathcal{C}(K_d)$ alors $\forall \epsilon > 0$ il existe $N \in \mathbb{N}$, des réels v_i, b_i et des vecteurs w_i de \mathbb{R}^d tel que si on définit

$$F(x) = \sum_{i=1}^N v_i \phi(\langle w_i, x \rangle + b_i)$$

on a

$$\forall x \in K_d, |F(x) - f(x)| \leq \epsilon$$

2.4.3 Le problème d'optimisation sous-jacent et la méthode de la rétropropagation du gradient

On peut maintenant écrire le problème d'optimisation associé à l'estimation des paramètres du problème, c'est-à-dire les poids et les biais, que l'on regroupe par simplicité d'écriture dans le vecteur des paramètres θ . Soit X la matrice représentant les données d'entraînement de taille (n, m) où n représente le nombre d'échantillons et m le nombre de variables explicatives. On note x_i la ligne i de cette matrice et y_i la i ème valeur du vecteur Y de taille n , représentant la sortie associée à l'entrée x_i . La phase d'entraînement d'un réseau de neurones consiste donc à minimiser la fonction de coût empirique suivante :

$$\arg \min_{\theta} C(\theta) = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n (f(\theta, x_i) - y_i)^2$$

Avec $f(\theta, x_i)$ la valeur de sortie du réseau de neurones ayant eu x_i en entrée, dimension la dimension de y_i . On remarquera qu'ici on choisit de minimiser l'erreur quadratique empirique ; le choix de cette fonction dépend encore une fois du type de problème traité, on traite ici du problème de régression.

Pour résoudre ce problème, l'approche naturelle consiste à utiliser un algorithme de descente de gradient. Le problème que l'on peut rencontrer avec des structures comme les réseaux de neurones est qu'il faut prendre en compte les différentes couches et les liaisons entre les neurones de ces couches dans le calcul du gradient. L'approche conçue par [LeCun et al., 1989] s'appelle la "rétropropagation du gradient" ou "backpropagation". L'idée derrière la méthode de la rétropropagation du gradient est de calculer récursivement le gradient à travers les couches cachées du réseau de neurones pour ajuster les poids. Formellement, on commence par définir les quantités suivantes :

$$\begin{aligned} z_j^{(L)} &= w_{jk}^{(L)} a_k^{(L-1)} + b_j^{(L)} \\ a_j^{(L)} &= \Phi(z_j^{(L)}) \end{aligned}$$

Où $a_k^{(L)}$ représente la valeur du neurone k de la couche L et $w_{jk}^{(L)}$ le poids de la liaison entre le neurone k de la couche $L-1$ et le neurone j de la couche L . Le calcul d'une composante du gradient de la fonction de coût (pour l'individu i du jeu de données d'entraînement), s'exprime alors, par la formule de la chaîne, comme suit :

$$\frac{\partial C_i}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_{L-1}} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_i}{\partial a_j^{(L)}}$$

On obtient la composante du gradient finale en sommant les dérivées partielles sur tous n individus du jeu de données d'entraînement :

$$\frac{\partial C}{\partial w_{(L)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial C_i}{\partial w_{(L)}}$$

On répète le processus pour tous les paramètres du vecteur $\theta = (w_{(0)}, b_{(0)}, w_{(1)}, \dots, b_{(L)})$ et on obtient le gradient de la fonction de coût qui va nous permettre de calculer une itération (ou *epoch* dans le jargon deep learning) grâce à une méthode de descente de gradient.

Cette formule valable en théorie se retrouve très coûteuse en pratique quand le jeu de données est grand. On utilise alors différentes méthodes permettant de réduire le coût de calcul comme la version stochastique du gradient de descente, où seul un sous-ensemble du jeu de données est nécessaire au calcul du gradient pour trouver une direction de descente. On développe ces subtilités d'optimisation dans la partie traitant de la mise en place d'un réseau de neurones en pratique.

2.4.4 Le réseau de neurones récurrent et le LSTM

Motivation pour la création d'une architecture récurrente Jeffrey L. Elman a été l'un des premiers à introduire les réseaux de neurones récurrents lorsqu'il a cherché un moyen de représenter avec précision des quantités fortement dépendantes du contexte avec un réseau de neurones [Elman, 1990]. Pour arriver à ce résultat, il a ajouté des unités dites de "contexte" dans un réseau de neurones classique (ANN), qui agissent comme des unités mémoire du réseau. Leurs valeurs sont une copie des valeurs des neurones présents dans les couches cachées. L'utilisation de ce type d'unités pour maintenir un état ou state permet, en théorie, de surpasser les performances d'un ANN dans des tâches telles que la régression sur des séries temporelles.

Les caractéristiques hydrologiques comme le débit sont des quantités qui dépendent fortement du temps et du contexte, c'est pourquoi leur modélisation peut, en théorie, bénéficier d'une structure de réseau récurrente au lieu d'une structure de réseau classique *en-avant* ou *feedforward*. De nombreuses variantes du RNN existent aujourd'hui, l'idée de base repose sur la notion d'état qui tient compte du contexte lors de la mise à jour des poids du réseau pendant la phase d'entraînement. En pratique, on considère surtout les réseaux de neurones Long Short-Memory (LSTM) qui sont une variante des RNN. On explique leurs spécificités dans la suite.

Remarque : On retrouve cette idée d'utiliser les valeurs passées d'une séquence pour la prédiction des suivantes dans la modélisation de processus autorégressifs avec la méthode ARIMA par exemple, souvent utilisée en finance.

Systèmes dynamiques Nous pouvons relier l'architecture des RNN à des systèmes dynamiques, car les équations sous-jacentes décrivant cette architecture peuvent être obtenues à partir d'une équation différentielle ordinaire non linéaire de premier ordre non homogène qui décrit l'évolution d'un signal d'état en fonction du temps t :

$$\frac{d\vec{s}(t)}{dt} = \vec{f}(t) + \vec{\phi}$$

Avec $\vec{f}(t) \in \mathbb{R}^d$ et $t \in \mathbb{R}^+$, $\vec{\Phi}$ constant $\in \mathbb{R}^{d \times d}$. f représente une fonction potentiellement non linéaire de l'état s et de l'entrée du réseau x . Il est montré dans [Sherstinsky, 2020] que la résolution de cette équation en utilisant la discrétisation d'Euler implicite (et en utilisant des hypothèses précises) donne le système suivant :

$$\begin{aligned}\vec{s}[n] &= W_s \vec{s}[n-1] + W_r \vec{r}[n-1] + W_x \vec{x}[n] + \vec{\theta}_s \\ \vec{r}[n] &= G(\vec{s}[n])\end{aligned}$$

Où W_s , W_r et W_x représentent des matrices de poids et $\vec{\theta}_s$ le vecteur des biais. On trouve en Fig8 une représentation schématique d'une cellule RNN.

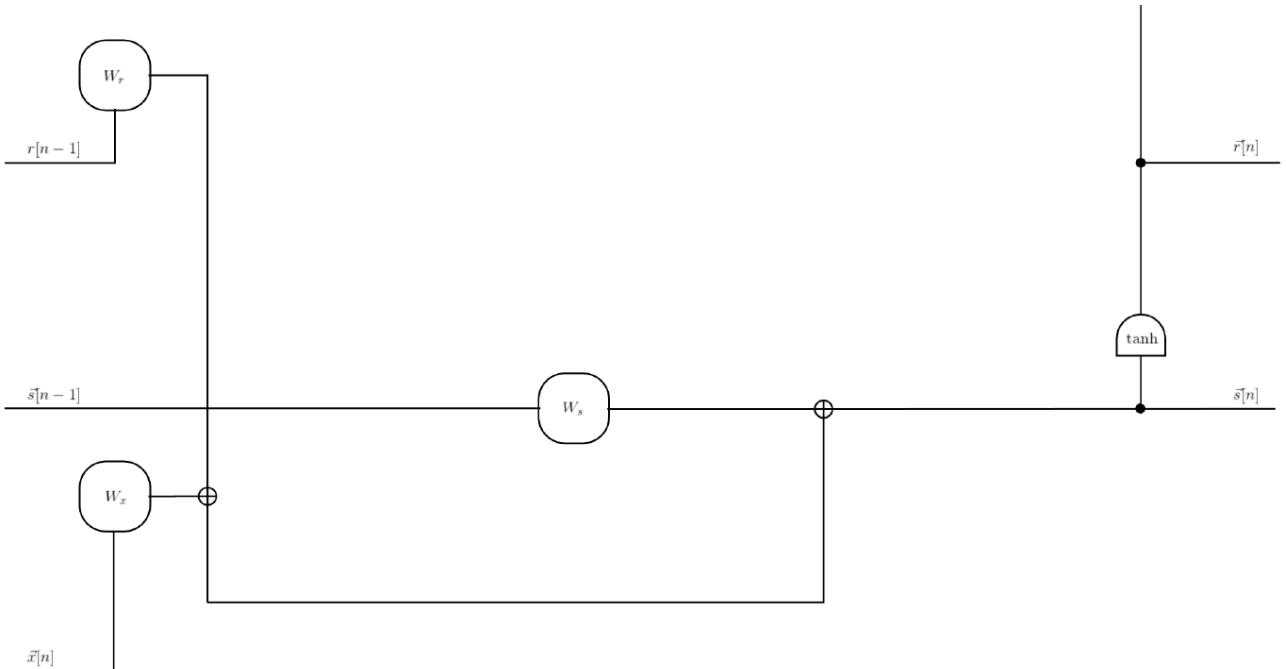


FIGURE 8: Illustration d'un RNN [Sherstinsky, 2020]

La phase d'apprentissage, comme pour les ANN, permet de trouver les meilleurs poids et biais pour mieux représenter la quantité ciblée. Ce qu'il faut retenir de ces équations, c'est que le signal d'état est calculé à partir du signal d'état

s , du signal d'état normalisé r du pas de temps précédent (d'où l'aspect "récurrent") ainsi que le vecteur d'entrée du pas de temps actuel. La procédure pour passer d'une représentation graphique cyclique à une représentation graphique acyclique d'un RNN est appelée "dépliement" ou "unfolding" et nous permet de conceptualiser un RNN comme étant une succession d'ANN (voir Figure 9). [Sherstinsky, 2020]. EN voyant le RNN comme une succession d'ANN il est facile de mettre en place une procédure d'apprentissage basé sur la rétropropagation du gradient de l'ANN. Cette variante que l'on ne détaille pas ici s'appelle la rétropropagation "dans le temps".

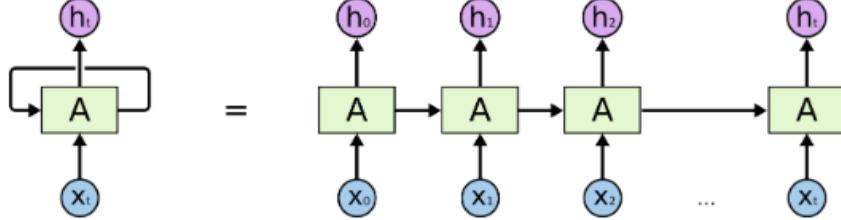


FIGURE 9: Illustration du déploiement d'un RNN. A représente un ANN, x_t l'observation au temps t d'une ou plusieurs variables explicatives et h_t la sortie de l'ANN associée à cette entrée. [Olah, 2020]

Un problème que l'on rencontre cependant avec une telle architecture est qu'elle n'est pas adaptée pour modéliser des interactions entre des observations très éloignées dans le temps.

Du RNN au LSTM Avec les techniques d'apprentissage classiques, la rétropropagation "dans le temps" par exemple, les valeurs du gradient de la fonction de coût ont tendance à disparaître (ou à exploser). [Hochreiter, 1998] montrent que cela conduit à des changements de poids insuffisants lors de l'apprentissage des dépendances à long terme. Sans entrer dans le détail des calculs, la formule de la chaîne appliquée successivement implique un produit successif sur la dérivée de la fonction d'activation. Sachant que les fonctions sigmoïdes ont une dérivée très proche de zéro (voir Figure 10), ce produit successif a tendance à tendre vers zéro lorsque l'on traite de dépendances lointaines dans le temps, donc quand le nombre de produits successifs est grand. C'est le problème de la disparition du gradient ou *vanishing gradient*.

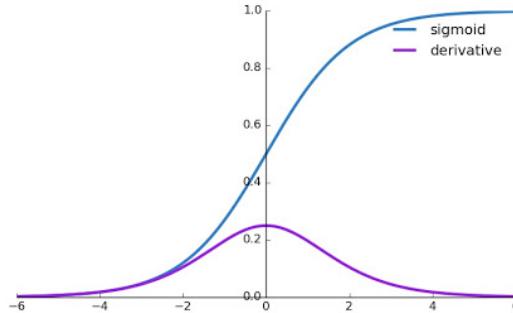


FIGURE 10: Graphe de la fonction logistique (sigmoid) et de sa dérivé.

L'architecture Long Short-Term Memory (LSTM), basée sur l'architecture RNN, s'attaque à ce problème en ajoutant des signaux de "porte" ou "gate" : $\vec{g}_{cs}(n)$, $\vec{g}_{cu}(n)$ et $\vec{g}_{cr}(n)$ [Hochreiter and Schmidhuber, 1997]. Les éléments de ces vecteurs sont fractionnaires et positifs. Les équations décrivant les réseaux LSTM sont les suivantes :

$$\begin{aligned}\vec{s}[n] &= g_{cs}(n) \odot (W_s \vec{s}[n-1] + W_r (\vec{g}_{cr}(n) \odot \vec{r}[n-1])) + g_{cu} \odot (W_x \vec{x}[n] + \vec{\theta}_s) \\ \vec{r}[n] &= G(\vec{s}[n])\end{aligned}$$

Où \odot représente le produit de Hadamard. On trouve en Figure 11 une représentation schématique d'une cellule LSTM.

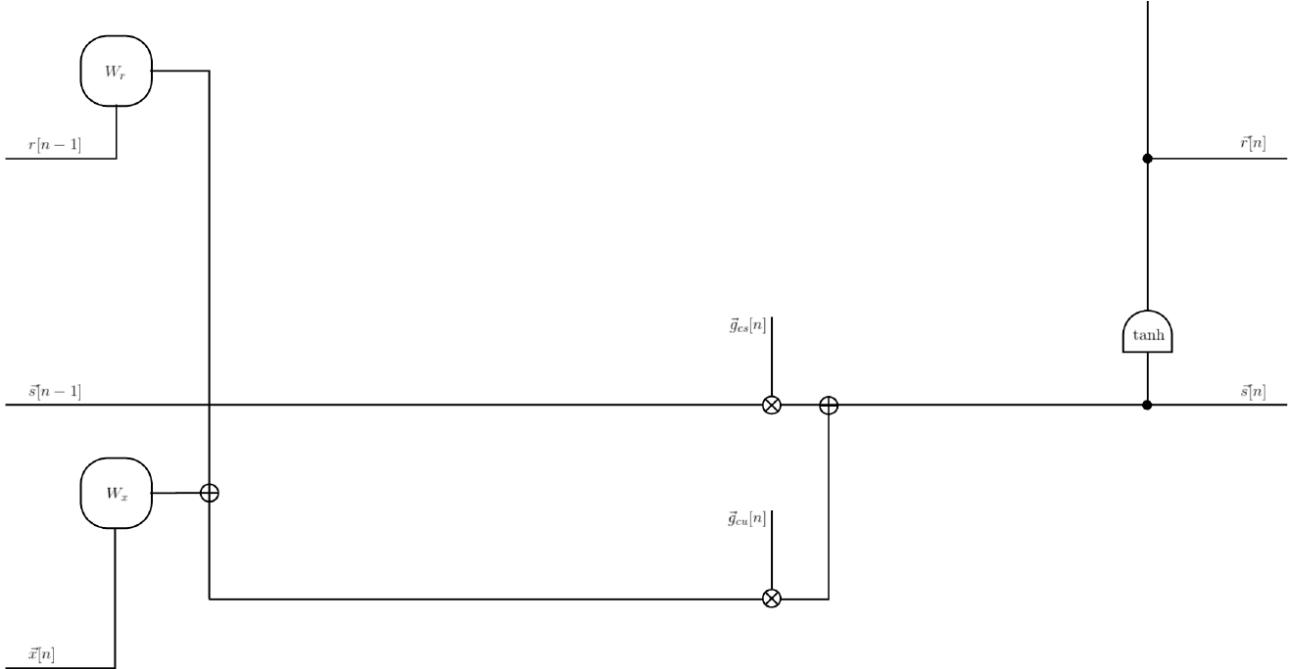


FIGURE 11: Illustration d'une cellule LSTM [Sherstinsky, 2020]

Ces signaux de porte $\vec{g}_{cs}(n)$, $\vec{g}_{cu}(n)$ et $\vec{g}_{cr}(n)$ contrôlent la quantité d'informations stockées dans l'état (là où il n'y a aucun contrôle dans un RNN) et leurs valeurs fractionnaires empêchent le problème de la disparition (explosion) du gradient de se produire. Ce résultat est purement calculatoire et repose, comme expliqué précédemment, sur les produits successifs qui apparaissent lors de l'application de la formule de la chaîne dans la rétropropagation du gradient. Le lecteur trouvera les détails de la démonstration dans [Hochreiter and Schmidhuber, 1997].

3 Données et outils utilisés

3.1 Présentation et analyse statistique des jeux de données

3.1.1 Définition de l'échelle et des variables étudiées

Avant d'entamer la description quantitative des jeux de données, définissons les échelles qui sont utilisées :

- Le *reach* (≈ 5 km), c'est l'échelle la plus large considérée pour les problématiques relatives à la mission SWOT.
- le *node* (≈ 200 m), c'est l'échelle de référence pour les modèles qui produisent des données "type SWOT". Les jeux de données présentés ici contiennent des mesures à l'échelle du *node*.

