séries temporelles de données à haute fréquence

(deep learning)

Les séries temporelles sont un type de données crucial dans de multiples domaines tels que la finance, la météorologie, la santé, l'ingénierie, et bien d'autres encore. Elles se caractérisent par des enregistrements effectués à des intervalles de temps réguliers, permettant ainsi de saisir l'évolution d'un phénomène au fil du temps. Une série temporelle représente une séquence de données organisée chronologiquement, où chaque observation est liée à un moment spécifique dans le temps. Contrairement aux données statiques comme les images ou les enregistrements tabulaires, les séries temporelles reflètent les variations et les tendances temporelles, les rendant ainsi dynamiques et évolutives. Formellement, une série temporelle peut être définie comme une suite de points de données {(t1,x1),(t2,x2),...,(tn,xn)}, où ti représente le temps auquel une observation xi a été enregistrée. Les séries temporelles peuvent être échantillonnées à des intervalles réguliers (par exemple, toutes les heures, tous les jours) ou irréguliers.

L'analyse des séries temporelles consiste souvent à repérer des modèles temporels, notamment les tendances (mouvements généraux à long terme), les saisons (variations périodiques régulières), les cycles (fluctuations récurrentes) et les anomalies (observations inhabituelles ou imprévues). Cette exploration est essentielle pour comprendre les évolutions, identifier les schémas, anticiper les futurs événements et guider les décisions dans divers domaines.

Les séries temporelles sont omniprésentes dans une variété de domaines et de problématiques, allant de la prévision à la réduction de dimensionnalité. Elles sont utilisées pour prédire les tendances futures dans des domaines tels que la finance, la météorologie et la santé, où la capacité à anticiper les variations temporelles est cruciale pour la prise de décision. La détection d'anomalies dans les séries temporelles est également essentielle, permettant d'identifier des événements inhabituels ou des comportements anormaux dans des données, ce qui peut avoir des implications dans les domaines de la cybersécurité, de la maintenance prédictive et de la détection de fraudes. La segmentation des séries temporelles vise à diviser une série en segments ou intervalles significatifs, ce qui peut être utile dans des applications telles que la reconnaissance de motifs ou la décomposition de signaux complexes. La classification des séries temporelles implique d'attribuer des étiquettes ou des catégories à des séquences temporelles en fonction de leurs caractéristiques, ce qui est utilisé dans des domaines tels que la reconnaissance d'activités humaines à partir de données de capteurs ou la classification de signaux physiologiques pour le

diagnostic médical. La génération de séries temporelles consiste à créer de nouvelles séquences temporelles à partir de modèles statistiques ou d'apprentissage automatique, ce qui peut être utile dans la simulation, la création de données synthétiques ou la génération de scénarios futurs. Enfin, l'interprétation des séries temporelles vise à comprendre et à expliquer les modèles et les relations présents dans les données, ce qui peut aider à prendre des décisions éclairées et à éclairer la compréhension des phénomènes temporels complexes.

Au fil des années, une grande variété de techniques ont été développées pour traiter les séries temporelles, allant des méthodes classiques aux approches plus récentes basées sur l'apprentissage automatique et le deep learning. Les méthodes traditionnelles comprennent les modèles ARIMA (AutoRegressive Integrated Moving Average), les lissages exponentiels et les méthodes de régression linéaire. Cependant, avec les avancées rapides dans le domaine de l'intelligence artificielle et du machine learning, les approches basées sur le deep learning ont gagné en popularité en raison de leur capacité à capturer des structures complexes et à modéliser des données de grande dimensionnalité [31].

ETAT DE L'ART :

Le tableau suivant offre une synthèse des diverses technologies de deep learning appliquées aux séries temporelles. Les données recueillies proviennent principalement des articles [60] et [61], avec l'ajout de technologies complémentaires identifiées lors de mes propres recherches, mises en évidence en bleu.