Les variables prises en compte lors de cette étude sont les suivantes :

1. Les mesures "type SWOT"
 - La pente S (cm/km) la surface de l'eau à l'échelle du *reach*
 - La hauteur Z ou *stage* (m) et la largeur W (m) de la surface de l'eau à l'échelle du *node*
2. Les variables hydrologiques standard
 - Le débit Q de la rivière étudiée (m^3/s)
 - L'aire drainée dans le bassin versant *flowacc* (m^2)
 - L'aire mouillée A (m^2)
 - La différence dA (m^2) entre l'aire mouillée à un instant t et l'aire mouillée non observable (elle correspond à l'élévation la plus basse)
 - Les variations d'élévations dH (m)
3. Les variables connexes dont on veut étudier l'influence sur la modélisation des variables hydrologiques d'intérêts
 - La constitution géologique du lit des rivières (%)
 - Le taux d'argile *clay*
 - Le taux de sable *sand*
 - Le taux de limon *silt*
 - Les classes de sols autour de la station de mesure
 - *LC1* à *LC4* : Taux de présence d'arbres
 - *LC5* : Taux de présence d'arbustes
 - *LC6* : Taux de présence d'herbacées
 - *LC7* : Taux de présence de terres cultivées
 - *LC8* : Taux de présence de terres souvent immergées
 - *LC9* : Taux de présence de constructions humaine
 - *LC10* : Taux de présence de neige ou de glace
 - *LC11* : Taux de présence de terres stériles ou arides
 - *LC12* : Taux de présence d'eau
4. Les variables d'identification des mesures (elles dépendent du jeu de données)
 - *river* : Le nom de la rivière
 - *reach* : Le reach dans lequel la mesure se situe (numéro)
 - *station_no* : L'identifiant de la station de mesure (pour HydroSWOT uniquement)
 - *day* : Le jour de la mesure
 - *lat* et *lon* : Les données de géolocalisation de la mesure (latitude, longitude)

On notera que toutes les variables sont quantitatives. Dans les deux sous-sections suivantes, nous menons une analyse statistique des deux jeux de données disponibles. Le but est de nous familiariser avec les données et leurs ordres de grandeur, d'identifier d'éventuels groupements ou *cluster* et les covariances évidentes entre certaines variables. Ces informations sont capitales pour mener à bien l'étape de modélisation.

3.1.2 PEPSI

La base de données PEPSI dont nous disposons est en fait la compilation des bases de données PEPSI construites lors du PEPSI challenge [Durand et al., 2016]. Cette base de données est composée d'une compilation de valeurs de débits calculées par plusieurs modèles hydrauliques, on considère qu'ils représentent une dynamique d'écoulement réelle. Les valeurs des observations "type SWOT" (caractéristiques géométriques de la rivière) sont aussi issues de ces modèles. On se place dans un contexte idéalisé, car aucune erreur n'a été ajoutée aux sorties des modèles.

Les données de la base PEPSI comprennent des données sur 31 rivières (Américaines, Européennes et Asiatiques). Certaines rivières sont divisées en plusieurs jeux de données, par exemple un pour l'amont et l'autre pour l'aval de la rivière.

Étude des corrélations Étudions en premier lieu les corrélations remarquables à l'échelle du jeu de données complet, c'est-à-dire en ne distinguant pas les régions géographiques.

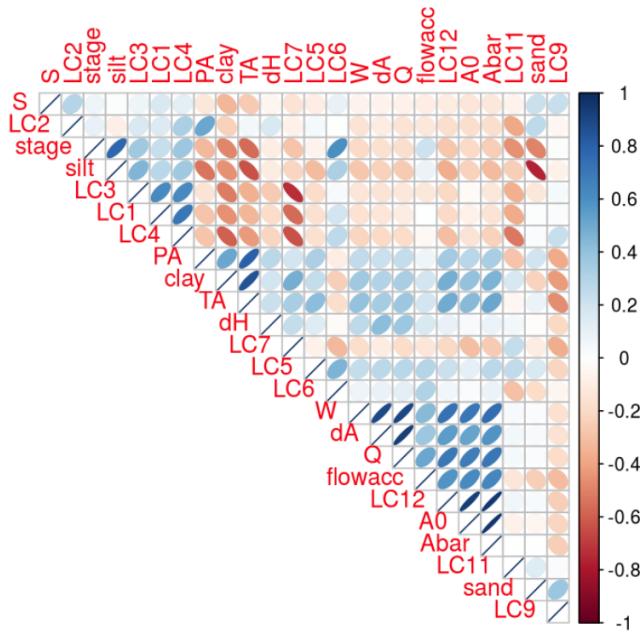


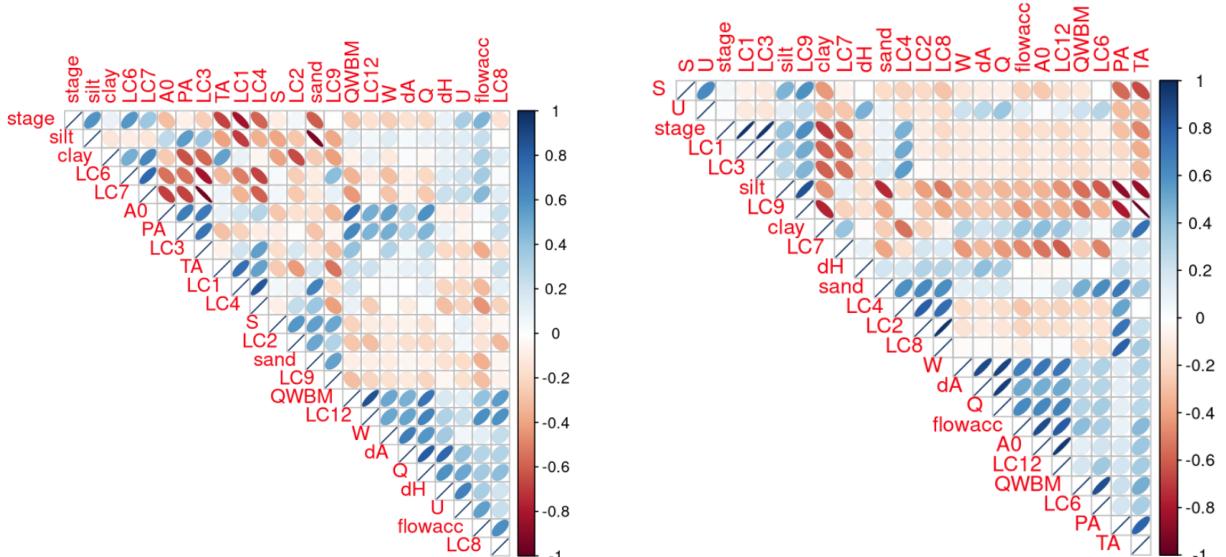
FIGURE 12: Matrice des corrélations pour le jeu de données PEPSI

De la matrice de corrélation en Figure 12 nous pouvons identifier plusieurs clusters de variables présentant une forte corrélation. Corrélation positive :

1. Forte corrélation positive : $W, dA, Q, A0, Abar, LC12$
2. Corrélation positive notable : $stage$ avec $silt$, TA avec PA et $clay$, $LC4$ avec $LC1$ et $LC3$

On retrouve dans le *cluster* le plus évident les variables hydrologiques rentrant dans la modélisation physique des écoulements, auquel vient s'ajouter la variable de classe de sol $LC12$ qui quantifie le taux de présence d'eau. On remarque cependant que la variable $stage$ n'est que très peu corrélée avec les autres variables hydrologiques ; on la retrouve plus fortement liée avec la variable de sol $silt$.

On compare maintenant la matrice de corrélation du jeu de données pris sur les rivières américaines et asiatiques séparément.

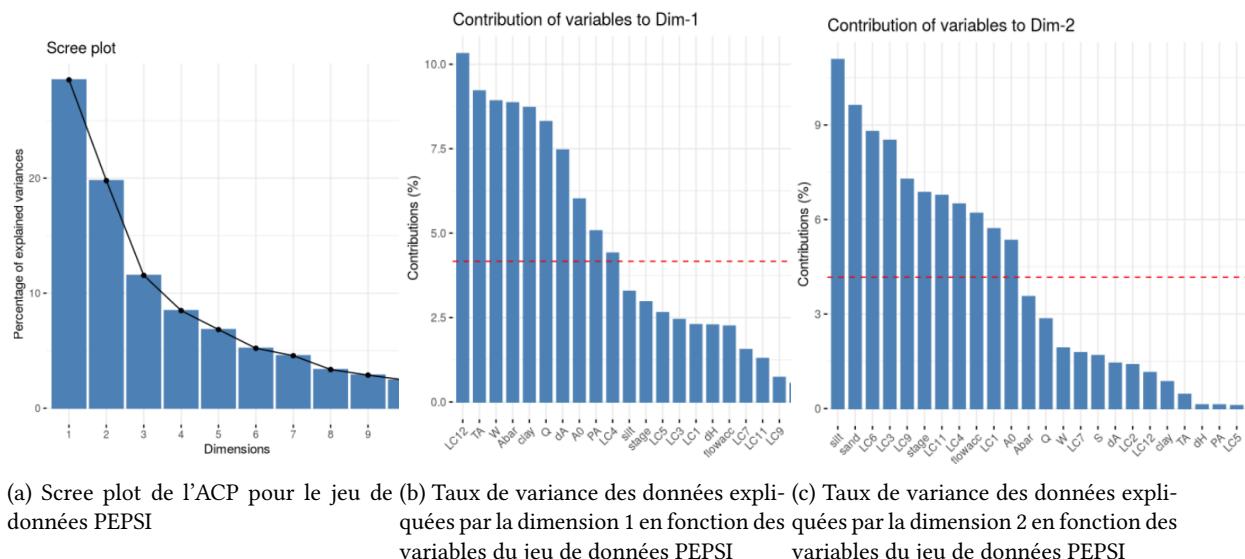


(a) Matrice de corrélation pour les rivières Américaines du jeu de données PEPSI (b) Matrice de corrélation pour les rivières Asiatiques du jeu de données PEPSI

FIGURE 13: Comparaison des matrices de corrélation sur deux partition du jeu de données : amérique et asie

On remarque directement en Figure 13 des corrélations plus marquées, témoignant de la variabilité dans l'espace des phénomènes hydrologiques et de la diversité du jeu de données. Du point de vue de la modélisation, ces informations nous indiquent qu'à l'échelle mondiale on retrouve bien l'importance des variables d'intérêts classiques qui permettent habituellement de calculer un débit. Cette variation des corrélations entre variables existantes entre les régions nous met en garde contre l'éventuel surapprentissage qui pourrait découler d'une mauvaise répartition des individus dans le jeu de donnée d'entraînement.

Identifications des clusters L'étude de la répartition des éventuels *clusters* dans les données est faite via une analyse en composantes principales (ACP). L'ACP fait partie des méthodes de réduction de dimensionnalité qui trouvent un grand intérêt dans énormément d'applications, notamment en statistique et en compression par exemple. Le principe est le suivant : on dispose d'un jeu de données à m variables et n individus que l'on se représente comme un nuage de points en m dimensions. Le but de l'ACP est de trouver les "directions" portant le maximum de variance à travers le nuage de points et de le représenter selon ces directions. Sachant que l'on dispose de m variables, on peut voir ces directions ou *composantes* comme des méta variables : ce sont des variables qui représentent plusieurs variables à la fois. En pratique, lorsque les données ne sont pas totalement indépendantes entre elles, une ACP peut permettre de se ramener à une représentation en 2 dimensions d'un jeu de données composé d'un très grand nombre de variables sans perdre beaucoup d'informations sur un *biplot*. Le cercle de corrélation, conjointement au *scree plot* (graphe représentant le taux de variance portée par chaque composante calculée par l'ACP), nous permet de comprendre comment les directions sont construites à partir des m variables de départ.



Le scree plot de la base de données PEPSI (voir Figure 14a) nous montre que la majorité de la variance des données peut s'expliquer avec peu de composantes, en effet 2 composantes suffisent à expliquer 50% de la variance totale. On fait donc le choix de n'utiliser que 2 composantes pour afficher les données. Les Figure 14b et Figure 14c nous indiquent la contribution de chacune des variables à ces composantes, on retrouve également cette information sur le cercle des corrélations qui a l'avantage d'indiquer si la corrélation d'une variable à la composante étudiée est positive ou négative. Cette information nous est très utile pour avoir une idée claire de ce que représente le *biplot* ci-dessous Figure 15. Les plus grandes contributions à la composante 1 sont des variables très corrélées au débit (*LC12*, *TA*, *Abar*, *W*). La composante 2 quant à elle n'est quasiment constituée que de contributions de variables liées au sol : on en trouve 8 dans les 10 premières. On peut donc interpréter la composante 1 comme une méta variable classant les rivières par géométrie croissante et la composante 2 comme celle discriminant les rivières en fonction de leurs caractéristiques géologiques.

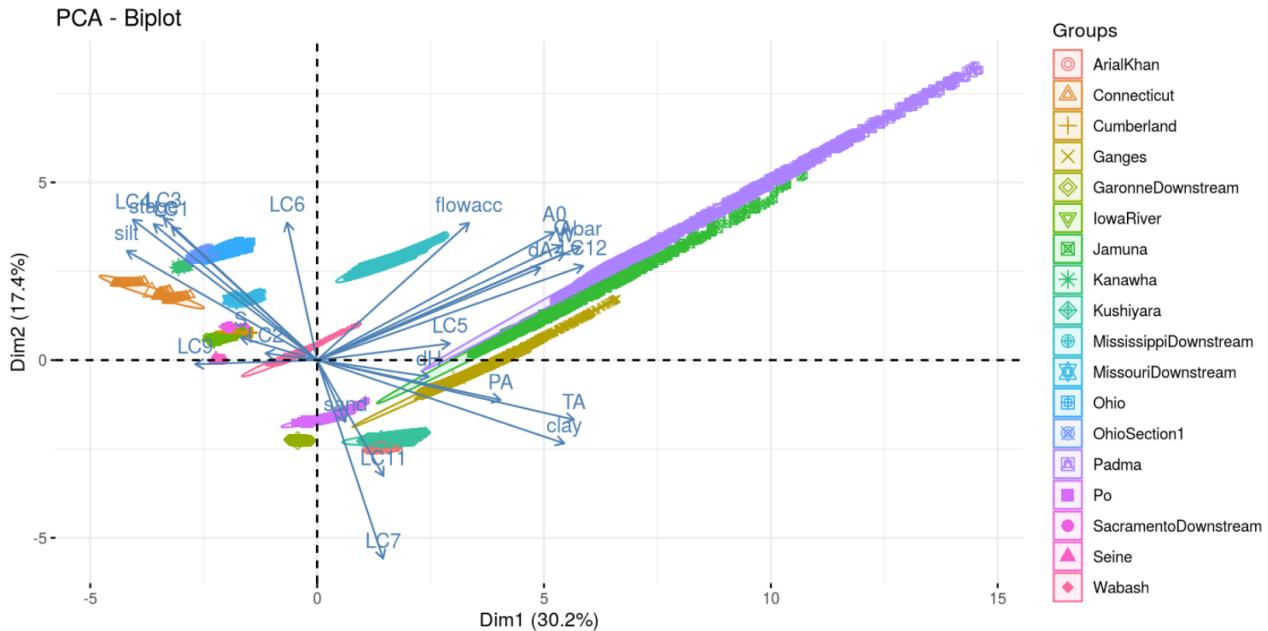


FIGURE 15: *Biplot* de l'ACP sur le jeu de données PEPSI

On retrouve dans ce *biplot* : les rivières argileuses de grande taille et à fort débit où on mesure des températures élevées dans le quart supérieur droit comme le Mississippi ou Padma. Dans le quart inférieur droit les petites rivières à faible débit localisées dans des endroits chauds et argileux comme la rivière Kushiyara. Dans le quart supérieur gauche, on retrouve les rivières à fort débit et à sol composé de limon et entouré d'arbres. On retrouve dans ce quart la grande majorité des rivières Américaines.

On tire de cette ACP que la géométrie de la rivière représente effectivement facteur discriminant entre la majorité des individus, mais que les caractéristiques géologiques de la rivière et des alentours joue un rôle important. On peut donc faire l'hypothèse qu'ajouter de telles variables à notre futur modèle.

3.1.3 HydroswoT

On réitère dans cette sous-section l'analyse statistique faite dans la sous-section précédente sur la base de données HydroSWOT. Contrairement au jeu de données PEPSI qui était composé d'une compilation de sorties de modèles hydrologiques, la base HydroSWOT est constituée de mesures réelles sur les mêmes variables. On retrouve cependant des données uniquement sur des rivières américaines (132).

Étude des corrélations On remarque tout de suite sur la matrice de corrélation en Figure 16 du jeu de données HydroSWOT que les corrélations entre les variables sont bien moins marquées que lors de l'étude du jeu de données PEPSI, bien que nous soyons ici dans le cas de figure où toutes les données sont spatialement proches (Amérique). On déduit assez rapidement que les données issues de modèles sont par essence corrélées de manière plus évidente que les observations réelles étant donné que l'on considère des conditions de simulation idéalisées.