Model Type	Name	Date	Reference
ANN	Artificial Neural Network	1988	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/ @inproceedings{smith1988using, title={Using the ADAP learning algorithm to forecast the onset of diabetes mellitus}, author={Smith, Jack W and Everhart, James E and Dickson, WC and Knowler,

			William C and Johannes, Robert Scott}, booktitle={Proceedings of the annual symposium on computer application in medical care}, pages={261}, year={1988}, organization={American Medical Informatics Association} }
RNN	Recurrent Neural Network	1989	https://direct.mit.edu/neco/article-abstract/1/2/270/5490/A-Learning-Algorithm-for-Continually-Running-Fully @article{williams1989learning, title={A learning algorithm for continually running fully recurrent neural networks}, author={Williams, Ronald J and Zipser, David}, journal={Neural computation}, volume={1}, number={2}, pages={270--280},

			year={1989}, publisher={MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info~…} }
CNN	Convolutional Neural Network	1995	https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e26cc4a1c717653f323715d751c8dea7461aa105 @article{lecun1995convolutional, title={Convolutional networks for images, speech, and time series}, author={LeCun, Yann and Bengio, Yoshua and others}, journal={The handbook of brain theory and neural networks}, volume={3361}, number={10}, pages={1995}, year={1995}, publisher={Citeseer} }
LSTM	Long, Short-Term Memory	1997	https://link.springer.com/chapter/10.1007/978-3-642-24797-

			2_4 @article{graves2012long, title={Long short-term memory}, author={Graves, Alex and Graves, Alex}, journal={Supervised sequence labelling with recurrent neural networks}, pages={37--45}, year={2012}, publisher={Springer} }
Bi-LSTM	Bidirectional Long, Short-Term Memory	1997	https://ieeexplore.ieee.org/abstract/document/650093 @article{schuster1997bidirectional, title={Bidirectional recurrent neural networks}, author={Schuster, Mike and Paliwal, Kuldip K}, journal={IEEE transactions on Signal Processing}, volume={45}, number={11}, pages={2673--2681}, year={1997}, publisher={Ieee}

			}
ESN	Les Echo State Networks	2001	https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf @article{jaeger2001echo, title={The “echo state” approach to analysing and training recurrent neural networks-with an erratum note}, author={Jaeger, Herbert}, journal={Bonn, Germany: German National Research Center for Information Technology GMD Technical Report}, volume={148}, number={34}, pages={13}, year={2001}, publisher={Bonn} }
HTM	Herarchical Term Memory	2004	https://www.sciencedirect.com/science/article/pii/S0925231217309864 @article{ahmad2017unsupervised, title={Unsupervised real-time anomaly

			detection for streaming data},
			  author={Ahmad, Subutai and Lavin, Alexander and Purdy, Scott and Agha, Zuha},
			
			journal={Neurocomputing},
			  volume={262},
			  pages={134--147},
			  year={2017},
			
			publisher={Elsevier}
			}

DBM	Deep Belief Machine	2006	https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf
			
			@article{hinton2006fast,
			  title={A fast learning algorithm for deep belief nets},
			  author={Hinton, Geoffrey E and Osindero, Simon and Teh, Yee-Whye},
			  journal={Neural computation},
			  volume={18},
			  number={7},
			  pages={1527--1554},
			  year={2006},
			  publisher={MIT Press One Rogers

			Street, Cambridge, MA 02142-1209, USA journals-info~…} }
ELM	Extreme Learning Machine	2007	https://ieeexplore.ieee.org/abstract/document/1380068 @inproceedings{huang2004extreme, title={Extreme learning machine: a new learning scheme of feedforward neural networks}, author={Huang, Guang-Bin and Zhu, Qin-Yu and Siew, Chee-Kheong}, booktitle={2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)}, volume={2}, pages={985--990}, year={2004}, organization={Ieee} }
GRU	Gated Recurrent Unit	2014	https://arxiv.org/abs/1406.1078 @article{cho2014learning, title={Learning

			phrase representations using RNN encoder-decoder for statistical machine translation},
			  author={Cho, Kyunghyun and Van Merri{\"e}nboer, Bart and Gulcehre, Caglar and Bahdanau, Dzmitry and Bougares, Fethi and Schwenk, Holger and Bengio, Yoshua},
			  journal={arXiv preprint arXiv:1406.1078},
			  year={2014}
			}