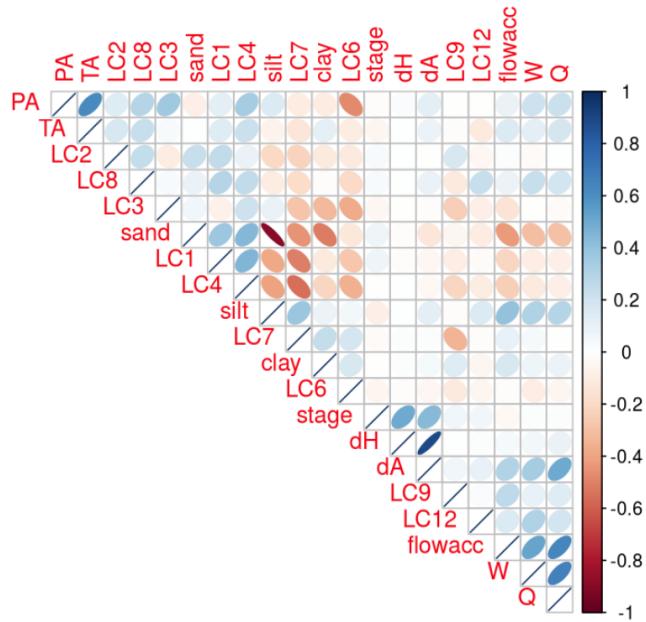


FIGURE 16: Matrice de corrélation sur le jeu de données HSWOT

Identification des clusters Malgré le fait qu'ici nous n'ayons ici que des données issues de rivières Américaines, il n'est pas aberrant de réaliser une ACP. Ici cependant, avec 132 rivières (individus) différentes, nous n'avons pas la possibilité de tirer parti de toutes les informations présentes dans le *Biplot* de l'ACP. De manière analogue à la sous-section précédente, nous commençons notre ACP par la construction des métavariables.

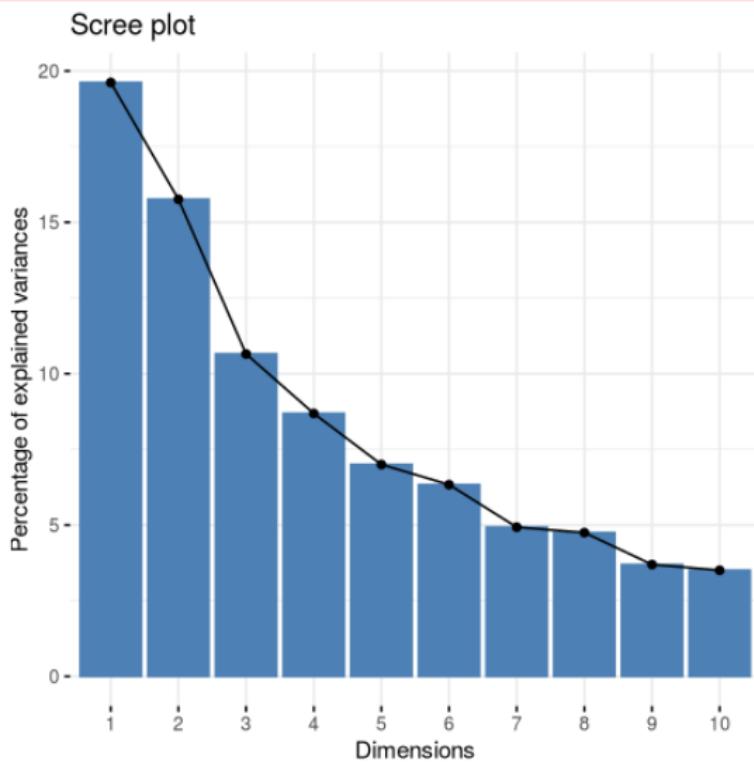
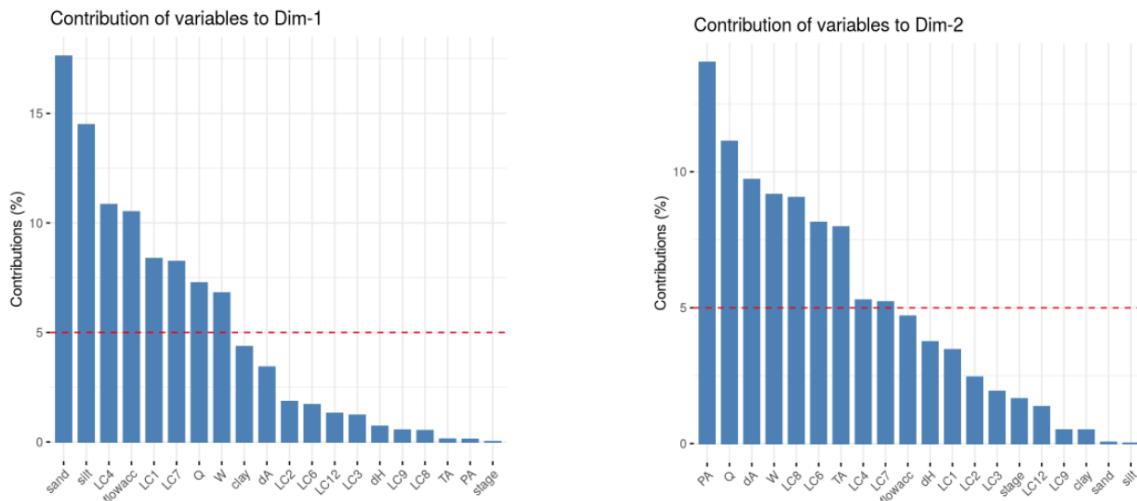


FIGURE 17: Scree plot de l'ACP pour le jeu de données HSWOT

On remarque directement en Figure 17 que la répartition de la variance dans les données à travers les métavariables n'est pas aussi marquée que pour le jeu de données PEPSI. C'est un premier indice que les données sont globalement proches dans l'espace des variables explicatives, et il concorde avec l'observation réalisée plus tôt sur la matrice de corrélation du jeu de données. On peut expliquer quasiment de 50% de la variance totale avec les 2 premières métavariables. Analysons maintenant la signification de ces 2 métavariables.



(a) Taux de variance des données expliquées par la dimension 1 en fonction des variables du jeu de données HSWOT
(b) Taux de variance des données expliquées par la dimension 1 en fonction des variables du jeu de données HSWOT

FIGURE 18: Contributions des variables aux dimensions 1 et 2 de l'ACP

On retrouve dans ces résultats les mêmes idées que dans l'ACP du jeu de données PEPSI. La première métavariable (voir Figure 18a) semble relative à la composition géologique de la rivière tandis que la seconde (voir Figure 18b) donne beaucoup de poids aux caractéristiques hydrologiques et climatiques.

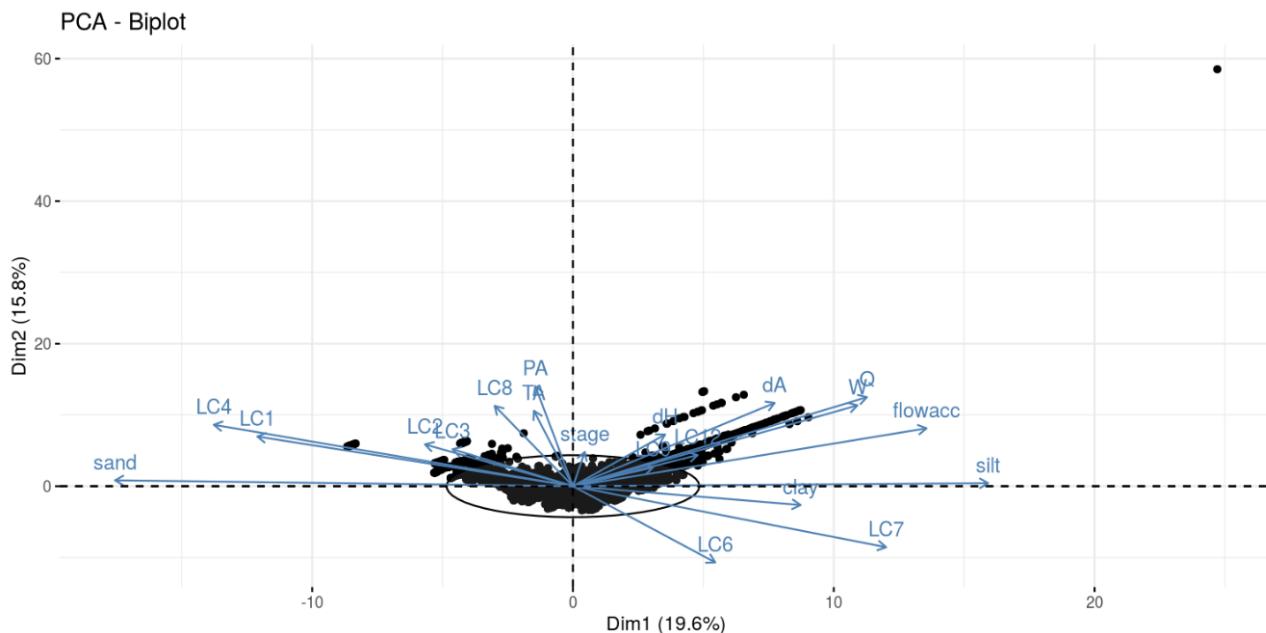


FIGURE 19: Biplot de l'ACP sur le jeu de données HSWOT

Comme précisé plus tôt, il n'est pas possible d'analyser la répartition des 132 individus sur le biplot simultanément. On peut néanmoins se rendre compte de la répartition globale du nuage de point. Ici ce qu'on doit retenir de la Figure 19 est que bien qu'une plus grande partie de la variance du jeu de données semble être expliquée par la métavariable hydrologique/climatique, l'importance des variables géologiques n'est pas négligeable. On suit donc la conclusion de la première analyse pour le jeu de donnée PEPSI qui est qu'inclure de telles variables dans les futures modélisations par réseaux de neurones paraît être un choix judicieux.

3.2 Présentation de l'environnement de programmation

La mise en place des réseaux de neurones, ainsi que toute la partie prétraitement des données et analyses des résultats est faite avec Python. Dans cette courte section, on présente les différentes librairies et leur intérêt pour notre problème de modélisation.

3.2.1 Traitement des données : Pandas, Numpy

Pandas et Numpy sont deux librairies *open source* respectivement axé sur la science des données et le calcul scientifique. Elles permettent une manipulation complète des données structurées sous forme de tableau (matrice, tenseur, etc.). De plus ces librairies implémentent des méthodes de manipulations de tableau dont les routines sous-jacentes sont précompilées en C, ce qui permet un gain très significatif en vitesse de calcul par rapport à du code Python pur.

3.2.2 Réseaux de neurones : Keras, note sur PyTorch

La bibliothèque choisie pour implémenter en python des réseaux de neurones est la librairie Keras [Keras, 2020]. Keras est une bibliothèque open source conçue pour permettre une expérimentation rapide des réseaux neuronaux profonds. Keras sert d'interface pour la bibliothèque *TensorFlow*, bibliothèque open source développée par Google pour des applications tournées vers le machine learning, mais dont la complexité n'en permet pas une prise en main rapide. Keras fait maintenant partie intégrante de la bibliothèque Tensorflow.

Keras contient de nombreuses implémentations de blocs de construction de réseaux neuronaux couramment utilisés tels que des couches, des fonctions de coûts, des fonctions d'activation, des "optimiseurs" et une grande quantité d'outils pour faciliter la mise en place rapide de tout type de réseaux de neurones. En plus des réseaux neuronaux standard, Keras prend en charge les réseaux neuronaux convolutifs et récurrents. Il prend en charge d'autres couches spéciales comme le *dropout*, la normalisation par lots et le *pooling*. Il permet également d'utiliser l'apprentissage distribué de modèles profonds sur des *clusters* d'unités de traitement graphique (GPU) et d'unités de traitement des tenseurs (TPU).

La librairie open source analogue PyTorch, développée par le laboratoire de recherche en intelligence artificielle de Facebook (Facebook's AI Research lab 'FAIR'), propose des fonctionnalités similaires, mais une prise en main plus difficile pour un non initié c'est pourquoi Keras a été choisi. Il est intéressant de noter le couple Keras/Tensorflow, leader actuel sur le marché des librairies orientées machine learning, perd en popularité dans le domaine de la recherche là où PyTorch est en constante croissance de popularité. La Figure 20 met en exergue l'évolution de popularité des deux librairies dans la recherche.

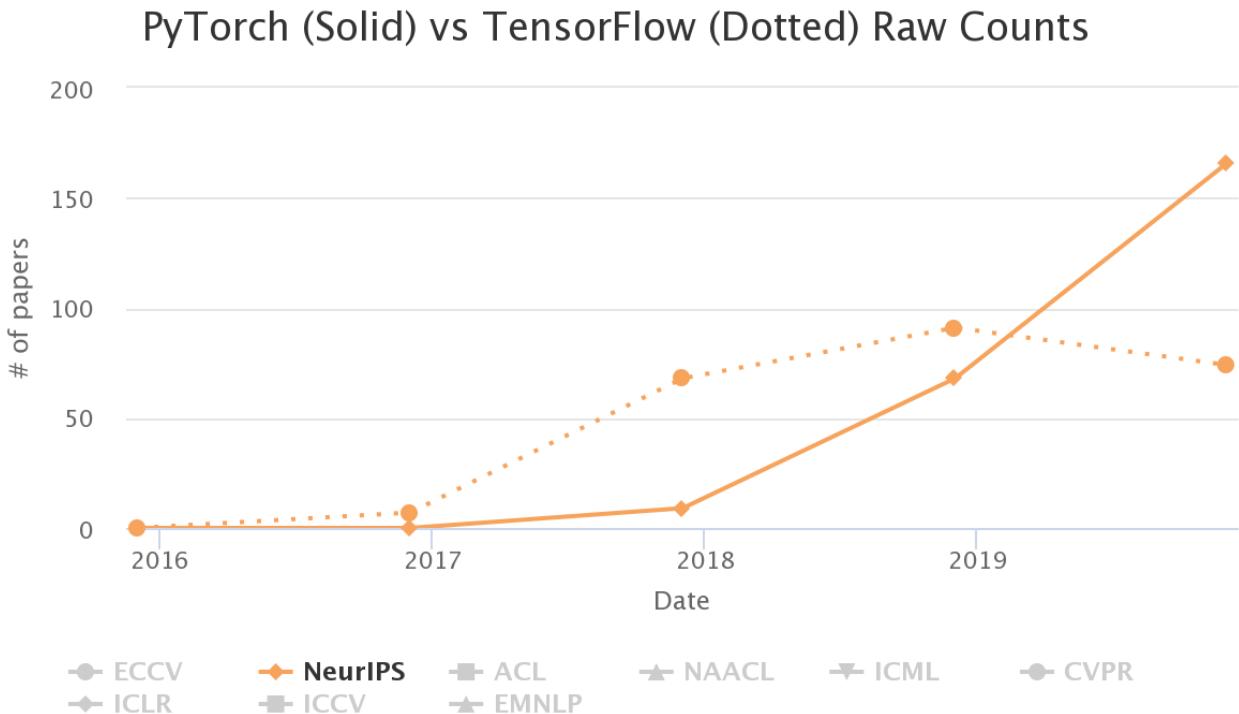


FIGURE 20: Utilisation de Keras et de Pytorch dans les articles de la conférence NeurIPS [horace, 2020]

4 Méthodologie

Dans cette section on détaille le processus de mise en place d'un réseau de neurones du prétraitement des données jusqu'à l'analyse des résultats. On sépare les architectures *feedforward* (ANN) et *recurrent* (LSTM) en 2 sous-sections bien que dans son ensemble, le processus soit très similaire.

4.1 Mise en place d'un ANN

Prétraitement des données Comme nous l'avons expliqué précédemment, une modélisation reposant sur l'apprentissage supervisé a besoin pour "s'entraîner" d'un jeu de données X des valeurs des variables explicatives pour un nombre n d'individus, ainsi que la valeur de la variable cible ou réponse Y correspondante. Dans notre cas, le prétraitement se limite à la division du jeu de données initial en plusieurs jeux de données : *Train*, *Test*. Aucun traitement particulier n'est ici nécessaire sur les variables du jeu de données pour les rendre utilisables. Le lecteur trouvera en Annexe A les notebooks contenant le code correspondant aux processus décrits ci-dessous. On sépare le travail de prétraitement en 5 étapes :

1. Nettoyage du dataset (suppression des valeurs aberrantes et suppression des individus si une valeur est manquante)
2. Suppression des variables (colonnes) qui ne nous intéressent pas (données de géolocalisation par exemple)
3. Sélection des variables explicatives et création du dataset X
4. Sélection de la variable cible et création du dataset Y
5. On applique une fonction de mise à l'échelle sur les données pour qu'elles soient uniformément petites en valeur absolue.
6. On sépare les données aléatoirement en 2 jeux de données : "Train" et "Test" avec une proportion 80/20.

Note sur la partie scale : On remarque en pratique une convergence plus rapide de l'entraînement d'un réseau de neurones lorsque la distribution des variables d'entrée est normale centré réduite ou à minima centré réduite. Les raisons théoriques à ce phénomène sont à ce jour mal connues.

En ce qui concerne la séparation de notre jeu de données en jeux de données *Train* et *Test*, on utilise une méthode de la librairie *ScikitLearn* qui nous permet de choisir la proportion des données que l'on souhaite avoir dans notre jeu de donnée *Test*. On se sert aussi beaucoup de la fonctionnalité du *seed* qui permet de contrôler la partie aléatoire du tirage des individus du dataset. En gardant en mémoire le *seed* de la séparation, on peut reconstruire *a posteriori* les deux jeux de données à l'identique, par exemple au moment d'analyser les résultats d'un modèle si cette analyse ne se fait pas en même temps que l'entraînement du modèle.

Création du réseau de neurone Nous l'avons vu dans la section 2.4, les paramètres d'un réseau de neurones (dans le cas simple de l'ANN du moins) sont le nombre de couches ainsi que le nombre de neurones par couches. Ensuite, chaque couche est paramétrable par une fonction d'activation et éventuellement des *kernel* d'initialisation des poids. Enfin, le modèle se paramètre par "l'optimiseur" et la fonction de coût. L'avantage de Keras est que le lien entre la structure théorique du réseau de neurone est le code est direct. On créé donc en quelques lignes un modèles respectant la paramétrisation décrite à l'instant et on code facilement une fonction qui va compiler et lancer l'entraînement du réseau de neurones grâce aux méthodes implémentés directement dans Keras. Le code, très simple, est présenté en Annexe A. On aborde dans la sous-section suivante la question de la sélection de ces paramètres.