NTM	Neural Turing Machines	2014	https://arxiv.org/abs/1410.5401
			
			@article{graves2014neural,
			  title={Neural turing machines},
			  author={Graves, Alex and Wayne, Greg and Danihelka, Ivo},
			  journal={arXiv preprint arXiv:1410.5401},
			  year={2014}
			}

Encoder-Decoder	Encoder-Decoder with Attention	2014	https://arxiv.org/abs/1412.1602

			@article{chorowski 2014end, title={End-to-end continuous speech recognition using attention-based recurrent nn: First results}, author={Chorowski, Jan and Bahdanau, Dzmitry and Cho, Kyunghyun and Bengio, Yoshua}, journal={arXiv preprint arXiv:1412.1602}, year={2014} }
CNN + LSTM	Convolutional LSTM	2015	https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html@article{shi2015convolutional, title={Convolutional LSTM network: A machine learning approach for precipitation nowcasting}, author={Shi, Xingjian and Chen, Zhourong and Wang, Hao and Yeung, Dit-Yan and Wong, Wai-Kin and Woo, Wang-chun},

			journal={Advances in neural information processing systems}, volume={28}, year={2015} }
CNN + GRU	Convolutional Gated Recurrent Unit	-	-
CONVLSTM	-	-	-
CONVGRU	-	-	-
Pointeur Network	Pointeur Network	2015	https://arxiv.org/abs/1506.03134
MGU	Minimal Gated Unit	2016	https://arxiv.org/pdf/1603.09420.pdf
TCN	Temporal Convolutional Network	2016	https://link.springer.com/chapter/10.1007/978-3-319-49409-8_7 @inproceedings{lea2016temporal, title={Temporal convolutional networks: A unified approach to action segmentation}, author={Lea, Colin and Vidal, Rene and Reiter, Austin and Hager, Gregory D}, booktitle={Computer Vision--ECCV 2016 Workshops: Amsterdam, The Netherlands,

			October 8-10 and 15-16, 2016, Proceedings, Part III 14}, pages={47--54}, year={2016}, organization={Springer} }
GNN/GCN	Graph Neural Neutwok / Graph Convolutional Network	2016	https://arxiv.org/abs/1609.02907 @article{kipf2016semi, title={Semi-supervised classification with graph convolutional networks}, author={Kipf, Thomas N and Welling, Max}, journal={arXiv preprint arXiv:1609.02907}, year={2016} }
DNC	Differentiable Neural Computers	2016	https://www.nature.com/articles/nature20101 @article{graves2016hybrid, title={Hybrid computing using a neural network with dynamic external memory}, author={Graves,

			Alex and Wayne, Greg and Reynolds, Malcolm and Harley, Tim and Danihelka, Ivo and Grabska-Barwi{\'n}ska, Agnieszka and Colmenarejo, Sergio G{\'o}mez and Grefenstette, Edward and Ramalho, Tiago and Agapiou, John and others}, journal={Nature}, volume={538}, number={7626}, pages={471--476}, year={2016}, publisher={Nature Publishing Group UK London} }
CapsNets	Capsule Network	2017	https://proceedings.neurips.cc/paper_files/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html @article{sabour2017dynamic, title={Dynamic routing between capsules}, author={Sabour, Sara and Frosst, Nicholas and Hinton, Geoffrey E},

			journal={Advances in neural information processing systems},
			  volume={30},
			  year={2017}
			}

TRF	Transformer	2017	https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
			
			@article{vaswani2017attention,
			  title={Attention is all you need},
			  author={Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, {\L}ukasz and Polosukhin, Illia},
			
			journal={Advances in neural information processing systems},
			  volume={30},
			  year={2017}
			}

NODE / RNODE	Neural Ordinary Differential	2018	https://arxiv.org/abs/1806.07366

	Equations / Recurrent Neural Ordinary Differential Equations		@article{chen2018 neural, title={Neural ordinary differential equations}, author={Chen, Ricky TQ and Rubanova, Yulia and Bettencourt, Jesse and Duvenaud, David K}, journal={Advances in neural information processing systems}, volume={31}, year={2018} }

Les Reseaux reccurent (RNN / LSTM / GRU / BI-LSTM):

RNN

Dans le domaine de la modélisation de séquences, les réseaux neuronaux récurrents (RNN) ont démontré une grande efficacité dans diverses tâches telles que la traduction de langues [1, 2, 3], la reconnaissance de la parole [4], la génération de légendes pour les images, qui consiste à résumer le sens sémantique d'une image en une phrase [5, 6, 7], la détection d'actions dans les vidéos[8, 9], et même la prédiction à court terme des précipitations [10].

Ces réseaux, développés par Elman (1990) [11] et Jordan (1997) [12], présentent des boucles de rétroaction entre les unités, ce qui leur permet de mémoriser des séquences temporelles dynamiques.

Dans un RNN, l'élément principal est l'état caché (hidden state) qui est la représentation interne du réseau qui capture l'information contextuelle à partir des données d'entrée précédentes. L'état caché à l'instant t est généralement calculé en fonction de l'état caché précédent ht−1 et de l'entrée actuelle xt, ainsi que des poids et des biais du réseau. Sa formule mathématique est la suivante :

$h_t = f(W_h \cdot h_{t-1} + U_x \cdot x_t + b)$

où f est une fonction d'activation non linéaire, telle que la fonction tangente hyperbolique (tanh), Wh est la matrice de poids associée à l'état caché précédent, Ux est la matrice de poids associée à l'entrée actuelle, et b est le biais.Cependant, ils peuvent rencontrer des problèmes de disparition du gradient lors de l'apprentissage de séquences très longues.

LSTM

Pour pallier ce problème, le modèle Long Short Term Memory (LSTM) a été proposé par Hochreiter et Schmidhuber (1997) [13], puis amélioré par Gers, Schmidhuber et Cummins (2000) [14] avec l'introduction d'une porte d'oubli. Les LSTM conservent et utilisent certaines informations liées aux données précédemment vues dans le processus de décision, ce qui les rend incontournables dans l'analyse des séries temporelles [18, 19]

Dans un LSTM, chaque porte représente un mécanisme de contrôle essentiel permettant de réguler le flux d'informations à travers le réseau à chaque étape temporelle.

La porte d'oubli (ft) est contrôlée par une fonction d'activation sigmoïde et a pour formule mathématique :

$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

Cette porte décide quelles informations antérieures doivent être oubliées ou conservées dans la mémoire à long terme. Elle est contrôlée par une fonction d'activation sigmoïde (σ), qui comprime les valeurs entre 0 et 1, indiquant ainsi la proportion d'informations à oublier (0) ou à conserver (1). Le terme $[h_{t-1}, x_t]$ représente la concaténation du vecteur d'état caché de l'instant précédent (h_{t-1}) et du vecteur d'entrée actuel (x_t). Wf est la matrice de poids associée à cette porte, tandis que bf est le biais correspondant.

La porte d'entrée (it et gt) est également contrôlée par une fonction sigmoïde et une fonction tanh et a pour formule :

$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g)$

Cette porte régule le flux d'informations nouvelles à ajouter à la mémoire à long terme. Elle est également contrôlée par une fonction sigmoïde (σ) pour la partie it, qui décide de l'importance de chaque information nouvelle, et une fonction tanh pour la partie gt, qui calcule la nouvelle information candidate. Encore une fois, $[h_{t-1}, x_t]$ représente la concaténation du vecteur d'état caché de l'instant précédent et du vecteur d'entrée actuel. Wi et Wg sont les matrices de poids associées à cette porte, et bi et bg sont les biais correspondants.

La porte de sortie (ot) a pour formule :

$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Cette porte détermine la sortie du réseau à chaque étape temporelle en fonction de la mémoire à court terme actuelle et de l'entrée courante. Elle est également contrôlée par une fonction d'activation sigmoïde (σ) et utilise la même structure [ht−1, xt] pour intégrer l'état caché précédent et l'entrée actuelle. La matrice de poids Wo et le biais bo sont utilisés pour ajuster la sortie de cette porte.