4.1.1 Sélection des paramètres

Une fois que l'on a une structurer notre modèle avec Keras il faut choisir la valeur les paramètres. Dans notre cas les paramètres sont :

1. Les paramètres du réseau de neurones : le nombre de neurones et de couches
2. La méthode d'optimisation utilisée pour effectuer la rétropropagation du gradient
3. Le nombre d'epochs et/ou la contrainte d'arrêt de l'entraînement du réseau de neurones

On peut répondre facilement aux questions relatives à la méthode d'optimisation et du nombre d'epochs à choisir étant donnée la littérature dense présente à ce sujet. Dans notre cas on considère la méthode d'optimisation *ADAM* [Kingma and Ba, 2017] dont les avantages par rapport à la méthode de gradient de descente classique en font un choix évident. On notera en particulier les avantages suivants :

- Facile à mettre en oeuvre, on trouve cette méthode dans les optimiseurs implémentés par défaut dans Keras.

- Efficace d'un point de vue temps de calcul et espace mémoire, car c'est une méthode de descente stochastique. Cela veut dire que chaque epoch ne nécessite pas d'utiliser la totalité du dataset d'entraînement, mais uniquement une sous partie de ce dernier, tiré aléatoirement avec remise.
- Adapté aux phénomènes non stationnaires.

Ensuite, le choix d'un nombre efficace d'epochs, c'est-à-dire adaptés à la vitesse de convergence de la fonction de coût, s'obtient après quelques tests. On restera cependant attentif à la décroissance de la fonction de coût lors des entraînements du réseau de neurones pour ajuster si besoin la valeur du nombre d'epochs. Ici 500 epochs ont semblé être un choix judicieux après plusieurs essais.

En ce qui concerne la question du nombre de neurones et de couches, il est plus difficile de trouver une réponse dans la littérature. On se contente donc de faire plusieurs tests et d'analyser l'influence de la valeur de ces paramètres sur les résultats. On note assez rapidement qu'une architecture profonde (32-64 couches) se débrouille mieux qu'une architecture peu profonde (4-8 couches) et termes d'indicateurs de performances. De même, un grand nombre de neurones sur chaque couche (64) semble plus performant.

Une fois les paramètres choisis, avec la possibilité de les ajuster rapidement si besoin dans le code, on peut se lancer dans les tests et faire varier ce qui nous intéresse c'est à dire : les variables explicatives.

4.2 Mise en place d'un LSTM

Dans cette sous-section on se concentre principalement sur la partie prétraitement où une étape intermédiaire vient s'ajouter au prétraitement qu'on effectue lorsque l'on modélise un simple ANN. La grande modularité de Keras nous permet de réutiliser tout le code de l'ANN et de ne changer que quelques lignes pour remplacer les couches de neurones par des unités LSTM (voir Annexe B)

4.2.1 Prétraitement

On rappelle que le LSTM est une architecture qui fait partie de la famille des réseaux de neurones récurrents, adaptés au traitement de séries temporelles. À la différence d'une architecture ANN, on va ici utiliser non pas la valeur des variables sur un seul instant, mais sur une séquence de valeurs. De façon à garder la même structure de jeux de données que dans toute méthode d'apprentissage supervisé il faut ajouter une étape entre les étapes 5 et 6 du prétraitement effectué pour une ANN décrite dans la sous-section précédente. Cette étape porte le nom de *series to supervised*. Le principe est d'appliquer une fenêtre glissante sur la séquence temporelle de valeurs pour construire un jeu de données dont chaque individu sera représenté par la valeur d'une ou plusieurs variables d'intérêts sur la plage temporelle couverte par la fenêtre glissante (voir Figure 21). On se retrouve donc, dans le cas où on veut utiliser plusieurs variables dans notre modélisation avec un jeu de données d'entraînement et de test en 3 dimensions.

De manière plus formelle, on procède de la façon suivante :

1. On choisit la taille j de la fenêtre temporelle à utiliser pour effectuer les prédictions
2. On itère sur chaque pas temporel de la séquence de données (ici les jours)
3. On applique la fenêtre temporelle et on récupère les j valeurs des variables. On obtient donc une matrice de taille $j \times m$ avec m le nombre de variables qui jouera le rôle de la matrice X .
4. On récupère la valeur au pas de temps suivant de la variable que l'on veut prédire et on l'ajoute dans la dernière colonne qui jouera le rôle du vecteur Y .

On se retrouve alors avec un jeu de données en 3 dimensions (n, j, m) avec n le nombre de fois que l'on peut faire glisser la fenêtre de taille j jours sur la totalité de la séquence de valeur disponible.

Remarque : Par manque de temps nous n'utilisons dans notre cas les valeurs d'une seule rivière à la fois, sur une durée d'1 an.

4.2.2 Modélisation

De façon analogue à la modélisation d'un ANN via la librairie Keras, nous pouvons facilement mettre en place un LSTM paramétrables grâce aux unités LSTM déjà implémentées dans la librairie. La façon de compiler de lancer l'entraînement reste la même que pour l'ANN (voir Annexe A). On procède de la même façon que pour l'ANN pour déterminer les paramètres les plus judicieux.

4.3 Évaluer efficacement les performances

Pour tout problème de modélisation, il est important d'automatiser la procédure d'analyse des résultats pour être efficace au cours des potentiels multiples tests. Ici, on explique par quelle procédure les résultats des divers tests d'ANN et de LSTM ont été traités pour avoir une idée claire de la performance des modèles sous-jacents.

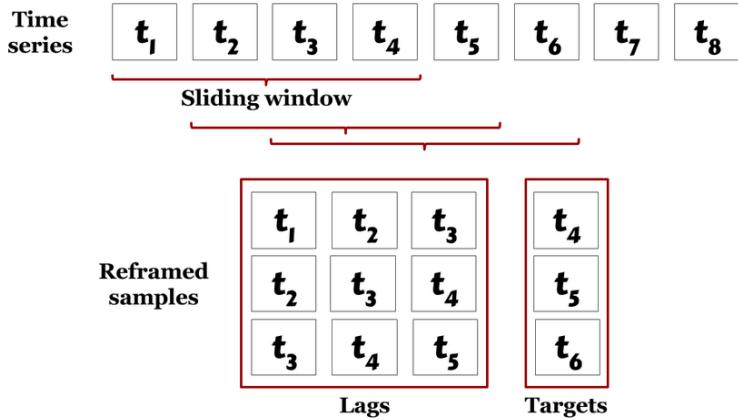


FIGURE 21: Illustration du processus *series to supervised*

4.3.1 Exporter le modèle

Une force de Keras est de proposer des méthodes permettant l'import et l'export des poids et biais du modèle que l'on vient d'entraîner facilement. On se sert de la méthode d'export pour sauvegarder nos modèles entraînés dans des dossiers. On prend bien soin d'automatiser la procédure en fin de chaque entraînement tout en paramétrant le nom des fichiers de sauvegarde par les valeurs de paramètres du modèle ainsi que les variables explicatives utilisées pour mieux s'y retrouver ensuite.

4.3.2 Import du modèle et recréation des dataset d'entraînement et de test

Dans un script ou *notebook* destiné à l'analyse des résultats, on met en place une routine d'import des modèles entraînés qui permet, connaissant les paramètres ainsi que les variables utilisées, d'importer n'importe quel modèle. On utilise donc la méthode d'import de modèle proposé par Keras combiné à une fonction de filtrage des modèles déjà enregistrés pour importer facilement un modèle. Avec les informations dont on dispose dans le nom du fichier de modèle, on peut ensuite reconstruire les jeux de données *Train* et *Test*. On rappelle au lecteur que la fonction utilisée pour effectuer la séparation du jeu de donnée de travail en deux jeux de données *Train* et *Test* aléatoirement prend en paramètre un *seed* qui permet de contrôler l'aléa. On peut donc facilement reproduire les mêmes jeux de données qui ont été utilisés lors de l'entraînement pour notre analyse.

4.3.3 Calcul des indicateurs

Une fois que l'on dispose de notre modèle entraîné ainsi que de nos deux jeux de données *Train* et *Test*, on peut effectuer des prédictions et analyser les résultats. Dans notre cas nous calculons les prédictions à partir du jeu de données *Test*. Nous pouvons ensuite calculer la valeur des indicateurs qui nous permettent d'évaluer la performance des modèles.

Un indicateur propre à l'hydrologie Dans un premier temps on définit l'indicateur *Nash–Sutcliffe model efficiency coefficient* (NSE) qui sera utilisé par la suite. Cet indicateur, proposé par [Nash and Sutcliffe, 1970] permet d'évaluer les performances de prédiction d'un modèle hydrologique et il est défini comme suit :

$$NSE = 1 - \frac{\sum_{t=1}^T (Q_m^t - Q_o^t)^2}{\sum_{t=1}^T (Q_o^t - \bar{Q}_o)^2}$$

Avec \bar{Q}_o la moyenne des débits observés, Q_m le débit calculé par le modèle et Q_o^t le débit observé à l'instant t .

Dans le cas d'un modèle parfait avec une variance d'erreur d'estimation égale à zéro, le NSE est égal à 1. Inversement, un modèle qui produit une variance d'erreur d'estimation égale à la variance de la série temporelle observée produit un NSE de 0. En réalité, NSE = 0 indique que le modèle a la même capacité prédictive que la moyenne des séries temporelles en termes de somme des erreurs au carré. Dans le cas d'une série temporelle modélisée dont la variance de l'erreur d'estimation est significativement plus grande que la variance des observations, le NSE devient négatif. Une efficacité inférieure à zéro (NSE < 0) se produit lorsque la moyenne observée est un meilleur prédicteur que le modèle. Des valeurs de NSE plus proches de 1 suggèrent un modèle avec une plus grande capacité prédictive.

On calcule la valeur de cet indicateur pour chaque rivière, mais on étudie aussi la répartition de ses valeurs pour chaque individu au travers d'une boîte à moustaches (*boxplot*).

Les indicateurs classiques

1. Coefficient de corrélation de Pearson : C'est un indicateur intéressant, car il nous permet de savoir rapidement si la relation entre les valeurs prédictées et les valeurs réelles est linéaire. Une corrélation égale à 1 indique que les prédictions sont exactement égales à la réalité. C'est un indicateur que l'on étudie à l'échelle de l'individu. Il est défini comme suit :

$$\rho_{X,Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

avec :

- σ_X l'écart type de X (ici X représente les données prédictées)
- σ_Y l'écart type de Y (ici Y représente les données réelles)
- μ_X l'écart type de X
- μ_Y l'écart type de Y

2. Erreur absolue moyenne : Elle est définie comme suit :

$$MAE = \frac{\sum_{i=1}^n |\hat{y}_i - y_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

Avec, dans notre cas, n le nombre d'observations disponibles pour un individu, \hat{y}_i la valeur d'une prédiction et y_i la valeur réelle.

On calcule la valeur de cet indicateur pour chaque rivière puis on étudie la répartition de ses valeurs au travers d'une boîte à moustaches. Une valeur moyenne proche de 0 indique que la prédiction est très précise, on s'intéresse cependant plus aux valeurs des quantiles de la répartition des valeurs de l'indicateur. Cette information nous permet de savoir si l'erreur est uniforme sur les individus, information capitale, car, on le rappelle, les individus sont parfois très distincts du point de vue de leurs caractéristiques hydrologiques et géologiques.

3. Écart quadratique moyen normalisé (NRMSE) : Elle est définie comme :

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

Avec n le nombre d'observations disponibles pour l'individu, \hat{y}_i la valeur d'une prédiction, y_i la valeur réelle et \bar{y} la moyenne des valeurs réelles pour l'individu.

Cet indicateur nous permet d'avoir une mesure d'erreur relative, qui n'est donc pas influencée par la variance au sein des individus. On calcule cet indicateur pour chaque rivière et, de même que pour le NSE et le MAE, pour tout le vecteur prédiction sans distinction d'individu et on affiche le résultat dans une boîte à moustaches pour les mêmes raisons que précédemment.

La présentation des résultats Pour rendre l'analyse complète, on met en forme les différentes valeurs d'indicateurs sous la forme de multiples graphes faciles à interpréter. Dans un premier temps, on affiche, sur une ligne, les boîtes à moustaches correspondantes à la répartition des valeurs de MAE, NRMSE et de NSE. En seconde ligne on détaille la répartition des valeurs d'indicateurs en fonction de l'individu, pour identifier les éventuels outliers. La Figure 22 propose un exemple d'affichage :

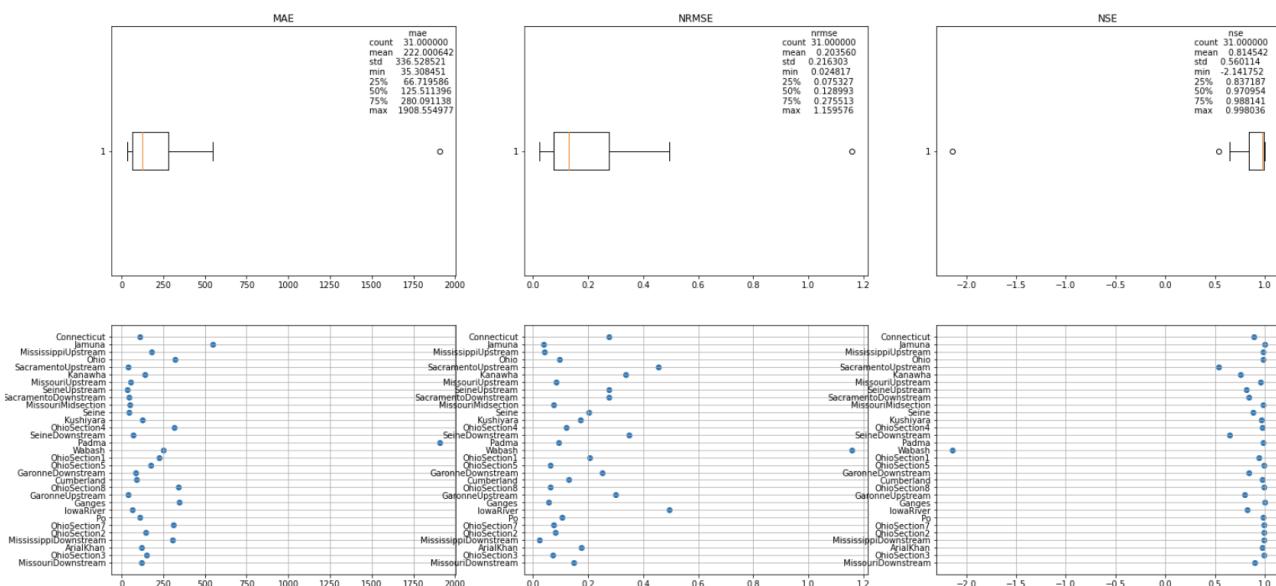


FIGURE 22: Exemple d'affichage des résultats globaux

Dans un second temps, on affiche, pour chaque rivière, 4 graphes.

1. Les prédictions en fonction de la valeur réelle, pour les jeux de donnée *Test* et *Train*, en utilisant une couleur différente pour les distinguer. On ajoute les coefficients de corrélation dans la légende. La droite identité est aussi tracée pour avoir une référence.
2. L'erreur relative en fonction de la valeur de la variable cible, uniquement avec les données de *Test*.
3. Deux boîtes à moustaches représentant la répartition de la valeur de la variable cible de la rivière en comparaison avec la répartition de la valeur de la variable cible dans tout le jeu de données. Ce graphe permet de savoir rapidement si on est en train étudier une rivière "extrême" en termes de caractéristiques hydrologiques ou pas.
4. La valeur de prédictions en fonction du temps en comparaison avec les valeurs réelles en fonction du temps (hydrogramme sur un seul *reach* dans le cas où on prédit le débit Q et la valeur de \bar{A} prédite en fonction du temps sur chaque *reach* de la rivière consécutivement)

On propose en Figure 23 un exemple d'affichage des résultats pour une rivière :

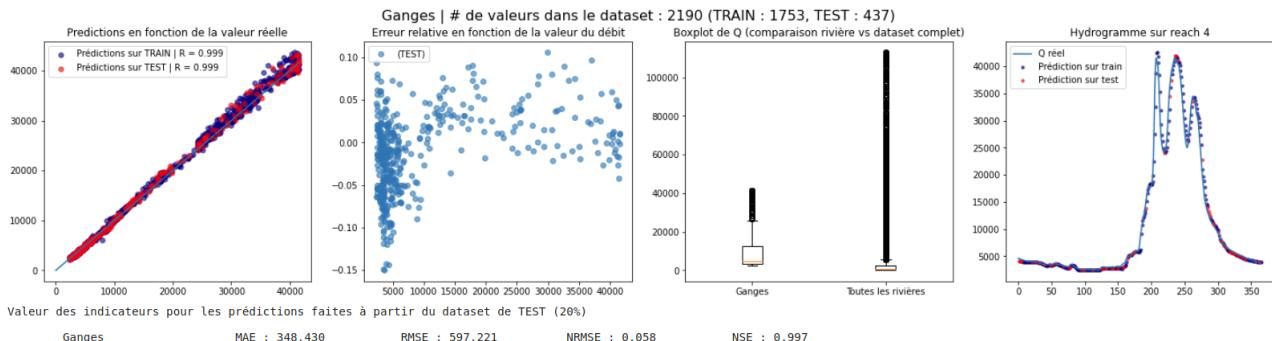


FIGURE 23: Exemple d'affichage des résultats pour une rivière

5 Résultats et analyse

Note préliminaire : Dans l'analyse suivante, nous nous concentrons sur les résultats fournis par les modèles utilisant comme données d'entrée les données PEPSI, pour deux raisons :

1. On observe un fort biais des prédictions pour les rivières ayant beaucoup d'observations dans le jeu de données HydroSWOT en comparaison avec un modèle analogue prenant les données PEPSI en entrée.
2. Le très grand nombre d'individus (132) ne permet pas une analyse efficace des indicateurs pris séparément sur les rivières.