GRU

D'autres RNN se sont également démarqués dans ce domaine [17, 75]. Le Gated Recurrent Unit (GRU), proposé en 2014 par Kyunghyun Cho et al., [15] offre une architecture moins complexe mais tout aussi performante que celle des LSTM. Alors qu'un LSTM comprend trois portes, le GRU en possède deux : une porte porte de reset et une porte de mise à jour.

La porte de réinitialisation (reset gate) : Contrôlée par une fonction d'activation sigmoïde, cette porte décide dans quelle mesure les informations antérieures doivent être oubliées ou conservées dans le calcul de l'état caché suivant. Sa formule mathématique est la suivante :

$$r_t=\sigma(W_r\cdot[h_{t-1},x_t]+b_r)$$

Cette porte détermine la proportion d'informations passées à oublier (0) ou à conserver (1). Le terme [ht−1, xt] représente la concaténation du vecteur d'état caché de l'instant précédent (ht−1) et du vecteur d'entrée actuel (xt). Wr est la matrice de poids associée à cette porte, tandis que br est le biais correspondant.

La porte de mise à jour (update gate) : Contrôlée également par une fonction d'activation sigmoïde, cette porte régule l'importance des informations nouvelles à ajouter à la mémoire à long terme. Sa formule est la suivante :

$$z_t=\sigma(W_z\cdot[h_{t-1},x_t]+b_z)$$

Elle utilise la même structure [ht−1, xt] que la porte de réinitialisation. Wz est la matrice de poids associée à cette porte, et bz est le biais correspondant.

La figure 1.1 illustre en détail ces trois architectures.

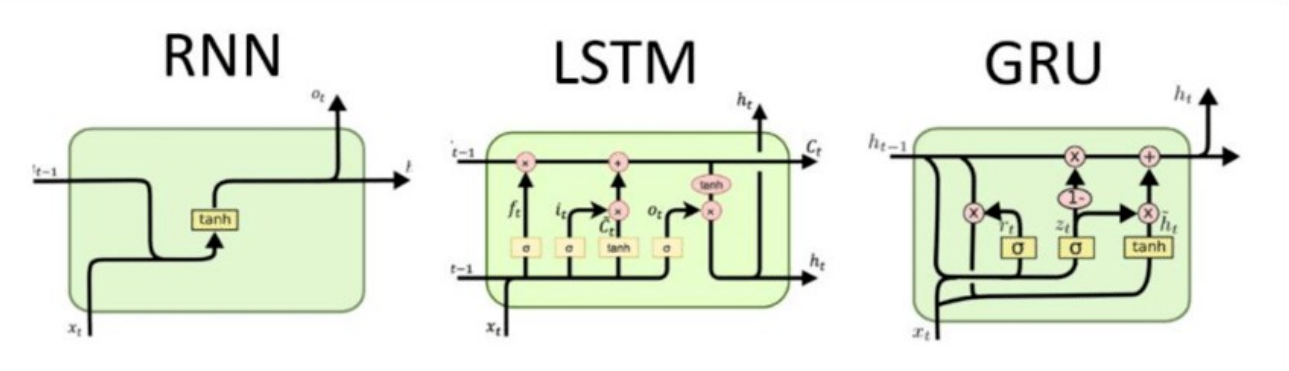


Fig. 1.1 : Architectures récurrentes (Toharudin et al., 2021)

MGU

Le Minimal Gated Unit (MGU) [16], proposé en 2016, se distingue par sa simplicité architecturale. Avec seulement une porte, le MGU a beaucoup moins de paramètres à apprendre que le GRU ou le LSTM, ainsi que moins de composants ou de variations à régler. Un schéma comparant les architecture de ces modèle est **disponible figure x.** Malgré cela il montre une précision comparable au GRU sur des jeux de donnée tel que MNIST ou IMDB [16]. Le fonctionnement du MGU implique de coupler la porte d'entrée à la porte d'oubli, en spécifiant que rt=ft,∀t. Cela signifie que MGU utilise uniquement la porte d'oubli, renommée ici comme la porte f, pour contrôler le flux d'informations dans le réseau. En considérant une série de données à un instant donné,xt, et un état caché précédent, ht−1, qui représente l'historique des données précédemment vues, le processus du MGU se déroule comme suit :

Porte d'oubli : La première étape consiste à décider ce que nous devons oublier de l'état caché précédent ht−1. Cela se fait en utilisant une fonction sigmoïde pour calculer la sortie de la porte d'oubli ft. Mathématiquement, cela peut être représenté comme suit :

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f)$$

où σ est la fonction sigmoïde, Wf et Uf sont des matrices de poids, xt est l'entrée actuelle, ht−1 est l'état caché précédent et bf est le biais de la porte d'oubli.

Mise à jour de l'état caché : Ensuite, nous mettons à jour l'état caché en utilisant les données actuelles et en prenant en compte ce que nous avons décidé de conserver de l'état caché précédent. Cela se fait en utilisant une fonction d'activation, comme la tangente hyperbolique tanh, pour calculer un nouvel état candidat ht, puis en combinant cela avec ce que nous avons décidé de conserver de l'état précédent. Mathématiquement, cela peut être représenté comme suit :

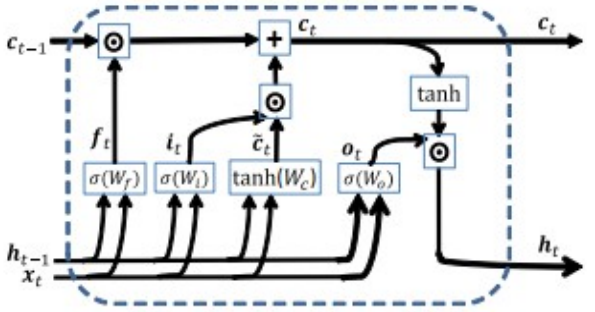
$$\hat{h}_t = \tanh(W_h \cdot x_t + U_h \cdot (f_t \odot h_{t-1}) + b_h)$$

où *h*Wh et *h*Uh sont des matrices de poids, ⊙représente le produit terme à terme, et bh est le biais.

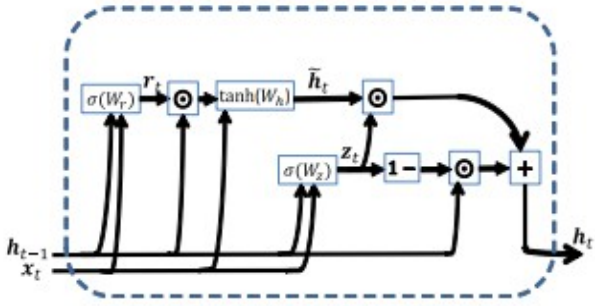
Calcul de la sortie : Enfin, nous combinons l'état caché mis à jour avec ce que nous avons décidé de conserver de l'état précédent pour obtenir l'état caché final ht. Mathématiquement, cela peut être représenté comme suit :

$$h_t = (1 - f_t) \odot h_{t-1} + f_t \odot \hat{h}_t$$

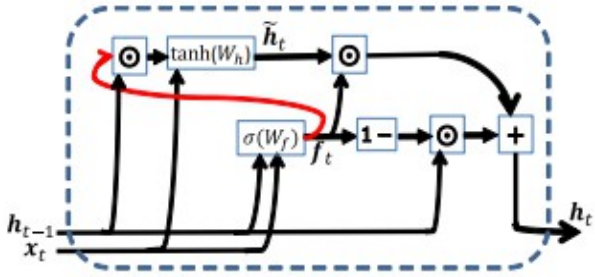
Ce processus permet au MGU de prendre en compte les données actuelles, de décider ce qu'il faut oublier des données précédentes, de mettre à jour son modèle interne en conséquence, et enfin de faire une prédiction sur le données futur.