Pour ces raisons, dont on a pris connaissance tôt dans le stage, les modèles n'ont pas été optimisés pour le jeu de donnée HydroSWOT.

5.1 Résultats pour la modélisation via ANN

5.1.1 Prédiction de Q

Modèle baseline On présente en premier lieu les résultats du modèle dit *baseline* (voir Table 1), c'est-à-dire avec peu de variables explicatives, mais judicieusement choisies pour que la modélisation ait un sens physique.

TABLE 1: Récapitulatifs des informations principales du modèle *baseline*

Epochs	Variables explicatives	Nombre de couches	Nombre de neurones / couche
500	$W, dH, stage$	32	64

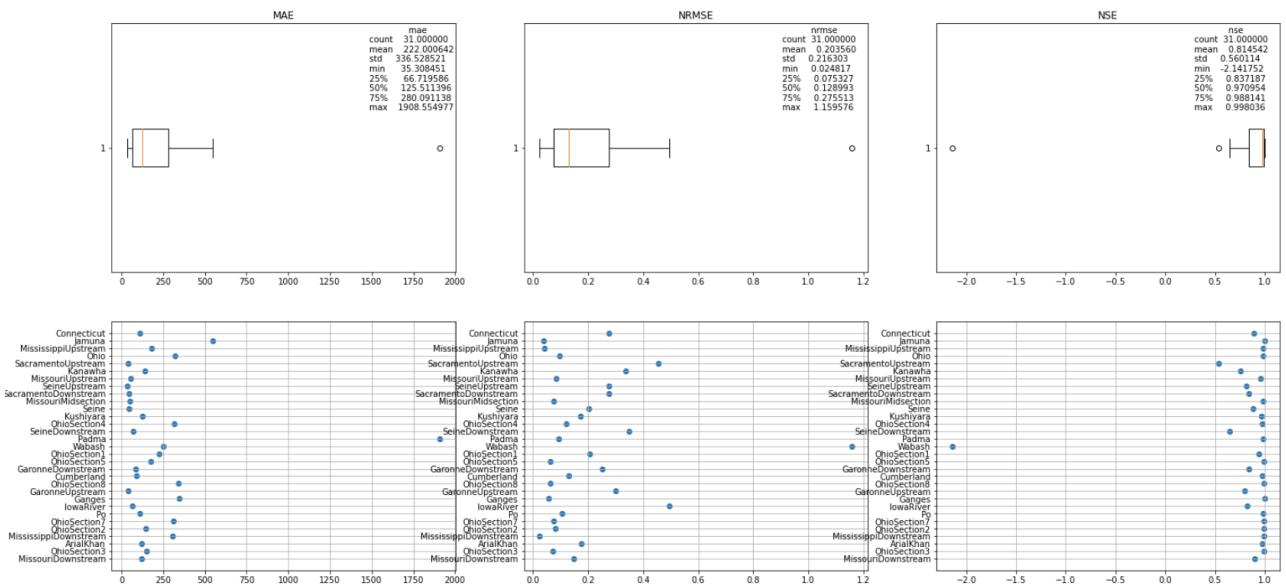


FIGURE 24: Résultats globaux de l'ANN *baseline* de prédiction du débit Q

On obtient déjà de très bons résultats avec ces variables : un NRMSE de 24% en moyenne et de 17% en valeur médiane, un NSE de 79% en moyenne et 96% en valeur médiane. On note un quantile à 75% de 37%, ce qui témoigne d'un étalement modéré des erreurs. On remarque que ce sont les petites rivières qui donnent le plus de mal à notre modèle, tandis que les grandes rivières donnent d'excellents résultats (voir Figure 24). Ces résultats viennent cependant contredire la physique, car on n'utilise pas la pente de la surface de l'eau, variable essentielle dans la modélisation du débit (voir Equation 3) et on obtient de très bons résultats, à la hauteur des modèles complexes que l'on a comparés en section 2.3.4.

En Figure 27 on compare les graphes de résultats pour une très grande rivière (Ganges) avec une petite rivière (Garonne).

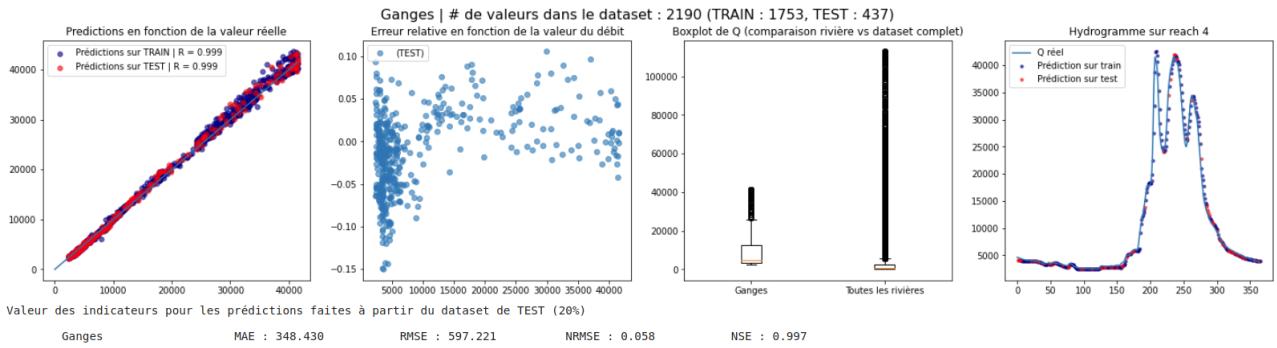


FIGURE 25: Résultats de l'ANN *baseline* pour la rivière du Ganges

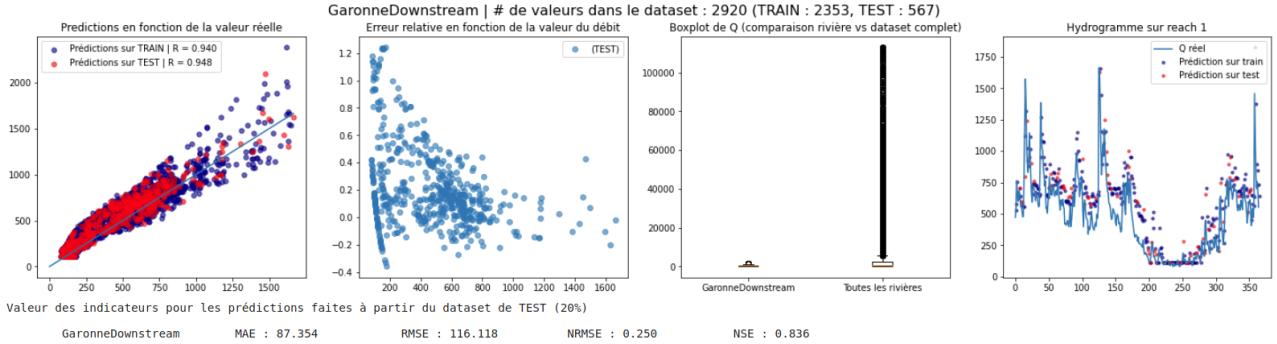


FIGURE 26: Résultats de l'ANN *baseline* pour la rivière de la Garonne

FIGURE 27: Comparaison entre les résultats sur le Ganges et la Garonne pour l'ANN *baseline*

On remarque que dans les deux cas, l'erreur relative est plus importante pour les observations de bas débit. Ce qu'on pourra noter aussi comme comportement des prédictions c'est la faible apparition d'un biais constant comme cela peut être le cas avec les méthodes conventionnelles de simulation du débit. Analysons maintenant les améliorations que peuvent permettre l'ajout de nouvelles variables.

Modèle *climato* On étudie ici une modélisation ANN avec comme seule différence avec le modèle *baseline* l'ajout des variables climatologique *PA* (pluviométrie moyenne annuelle) *TA* (température moyenne annuelle) ainsi que la pente *S* de la surface de l'eau (voir Table 2). On s'attend à de meilleurs résultats, car ces variables portent un réel sens physique dans la modélisation du débit, même s'il n'est pas ressorti des corrélations notables avec le débit lors de l'analyse statistique.

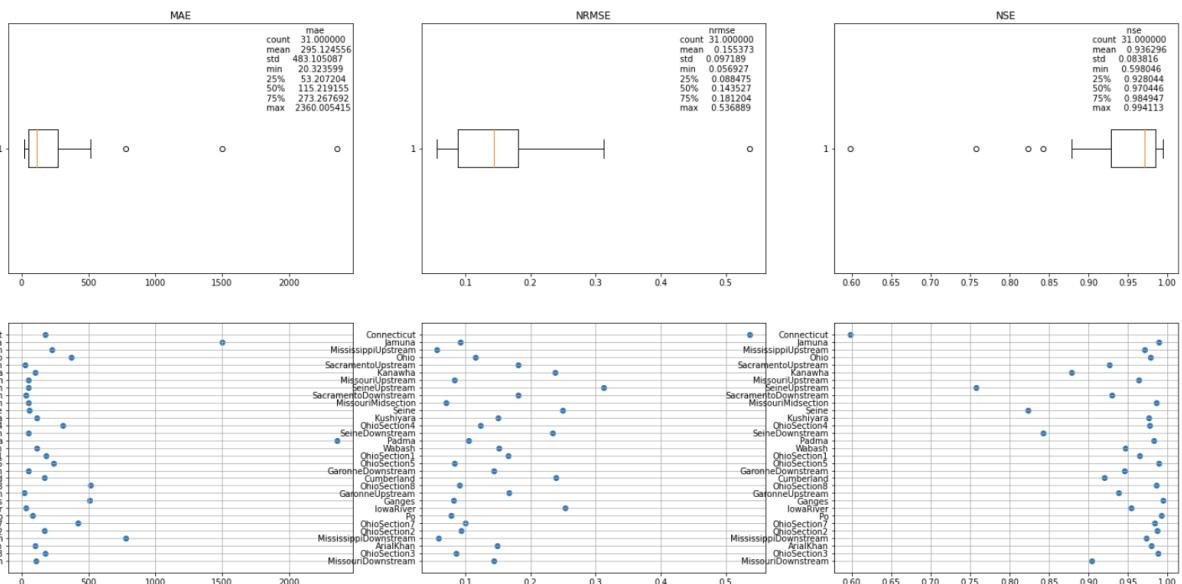


FIGURE 28: Résultats globaux de l'ANN *climato* de prédiction du débit *Q*

TABLE 2: Récapitulatifs des informations principales du modèle *climato*

Epochs	Variables explicatives	Nombre de couches	Nombre de neurones / couche
500	$W, dH, stage, S, PA, TA$	32	64

Comme on peut le voir en Figure 28, les résultats sont meilleurs. On note d'une part un meilleur NSE moyen avec 93% et un NRMSE sensiblement meilleur avec une valeur moyenne de 16% et une valeur médiane de 14%. Les quantiles sont aussi plus rapprochés, avec une valeur de 9% et 18% pour le quantile à 25% et le quantile à 75% respectivement témoignant d'une plus grande uniformité dans l'erreur que pour le modèle *baseline*.

Les observations faites individuellement sur les rivières peuvent se résumer à une réduction du biais, si tant est qu'il y en eu un. On se propose de comparer les résultats sur les rivières qui ont été assez mal modélisées par le modèle *baseline* pour rendre compte de l'amélioration des prédictions.

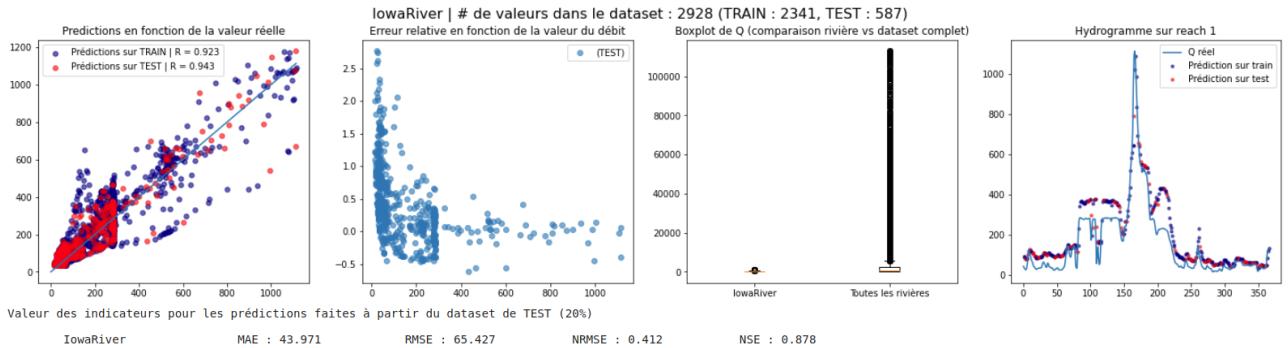


FIGURE 29: Résultats de l'ANN sur Q *baseline* pour la rivière Iowa

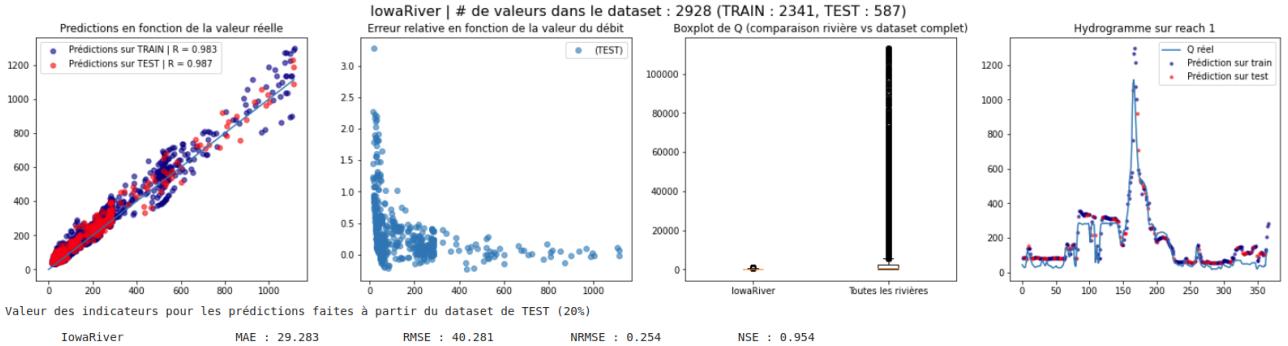


FIGURE 30: Résultats de l'ANN sur Q *climato* pour la rivière Iowa

FIGURE 31: Comparaison entre les résultats du modèle ANN sur Q *baseline* et *climato* pour la rivière Iowa

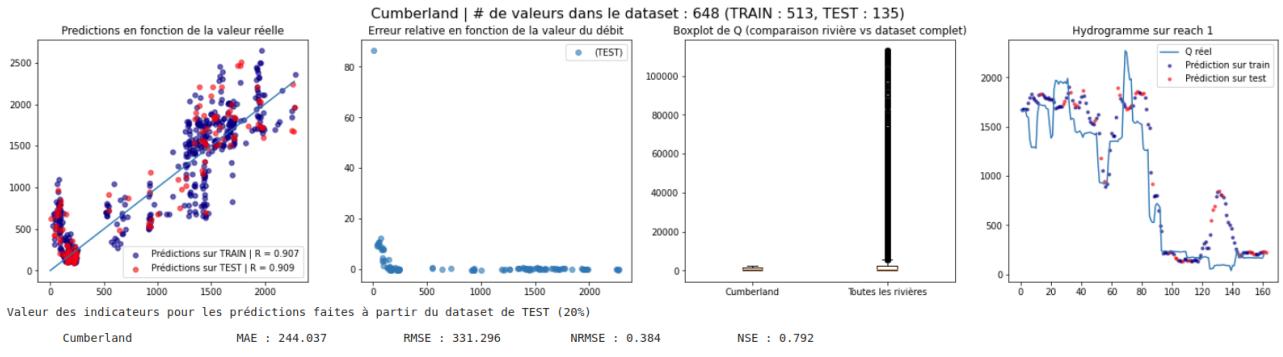


FIGURE 32: Résultats de l'ANN sur Q *baseline* pour la rivière Cumberland

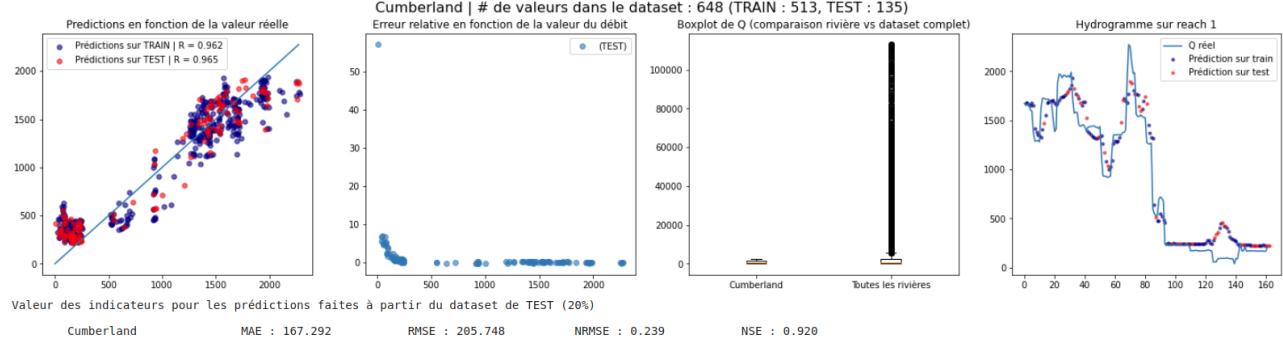


FIGURE 33: Résultats de l'ANN sur Q *climato* pour la rivière Cumberland

FIGURE 34: Comparaison entre les résultats du modèle ANN sur Q *baseline* et *climato* pour la rivière Cumberland

Pour l'Iowa qu'on compare en Figure 31, la diminution du biais est flagrante visuellement sur l'hydrogramme et se répercute sur la valeur de NRMSE notamment, qui passe de 41% à 25%. De la même manière pour la rivière Cumberland (voir Figure 34, on remarque une amélioration du NRMSE, mais surtout un meilleur *fit* sur l'hydrogramme réel des valeurs prédites, hydrogramme assez singulier d'ailleurs en comparaison aux autres rivières, ce qui peut expliquer, entre autres choses, le manque de précision dans le modèle *baseline* au départ.

Maintenant qu'on a vu qu'il était possible d'améliorer les résultats en ajoutant des variables *climato*, regardons ce qu'il se passe quand on utilise toutes les variables liées au sol dont on dispose.

Modèle complet Cette fois-ci on repart sur la base du modèle *climato* et on ajoute toutes les variables géologiques (voir Table 3)

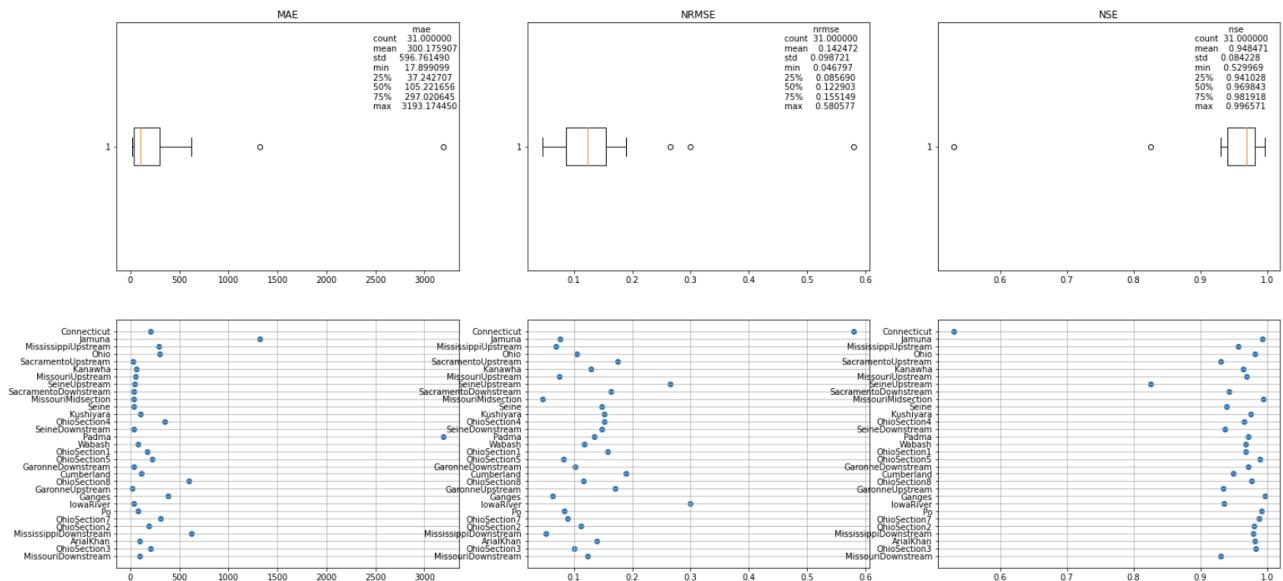


FIGURE 35: Résultats globaux de l'ANN *complet* de prédition du débit Q

TABLE 3: Récapitulatifs des informations principales du modèle *complet*

Epochs	Variables explicatives	Nombre de couches	Nombre de neurones / couche
500	$W, dH, stage, S, PA, TA, clay, sand, silt, LC1 - 12$	32	64

Comme nous le montre la Fig 35 cette nouvelle itération de modèle ne propose pas une grande amélioration par rapport au modèle *climato* relativement à la quantité d'informations qu'on ajoute au modèle. On note encore une fois un resserrement de l'étalement du NRMSE et d'excellentes valeurs de NSE.

5.1.2 Prédiction de \bar{A}

Pour la prédiction de \bar{A} , en plus de jouer sur les variables explicatives du modèle, on sépare aussi le jeu de données entre les rivières dont au moins une observation correspond à une valeur de débit réel de plus de $10\,000\,\text{m}^3/\text{s}$, qu'on appelle le jeu de données "grandes rivières", et les autres, qu'on appelle "petites rivières". On reprend ici les mêmes terminologies qu'utilisées dans la sous-section précédente pour désigner les modèles, c'est-à-dire modèle *baseline*, modèle *complet*.

Prédiction sur les "petites rivières" Le modèle *baseline* et le modèle *complet* fournissent tout deux d'excellents résultats avec un NRMSE moyen aux alentours des 2% (voir Figure 38). On pouvait s'y attendre compte tenu de l'ACP qui mettait en avant le facteur discriminant que représentait la géométrie de la rivière. On ne peut pas prétendre que le modèle soit robuste sur les "grandes rivières" mais on peut se servir de tels résultats pour envisager plus sérieusement la création de plusieurs modèles entraînés sur une classe de rivière.

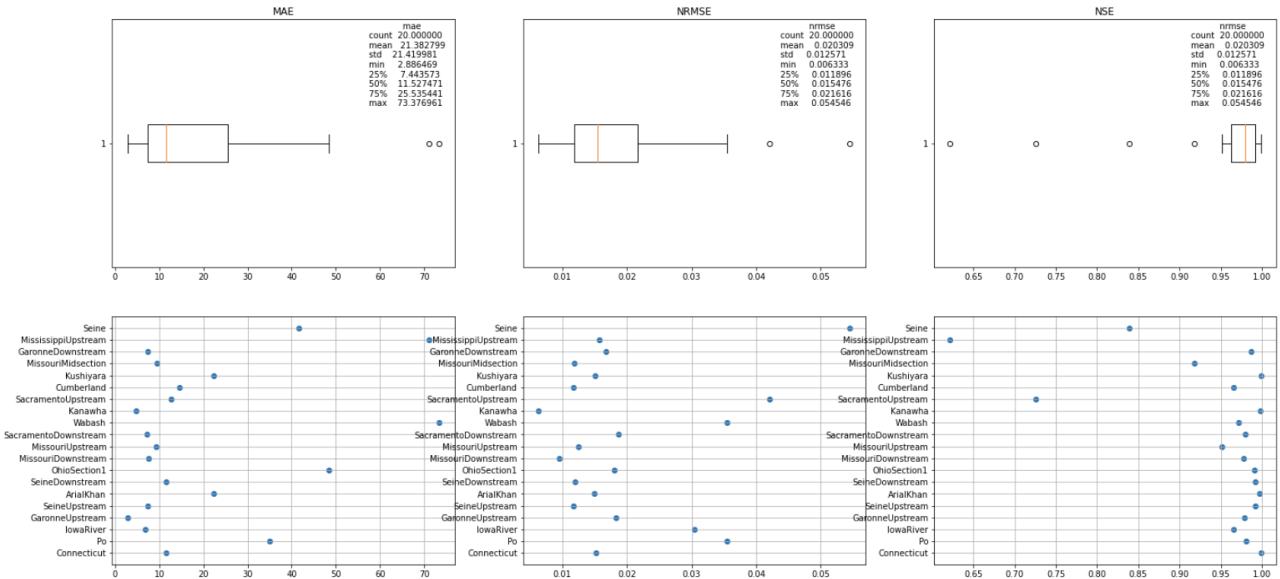


FIGURE 36: Résultats globaux de l'ANN *baseline*

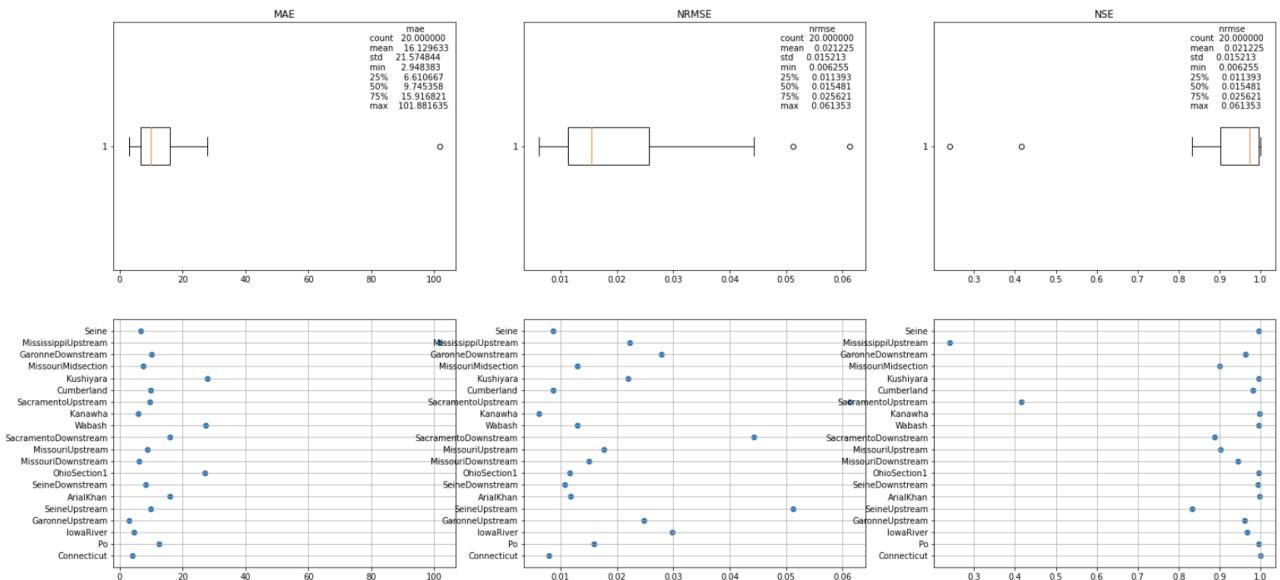


FIGURE 37: Résultats globaux de l'ANN *complet*

FIGURE 38: Comparaison entre les résultats du modèle ANN sur \bar{A} *baseline* et *complet* pour les "petites rivières"

En Figure 41 on affiche les résultats individuels pour quelques rivières :

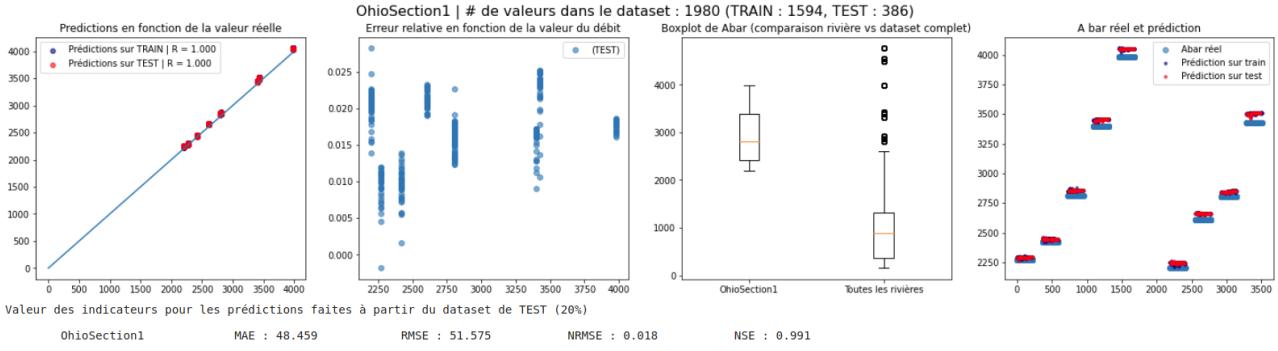


FIGURE 39: Résultats de l'ANN sur \bar{A} complet pour la rivière Ohio

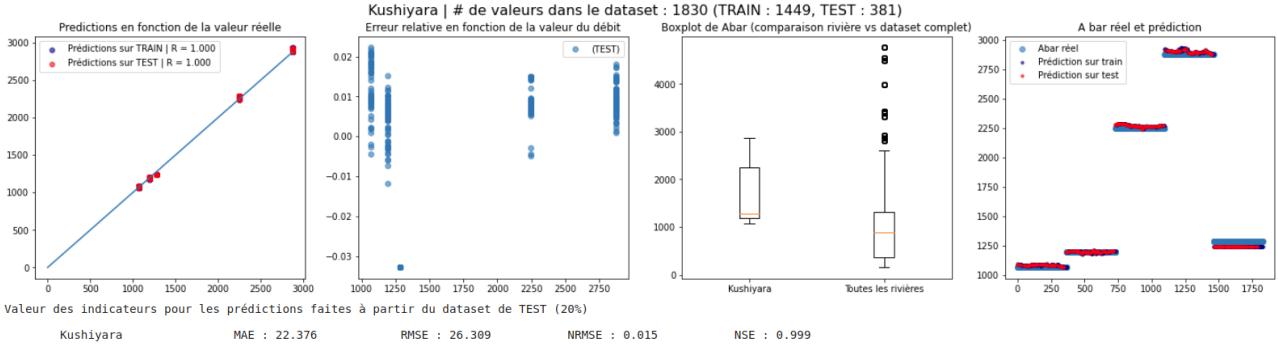


FIGURE 40: Résultats de l'ANN sur \bar{A} complet pour la rivière Kushiyara

FIGURE 41: Démonstration des résultats du modèle ANN sur \bar{A} complet pour deux "petites rivières"

Prédiction sur les "grandes rivières" Pour ce jeu de données, les résultats sont encore plus impressionnantes, comme on aurait pu s'y attendre (voir Figure 44). À ce niveau de précision, le modèle *complet* n'apporte pas de gain significatif relativement au nombre de nouvelles variables qu'il utilise par rapport au modèle *baseline*. Ces résultats confirme la pertinence d'un partitionnement entre les "grandes" et "petites" rivières selon la valeur de débit seuil des $10000m^3/s$.

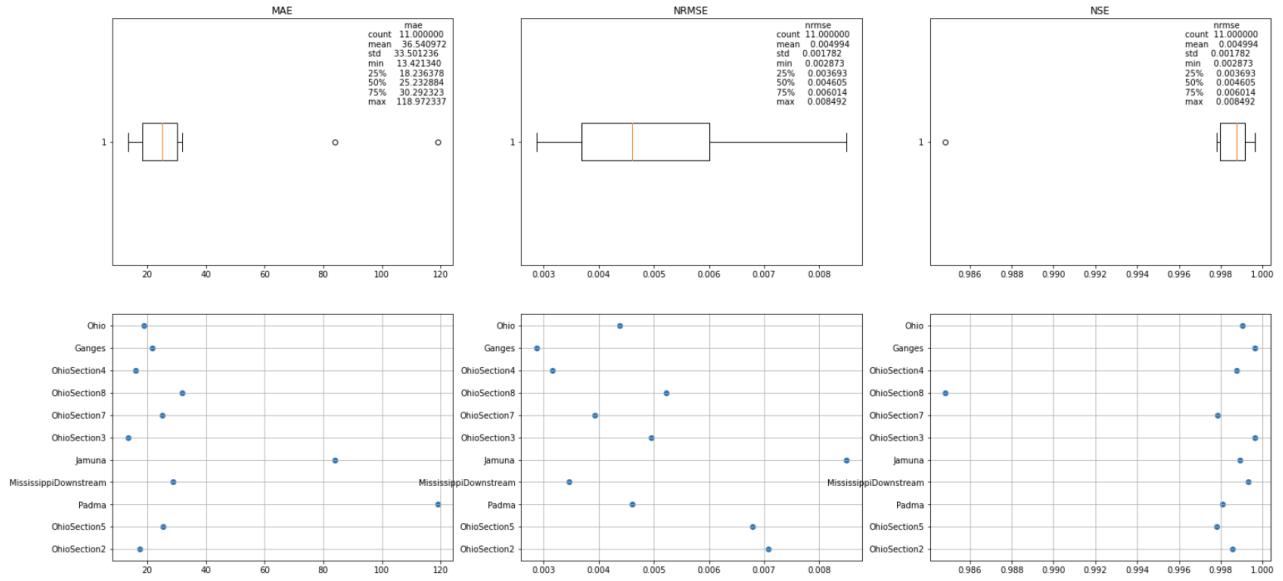


FIGURE 42: Résultats globaux de l'ANN *baseline*

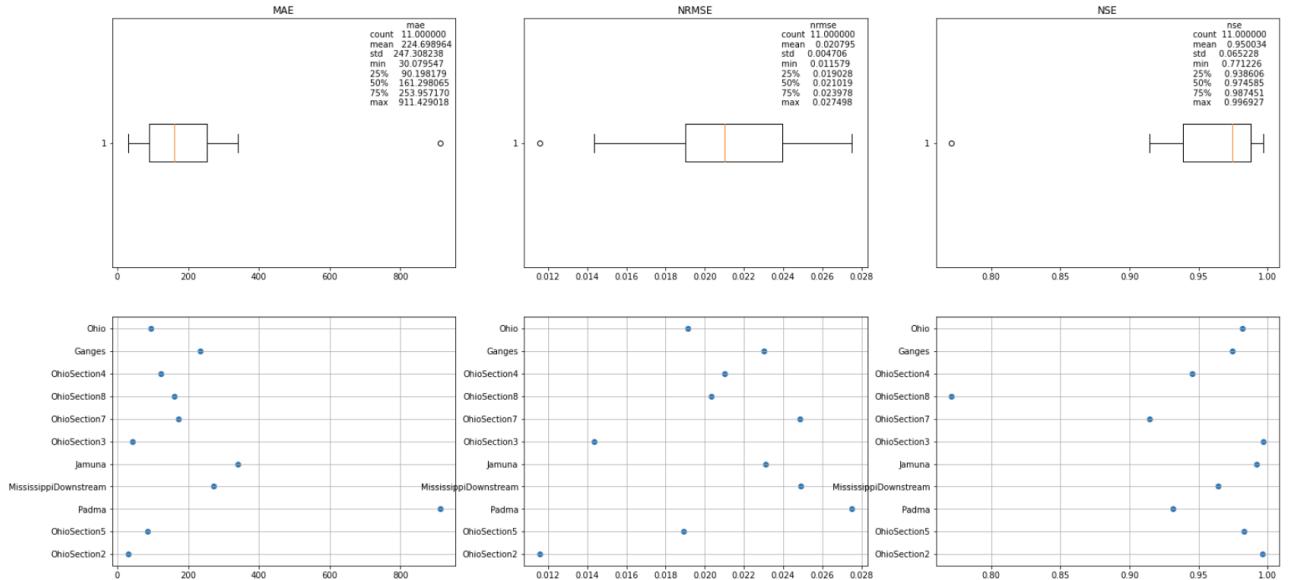


FIGURE 43: Résultats globaux de l'ANN *complet*

FIGURE 44: Comparaison entre les résultats du modèle ANN sur *A baseline* et *complet* pour les "grandes rivières"