(a) Long Short-Term Memory (LSTM)



(c) Gated Recurrent Unit (GRU)



(d) Minimal Gated Unit (MGU, the proposed method)

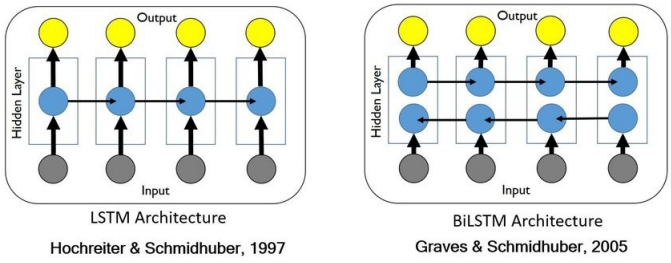
BI LSTM

Une autre variation importante est le Bi-LSTM, ou Long Short-Term Memory bidirectionnel, introduit par M. Schuster et al en 1997. Contrairement aux LSTM traditionnels qui traitent la séquence dans une seule direction, le Bi-LSTM analyse la séquence à la fois dans le sens ascendant et descendant, ce qui lui permet de capturer

plus efficacement le contexte temporel. Cette approche bidirectionnelle présente des avantages significatifs en intégrant à la fois les informations passées et futures à chaque étape de la séquence (Sima et al).

La figure 3 présente un schéma comparatif entre un réseau de neurones LSTM standard et BiLSTM deplus, une comparaison entre ces architectures est proposée par Sima Siami-Namini et al. [70]. Dans le LSTM, chaque cellule de la couche cachée reçoit une entrée basée sur les calculs de la cellule précédente dans la séquence, ce qui lui permet de mémoriser les informations séquentielles. En revanche, le BiLSTM permet un flux d'informations bidirectionnel, avec deux réseaux LSTM - un fonctionnant dans le sens avant et l'autre dans le sens inverse. Ces deux réseaux sont reliés à la même couche de sortie, offrant une précision améliorée pour la modélisation du langage, la classification de séquences, la traduction automatique, la reconnaissance vocale et le traitement du langage naturel [20, 21, 32, 33, 34, 70, 71].

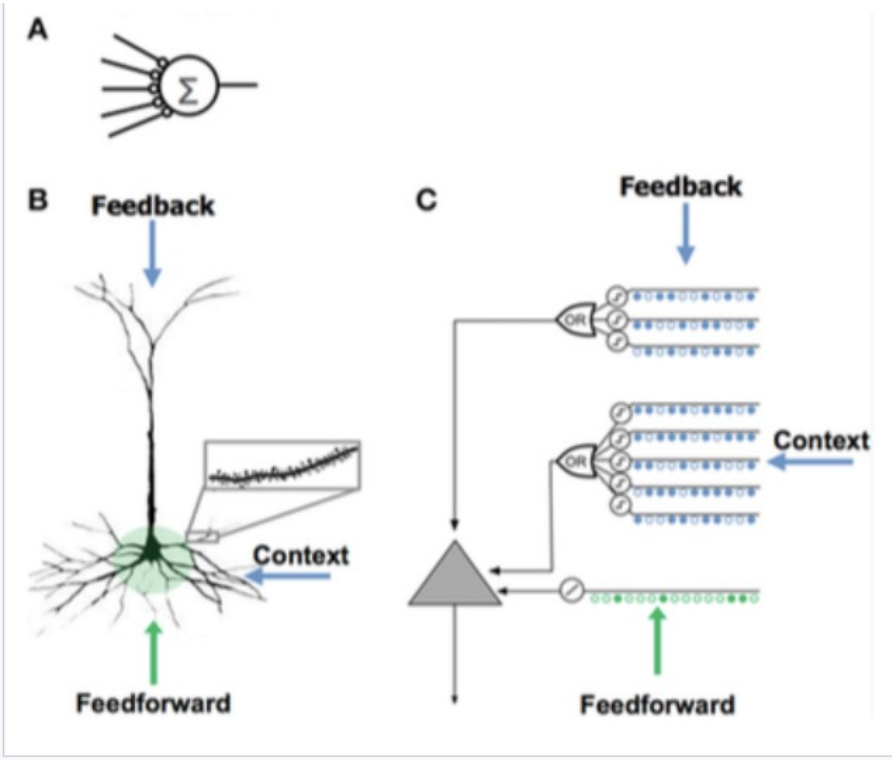
Cité figure : https://arxiv.org/pdf/1804.09269.pdf