Observons quelques résultats individuels (Ganges et Ohio) en Figure 47

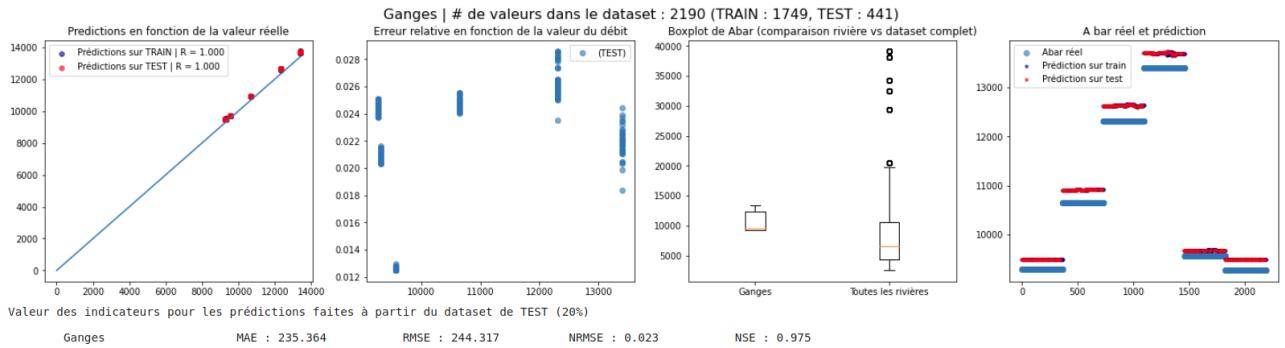


FIGURE 45: Résultats de l'ANN sur \bar{A} complet pour la rivière Ganges

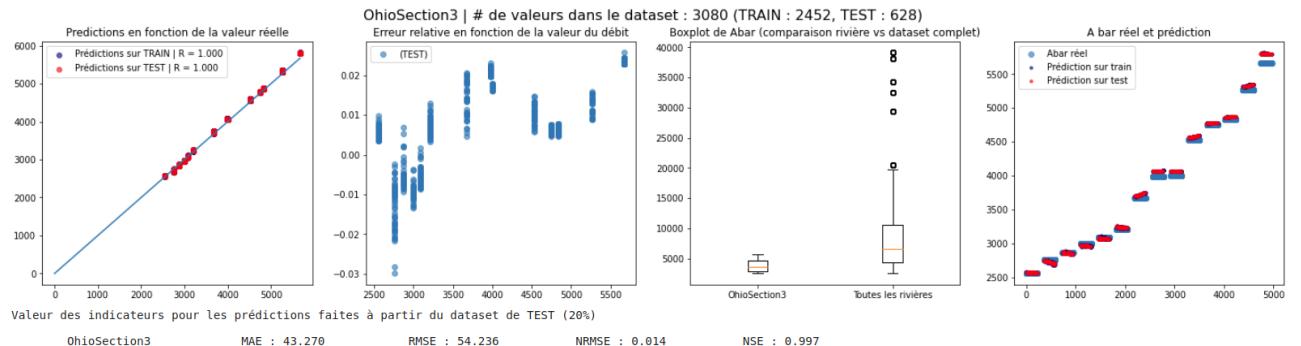


FIGURE 46: Résultats de l'ANN sur \bar{A} complet pour la rivière Ohio

FIGURE 47: Démonstration des résultats du modèle ANN sur \bar{A} complet pour deux "grandes rivières"

La conclusion que l'on peut tirer de ces résultats de modèles entraînés indépendamment sur 2 classes de rivières distinctes est que l'on peut sérieusement envisager de créer plusieurs modèles en fonction des classes de rivières principales. De plus, la division en "grandes" et "petites" rivières corrobore nos résultats d'ACP. On ne prétend donc pas ici produire des modèles robustes, mais des modèles très performants quand la rivière que l'on désire utiliser pour une prédition appartient bien à la classe de rivière sur laquelle le modèle a été entraîné.

5.2 Résultats pour la modélisation via LSTM

Les résultats présentés dans le cas du LSTM ont vocation à donner une intuition sur le potentiel de ces derniers dans la modélisation de Q , on ne présente qu'un cas très particulier ici. En Figure 48 sont présentés les résultats globaux de NRMSE pour un LSTM entraîné sur la Seine avec les mêmes variables que le modèle *baseline* décrit précédemment. On utilise une fenêtre glissante de 10 jours pour prédire la valeur du 11e jour.

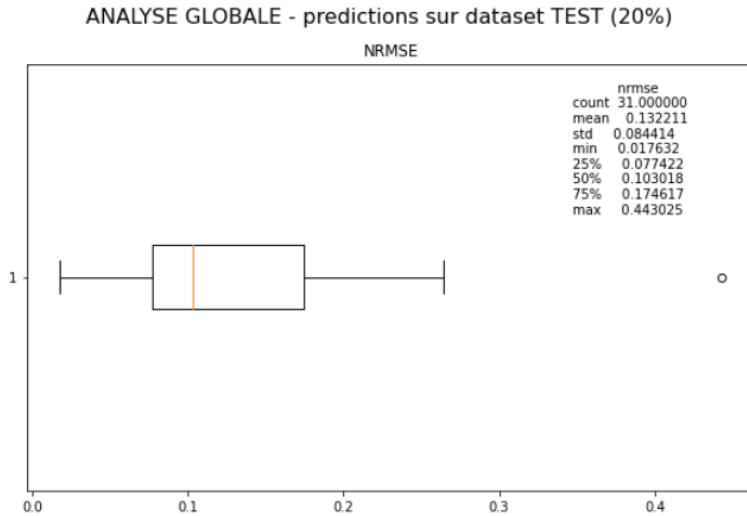


FIGURE 48: Résultats globaux du LSTM entraîné sur la Seine

On remarque que les résultats sont très bons, malgré que le problème soit plutôt simple, car on ne prédit que pour le jour suivant la fenêtre. Cependant il faut noter que l'entraînement ne s'est fait que sur une seule rivière et que le NRMSE moyen et médian se situe aux alentours des 10% ce qui témoigne d'une bonne faculté de généralisation du modèle.

On essaye de rendre compte de cette faculté de généralisation en entraînant cette fois-ci le LSTM sur les données d'une "grande rivière", ici le Ganges. En Figure 49 on présente les résultats sur le NRMSE global.

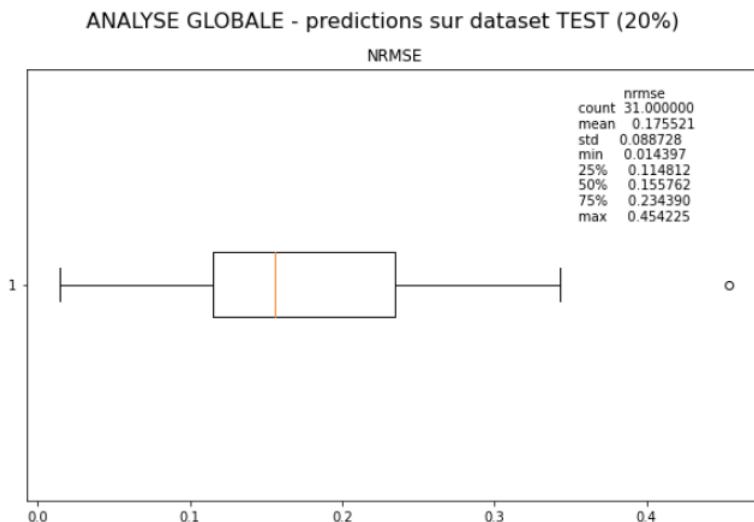


FIGURE 49: Résultats globaux du LSTM entraîné sur la Ganges

On constate une erreur relativement uniforme avec une valeur médiane et moyenne aux alentours des 16% et la majorité des NRMSE contenus sous la barre des 30%.

5.3 Analyse des résultats

Suite à cette analyse, il ressort plusieurs points :

- Les réseaux de neurones sont un outil puissant, mais dangereux scientifiquement. On peut obtenir de bons résultats avec une modélisation ne respectant pas la physique comme nous le montre le modèle *baseline*. Il faut donc se montrer méfiant et être rigoureux dans le processus de test des modèles si l'on veut s'assurer d'avoir des résultats fiables et utilisables dans des applications critiques.
- L'ajout de variables liées à géologie des rivières a su montrer un gain en performance significatif dans certains cas de figure. Il n'est cependant pas assez significatif pour affirmer avec fermeté que cet ajout est réellement pertinent.

- Comme dans beaucoup de méthodes d'apprentissage supervisé, les classes d'individus que l'on présente à l'algorithme lors de l'entraînement conduisent à un fort biais. C'est ce qu'on a pu mettre en évidence en analysant plus en détail les résultats fournis par l'ANN sur Q et avec ceux fournis par l'ANN sur \bar{A} , là où l'on a manuellement séparé le jeu de données en deux classes. On obtient certes de très bons résultats, mais il faut être sûr de la classe à laquelle appartient une rivière pour laquelle on veut obtenir une prédition fiable.
- Les LSTM peuvent apporter encore de nouvelles pistes pour une modélisation plus robuste. Ces derniers ont montré dans cette analyse non exhaustive une bonne capacité de généralisation même si des tests plus approfondis sont nécessaires avant d'affirmer fermement cela.

6 Conclusion

En conclusion, on peut exprimer sans appel que le potentiel des réseaux de neurones est immense dans la modélisation de phénomènes hydrologiques complexes, fortement non linéaires, et multi échelles comme le débit. Les méthodes sont certes facilement mises en place grâce aux librairies spécialisées comme Keras, mais il ne faut pas oublier que de nombreux paramètres, dont l'influence est, pour certains, encore peu comprise, peuvent être mieux ajustés au problème. De plus, le domaine des réseaux de neurones et consorts étant en effervescence dernièrement, il faut savoir faire la part des choses quand aux résultats que l'on pourrait trouver dans la littérature. Ce point est critique dans notre cas, car le manque de discernement peut parfois amener à donner plus de poids qu'il n'en faut à des articles peu fiables pour un esprit averti. Finalement, bien que l'expertise métier ne soit pas tout le temps nécessaire lors de projet mettant en oeuvre des méthodes de machine learning, il est important dans notre cas de garder un état d'esprit de physicien/mathématicien et de comprendre les problématiques liées à l'altimétrie spatiale et les questions de modélisations complexes qui en découlent pour mieux analyser nos résultats.

Apports personnel Comme mentionnés précédemment, les réseaux de neurones sont aujourd'hui un domaine vaste, et être encadré par des chercheurs lors de ma phase de revue de l'état de l'art a su aiguiser mon sens critique, mon esprit de synthèse et surtout me faire gagner en maturité scientifique. En tant que première réelle expérience dans le monde de la recherche et de la science des données, surtout en ces temps inédits de confinement, je suis très satisfait du déroulement du stage et des résultats obtenus et très enthousiaste de mettre en oeuvre mes nouvelles connaissances et nouveaux réflexes au service de projets aussi stimulants que celui-ci.

Annexe A

Notebook ANN

Artificial Neural Network

Estimation du débit à partir de la base de donnée PEPSI

Le but de ce notebook est de construire un approximateur de la variable débit grâce à un réseau de neurones profonds. On utilise ici la librairie **keras** et le jeux de données issues de simulation modèle **PEPSI**.

Note : Une partie du processus de pré-traitement est réalisée dans le notebook **preprocess.ipynb** dont les fonctions sont importées ici.

```
In [ ]: 1 import time
2 import pickle
3 import kerastuner
4 import numpy as np
5 import pandas as pd
6 from scipy import stats
7 import tensorflow as tf
8 from tensorflow import keras
9 import matplotlib.pyplot as plt
10 import tensorflow_probability as tfp
11 from sklearn.preprocessing import StandardScaler
12 from ipynb.fs.full.preprocess import pepsiv2_data
13 from tensorflow.keras.callbacks import EarlyStopping
14 from sklearn.model_selection import train_test_split
15 from kerastuner.tuners import RandomSearch, Hyperband
16 from tensorflow.keras import models, layers, initializers
```

Pré-traitement

On part d'un jeu de données brutes contenant les valeurs de différentes variables pour plusieurs mesures à des instants et localisation différentes. On veut construire un jeu de donnée **X** (variables explicatives) et **y** (variable(s) cible(s)).

1. Nettoyage du dataset (fait dans preprocess.ipynb)
2. Suppression des variables (colonnes) qui ne nous intéressent pas
3. Sélection des variables explicatives et création du dataset **X**
4. Sélection de la variable cible et création du dataset **y**
5. On scale les données pour qu'elles soient uniformément petite en valeur absolue.

```
In [ ]: 1 pepsi = pepsiv2_data()
2 pepsi.drop(pepsi.loc[:, "lon" : "QWBM"], inplace = True, axis = 1)
3 pepsi.drop(["river_name", "U", "A0", "Abar"], inplace = True, axis = 1)
4 # pepsi.drop(pepsi.loc[:, "LC1": "LC12"], inplace = True, axis = 1)
5 print("Variables disponibles :")
6 print(pepsi.columns)
```

Création du dataset X

Note : Les variables **day** et **reach** sont sélectionnées ci-dessous mais n'entrent pas en jeu dans le modèle. Elles servent juste à garder un moyen **d'ordonner les mesures dans l'espace et le temps** au moment de l'analyse des résultats dans le notebook **analysis.ipynb**

```
In [ ]: 1 var = ["day", "reach", "dH", "W", "stage", "S"]
2 pepsi_X = pepsi.copy()
3 pepsi_X = pepsi_X[var]
4 pepsi_X
5 # pepsi_train.drop(pepsi_train.loc[:, "Q5_GSCD": "Q"], inplace = True, axis = 1)
6 # var = list(pepsi_train.columns)
7 # print(var)
```

Création du dataset y

Idem ici, on garde **day** et **reach**

```
In [ ]: 1 pepsi_y = pepsi.copy()
2 pepsi_y.drop(pepsi_y.loc[:, "flowacc": "Q95_GSCD"], inplace = True, axis = 1)
3 pepsi_y
```

Fonction de scaling

Ici on implémente une fonction basée sur la routine StandarScaler de la librairie SkLearn. Son effet et de soustraire aux données leur moyenne et diviser par leur variance (centré réduit)

```
In [ ]: 1 def scaling(x_train, x_test):  
2     scaler = StandardScaler()  
3     scaler.fit(x_train)  
4     x_train_scaled = scaler.transform(x_train)  
5     x_test_scaled = scaler.transform(x_test)  
6     return x_train_scaled, x_test_scaled
```

Modélisation et entraînement

Dans cette partie on construit un réseau de neuronne artificiel (ANN, feedforward) avec la librairie keras.

Dans le bloc ci-dessous on customise une classe de *callback* proposée par Keras (fonction qui se lance à chaque fin d'**epoch** pendant l'entraînement) pour qu'elle applique une condition d'arrêt (voir fonction `on_epoch_end()` de la classe `EarlyStopByMAE`)

Lors de l'entraînement keras fournit des valeurs d'indicateurs calculées à partir des données d'entraînement et des données de validation. (Les données de validation sont un subset des données d'entraînement qui permettent de vérifier si le modèle est en situation d'overfitting)

Les indicateurs

- MAE
- Coeficient de correlation de Pearson
- Erreur relative

In []:

```
1 # Création et entrainement du modèle
2
3 # Custom metric / passe en linéaire
4 def log_mae(y_true, y_pred):
5     y_true_linear = tf.exp(y_true)
6     y_pred_linear = tf.exp(y_pred)
7     return keras.losses.MAE(y_true_linear, y_pred_linear)
8
9 def log_pearsonr(y_true, y_pred):
10    y_true_linear = tf.exp(y_true)
11    y_pred_linear = tf.exp(y_pred)
12    return tfp.stats.correlation(y_true_linear, y_pred_linear)
13
14 def pearsonr(y_true, y_pred):
15    return tfp.stats.correlation(y_true, y_pred)
16
17 def realative_error(y_true, y_pred):
18    return (tf.abs(y_true - y_pred))/y_true
19
20 ## fonction callback pour appliquer une règle d'arrêt de l'entraînement
21 class EarlyStopByMAE(keras.callbacks.Callback):
22     def __init__(self,
23                  monitor = "val_loss",
24                  verbose = 0,
25                  traning_set=pd.DataFrame(),
26                  logmode = False,
27                  patience = 0):
28         super(keras.callbacks.Callback, self).__init__()
29         self.verbose = verbose
30         self.monitor = monitor
31         self.traning_set = traning_set
32         self.logmode = logmode
33         self.patience = patience
34         self.wait = 0
35
36     def get_monitor_value(self, logs):
37         logs = logs or {}
38         monitor_value = logs.get(self.monitor)
39         if monitor_value is None:
40             logging.warning('Early stopping conditioned on metric `%s` '
41                             'which is not available. Available metrics are: %s',
42                             self.monitor, ','.join(list(logs.keys())))
43         return monitor_value
44
45     def on_epoch_end(self, epoch, logs={}):
46         current = self.get_monitor_value(logs)
47
48         # Threshold = 10% of mean streamflow over all training data set
49         threshold = np.exp(self.traning_set).mean() * 0.1 if logmode else self.traning_set.mean() * 0.1
50         if current < threshold:
51             self.wait +=1
52             if self.wait >= self.patience:
53                 if self.verbose >0:
54                     print("Epoch %05d: early stopping Threshold" % epoch)
55                 self.model.stop_training = True
56             else:
57                 self.wait = 0
58
59     def build_model(logmode, n_layers, width):
60         """
61             Return a compiled keras ANN with n_layers layers and width neurons by layer
62         """
63         model = models.Sequential()
64         # hidden Layers
65         for i in range(n_layers):
66             model.add(layers.Dense(width, activation = "relu",
67                                   kernel_initializer=initializers.RandomNormal(mean=0.0, stddev=0.02, seed=2020)))
68
69         # output Layers
70         model.add(layers.Dense(1, activation = "linear",
71                               kernel_initializer=initializers.RandomNormal(mean=0.0, stddev=0.02, seed=2020)))
72         # Optimisation parameters
73         metrics = ["mae", log_mae, log_pearsonr] if logmode else ["mae", pearsonr, realative_error]
74         model.compile(optimizer = 'adam', loss='mse', metrics = metrics)
75
76         return model
77 # es = EarlyStopping(monitor="Log_mae", baseline = 100, verbose = 1, patience = 5)
78
79     def train_model(x_train_scaled, y_train, model, logmode) :
80         """
81             Run the training procedure and return the trained model,
```

```

82 """
83     monitor = "log_mae" if logmode else "mae"
84     es = EarlyStopByMAE(monitor = monitor,
85                           verbose = 1,
86                           traning_set = y_train,
87                           logmode = logmode,
88                           patience = 100)
89     history = model.fit(x_train_scaled,
90                           y_train,
91                           epochs = 500,
92                           validation_split=0.2,
93                           verbose = 2,
94                           callbacks=[es])
95     return history, model

```

In []:

```

1 if __name__ == "__main__":
2
3     logmode = False
4     n_layers = 32
5     width = 64
6
7     x_train, x_test, y_train, y_test = train_test_split(pepsi_X, pepsi_y, test_size=0.2,random_state=42)
8     x_train_scaled, x_test_scaled = scaling(x_train, x_test)
9
10    if logmode:
11        y_train_norm, y_test_norm = np.log(y_train)[ "Q" ].to_numpy(), np.log(y_test)[ "Q" ].to_numpy()
12        log_string = '_log'
13    else:
14        log_string = ''
15        y_train_norm, y_test_norm = y_train[ "Q" ].to_numpy(), y_test[ "Q" ].to_numpy()
16
17    model = build_model(logmode, n_layers, width)
18    history, trained_model = train_model(x_train_scaled, y_train_norm, model, logmode)
19
20    # Enregistrement du modèle entraîné et de l'historique d'entraînement
21
22    trained_model.save(f'../results/models/pepsiv2/Q_{n_layers}x{width}_{tuple(var)}{log_string}')
23    pickle.dump(history.history, open(f'../results/performance/pepsiv2/Q_{n_layers}x{width}_{tuple(var)}{log_string}', 'wb'))
24    plt.plot(history.history['loss'])

```

Annexe B

Notebook LSTM

LSTM

Practical part

1. Preprocessing
 - A. Dataset import and variable selection
 - B. Series to supervised procedure
2. Modelling with keras

Very usefull blog reference : <https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>
[\(https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/\)](https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/)

```
In [ ]: 1 import time
2 import pickle
3 import kerastuner
4 import numpy as np
5 import pandas as pd
6 from scipy import stats
7 import tensorflow as tf
8 from tensorflow import keras
9 import matplotlib.pyplot as plt
10 from IPython.display import Latex
11 import tensorflow_probability as tfp
12 from sklearn.preprocessing import StandardScaler
13 from sklearn.preprocessing import StandardScaler
14 from tensorflow.keras.callbacks import EarlyStopping
15 from sklearn.model_selection import train_test_split
16 from kerastuner.tuners import RandomSearch, Hyperband
17 from tensorflow.keras import models, layers, initializers
18 from ipynb.fs.full.preprocess import pepsiv2_data, pepsiv3_data
```

Data import and variable selection

1. Remove useless columns
2. Select a river (to go further : train a LSTM with multiple rivers if possible)
3. Select a reach

```
In [ ]: 1 pepsi = pepsiv3_data()
2 print("Columns before cleaning : \n", list(pepsi.columns))
3 pepsi.drop(pepsi.loc[:, "lon" : "QWBM"], inplace = True, axis = 1)
4 pepsi.drop(pepsi.loc[:, "Q5_GSCD" : "Q95_GSCD"], inplace = True, axis = 1)
5 pepsi.drop(["river_name", "A0", "dA"], inplace = True, axis = 1)
6 # pepsi.drop(pepsi.loc[:, "LC1" : "LC12"], inplace = True, axis = 1)
7 print("\n Columns after cleaning : \n", list(pepsi.columns))
8 rivers = pepsi.index.unique()
9 selected_river = "Ganges"
10 data_river = pepsi.loc[selected_river]
11 data_river_reach = data_river[data_river["reach"] == 4] # First reach = #9 for the Seine
12 data_river_reach = data_river_reach.set_index("day")
13 data_river_reach
```

```
In [ ]: 1 data_river_reach["Q"].describe()
```

Preprocessing [FR]

1. Sélection des variables explicatives + target
2. Scaling des valeurs (StandardScaler Sklearn -> centré réduit)
3. Passage du dataset dans un algorithme de type "series_to_supervised"
 - On choisit le nombre n de jours dans le passé qui seront utilisés par le LSTM pour effectuer la prédiction
 - Génération d'une matrice 2D par l'algorithme
 - les colonnes sont les valeurs des variables explicatives, dupliquées n fois, avec un shift de 1 time step par duplication de la colonne
 - ex de colonnes : var1(t-3) | var1(t-2) | var1(t-1) | var1(t) pour n = 3 avec une seule variable explicative utilisée
 - Il y a autant de lignes que de timestep dans le jeux de donnée original (-n jours, qui contiennent des valeurs NaN à cause du shift)
4. Reshape de la matrice 2D en 3D
 - On garde la 1ere dimension intacte (le nombre d'échantillons)
 - La 2eme dimension devient le nombre de jours n
 - La 3eme dimension les variables explicatives

```
In [ ]: 1 group = ["dH", "W", "stage", "Q"] # Selected variables
2 # ajouter la pente
3 var = group[:-1] # pour le nom du fichier à l'enregistrement -> noms de variables explicatives
4 fig, ax = plt.subplots(len(group), 1, figsize=(10,5))
5 k = 0
6 for col in group:
7     ax[k].plot(data_river_reach[col])
8     ax[k].set_title(col,y=0.5,loc="right")
9     k+=1
10 plt.show()
```

Variable selection

```
In [ ]: 1 data_river_reach = data_river_reach[group].copy()
2 data_river_reach.reset_index(drop=True, inplace=True)
3 data_river_reach
```

```
In [ ]: 1 data_river_reach
```

Scaling

Note : Scaling values is usefull for improving training speed and can also improve precision in practice even if there is no clear evidence why (from my personal researches)

```
In [ ]: 1 scaler = StandardScaler()
2 data_river_reach = scaler.fit_transform(data_river_reach.values)
3 data_river_reach
```

Series to supervised

Copied function from <https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>
[\(https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/\)](https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/)

```
In [ ]: 1 from pandas import DataFrame
2 from pandas import concat
3
4 def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
5     """
6     Frame a time series as a supervised learning dataset.
7     Arguments:
8         data: Sequence of observations as a list or NumPy array.
9         n_in: Number of lag observations as input (X).
10        n_out: Number of observations as output (y).
11        dropnan: Boolean whether or not to drop rows with NaN values.
12    Returns:
13        Pandas DataFrame of series framed for supervised learning.
14    """
15    n_vars = 1 if type(data) is list else data.shape[1]
16    df = DataFrame(data)
17    cols, names = list(), list()
18    # input sequence (t-n, ... t-1)
19    for i in range(n_in, 0, -1):
20        cols.append(df.shift(i))
21        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
22    # forecast sequence (t, t+1, ... t+n)
23    for i in range(0, n_out):
24        cols.append(df.shift(-i))
25        if i == 0:
26            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
27        else:
28            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
29    # put it all together
30    agg = concat(cols, axis=1)
31    agg.columns = names
32    # drop rows with NaN values
33    if dropnan:
34        agg.dropna(inplace=True)
35    return agg
36
```

```
In [ ]: 1 n_days = 10 # We use 10 days windows to train the LSTM
2 n_features = len(group)
3 reframed = series_to_supervised(data_river_reach, n_days, 1)
4 reframed.drop(reframed.columns[[-2,-3,-4,-5]], axis=1, inplace=True)
5 reframed
```

Reshaping for keras (2D to 3D)

```
In [ ]: 1 n_obs = n_features * n_days
2 values = reframed.values
3 train = values[:int(values.shape[0]*0.8), :]
4 test = values[int(values.shape[0]*0.8):, :]
5 train_X, train_y = train[:, :n_obs], train[:, -1]
6 test_X, test_y = test[:, :n_obs], test[:, -1]
7 train_X = train_X.reshape((train_X.shape[0], n_days, n_features))
8 test_X = test_X.reshape((test_X.shape[0], n_days, n_features))
9 print("Shape : (samples, timesteps, features)\n")
10 print("Train X shape :",train_X.shape, "\nTrain y shape :",train_y.shape)
11 print("Test X shape :",test_X.shape, "\nTest y shape :",test_y.shape)
12
```

Modelling and training

- Similarly to ANNs with keras, we choose the number of epochs, the optimizer etc.
- Number of units corresponds to the number of parallel LSTMs working together (should improve precision to have many, need clarification)

```
In [ ]: 1 n_units = 50
2 epochs = 500
3 model = models.Sequential()
4 model.add(layers.LSTM(n_units, input_shape=(train_X.shape[1], train_X.shape[2])))
5 model.add(layers.Dense(1))
6 model.compile(loss='mae', optimizer='adam')
7 # fit network
8 history = model.fit(train_X, train_y, epochs=epochs, validation_data=(test_X, test_y), verbose=2, shuffle=False)
9 model.save(f'../results/models/lstm/Q_{n_days}days_{epochs}_{n_units}units_{tuple(var)}_{selected_river}.h5')
10 pickle.dump(history.history, open(f'../results/performance/lstm/Q_{n_days}days_{epochs}_{n_units}units_{tuple(var)}_{selected_river}.pkl', 'wb'))
```

Bibliographie

- [Andreadis et al., 2013] Andreadis, K. M., Schumann, G. J.-P., and Pavelsky, T. (2013). A simple global river bankfull width and depth database. *Water Resources Research*, 49(10) :7164–7168.
- [Beck et al., 2015] Beck, H. E., de Roo, A., and van Dijk, A. I. J. M. (2015). Global Maps of Streamflow Characteristics Based on Observations from Several Thousand Catchments*. *Journal of Hydrometeorology*, 16(4) :1478–1501.
- [Biancamaria et al., 2016] Biancamaria, S., Lettenmaier, D. P., and Pavelsky, T. M. (2016). The swot mission and its capabilities for land hydrology. *Surveys in Geophysics*, 37(2) :307–337.
- [Braca, 2020] Braca, G. (2020). Stage-discharge relationships in open channels : Practices and problems.
- [CSGroup, 2020] CSGroup (2020). Csgroup. <https://www.c-s.fr/>. (Accessed on 10/25/2020).
- [Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314.
- [Di Baldassarre and Montanari, 2009] Di Baldassarre, G. and Montanari, A. (2009). Uncertainty in river discharge observations : a quantitative analysis. *Hydrology and Earth System Sciences*, 13(6) :913–921.
- [Durand et al., 2016] Durand, M., Gleason, C. J., Garambois, P. A., Bjerklie, D., Smith, L. C., Roux, H., Rodriguez, E., Bates, P. D., Pavelsky, T. M., Monnier, J., Chen, X., Di Baldassarre, G., Fiset, J.-M., Flipo, N., Frasson, R. P. d. M., Fulton, J., Goutal, N., Hossain, F., Humphries, E., Minear, J. T., Mukolwe, M. M., Neal, J. C., Ricci, S., Sanders, B. F., Schumann, G., Schubert, J. E., and Vilmin, L. (2016). An intercomparison of remote sensing river discharge estimation algorithms from measurements of river height, width, and slope. *Water Resources Research*, 52(6) :4527–4549.
- [Edwards et al., 2015] Edwards, P. J., Williard, K. W., and Schoonover, J. E. (2015). Fundamentals of watershed hydrology. *Journal of Contemporary Water Research & Education*, 154(1) :3–20.
- [Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2) :179–211.
- [Gaona et al., 2016] Gaona, M., Overeem, A., Leijnse, H., and Uijlenhoet, R. (2016). First-year evaluation of gpm rainfall over the netherlands : Imerg day 1 final run (v03d). *Journal of Hydrometeorology*, 17.
- [Hochreiter, 1998] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6 :107–116.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8) :1735–1780.
- [horace, 2020] horace (2020). horace.io/pytorch-vs-tensorflow/. <http://horace.io/pytorch-vs-tensorflow/>. (Accessed on 10/26/2020).
- [Hornik, 1991] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2) :251–257.
- [IMT, 2020] IMT (2020). Institut de mathématiques de toulouse. <https://www.math.univ-toulouse.fr/?lang=fr>. (Accessed on 10/25/2020).
- [INSA, 2020] INSA (2020). Enquête 1er emploi, promotion 2019. <https://en.calameo.com/read/00105768337d1500294bb>. (Accessed on 10/25/2020).
- [Keras, 2020] Keras (2020). Keras : the python deep learning api. <https://keras.io/>. (Accessed on 10/26/2020).
- [Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam : A method for stochastic optimization.
- [Kouraev et al., 2004] Kouraev, A. V., Zakharova, E. A., Samain, O., Mognard, N. M., and Cazenave, A. (2004). Ob' river discharge from topex/poseidon satellite altimetry (1992–2002). *Remote Sensing of Environment*, 93(1) :238 – 245.
- [Larnier et al., 2020] Larnier, K., Monnier, J., Garambois, P.-A., and Verley, J. (2020). River discharge and bathymetry estimations from SWOT altimetry measurements. working paper or preprint.
- [LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4) :541–551.

- [Lin et al., 2019] Lin, P., Pan, M., Beck, H. E., Yang, Y., Yamazaki, D., Frasson, R., David, C. H., Durand, M., Pavelsky, T. M., Allen, G. H., Gleason, C. J., and Wood, E. F. (2019). Global reconstruction of naturalized river flows at 2.94 million reaches. *Water Resources Research*, 55(8) :6499–6516.
- [McCabe et al., 2017] McCabe, M. F., Rodell, M., Alsdorf, D. E., Miralles, D. G., Uijlenhoet, R., Wagner, W., Lucieer, A., Houborg, R., Verhoest, N. E. C., Franz, T. E., Shi, J., Gao, H., and Wood, E. F. (2017). The future of earth observation in hydrology. *Hydrology and Earth System Sciences*, 21(7) :3879–3914.
- [Mladenova et al., 2014] Mladenova, I., Jackson, T., Njoku, E., Bindlish, R., Chan, S., Cosh, M., Holmes, T., de Jeu, R., Jones, L., Kimball, J., Paloscia, S., and Santi, E. (2014). Remote monitoring of soil moisture using passive microwave-based techniques – theoretical basis and overview of selected algorithms for amsr-e. *Remote Sensing of Environment*, 144 :197 – 213.
- [NASA, 2020] NASA (2020). Flight systems | mission – nasa swot. <https://swot.jpl.nasa.gov/mission/flight-systems/>. (Accessed on 09/11/2020).
- [Nash and Sutcliffe, 1970] Nash, J. and Sutcliffe, J. (1970). River flow forecasting through conceptual models part i – a discussion of principles. *Journal of Hydrology*, 10(3) :282 – 290.
- [Nordberg, 1965] Nordberg, W. (1965). Geophysical observation from nimbus i. *Science*, 150(3696) :559–572.
- [Olah, 2020] Olah, C. (2020). Understanding lstm networks – colah’s blog. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. (Accessed on 10/26/2020).
- [Sherstinsky, 2020] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D : Nonlinear Phenomena*, 404 :132306.
- [Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time : what it does and how to do it. *Proceedings of the IEEE*, 78(10) :1550–1560.
- [Wisser et al., 2010] Wisser, D., Fekete, B. M., Vörösmarty, C. J., and Schumann, A. H. (2010). Reconstructing 20th century global hydrography : a contribution to the global terrestrial network- hydrology (gtn-h). *Hydrology and Earth System Sciences*, 14(1) :1–24.