HTM :

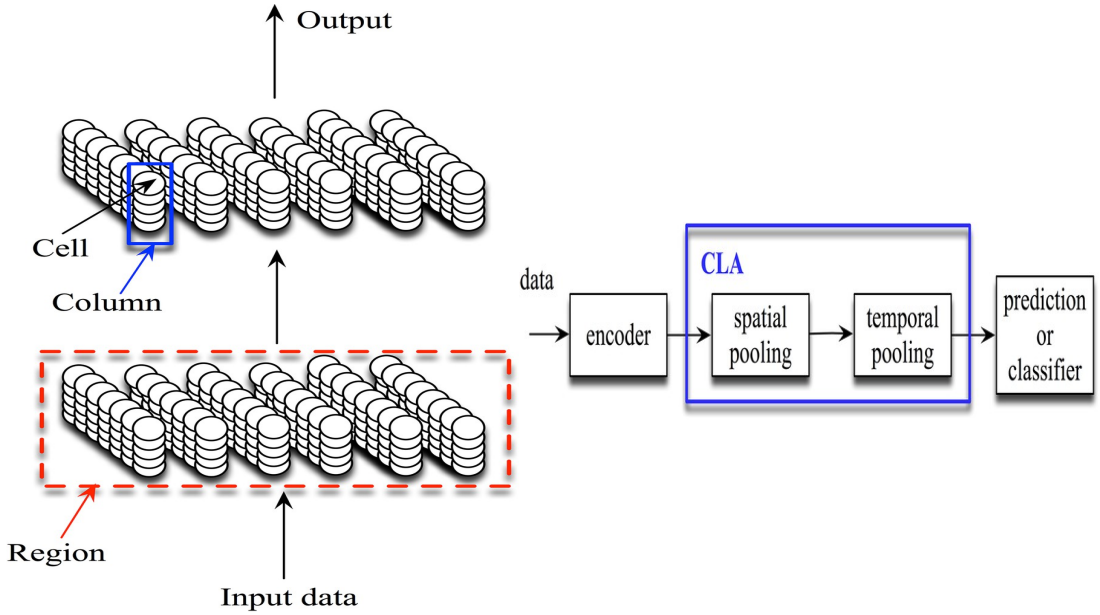
Les réseaux de mémoire temporelle hiérarchique (HTM) ont été présentés dans le livre "On intelligence" par Jeff Hawkins et Sandra Blakeslee en 2004 [22], puis développés et publiés par la société Numenta en 2017. Inspirés du fonctionnement des neurones du néocortex humain, les HTM sont composés de plusieurs couches appelées régions, chacune abritant des colonnes corticales formées de multiples neurones interconnectés. Le réseau HTM est formé par un algorithme d'apprentissage cortical (CLA). Les données d'entrée sont encodées en vecteurs parsemés, puis présentées à la première région où les colonnes de neurones s'activent en réponse aux motifs dans les données. Un processus de pooling spatial est ensuite appliqué pour simplifier la représentation des données, avant que les activations ne soient transférées aux régions supérieures pour un traitement ultérieur. Les neurones HTM peuvent recevoir trois types d'entrées : feedforward, feedback et contexte, permettant ainsi d'intégrer des informations contextuelles, des prédictions et des données temporelles dans le processus de traitement des données. Ces réseaux sont conçus pour l'apprentissage non supervisé et peuvent s'adapter de manière continue, étant robustes au bruit et capables d'apprendre plusieurs modèles simultanément. Ils sont notamment utilisés pour la détection d'anomalies, la classification [23], ainsi que pour explorer comment les robots mobiles autonomes peuvent apprendre à naviguer en

observant des démonstrations humaines [24]. Pour une analyse complète référez vous au travaux de Y Cui et al. [74]



Comparing the artificial neural network (A), the biological neuron (B), and the HTM neuron (C).

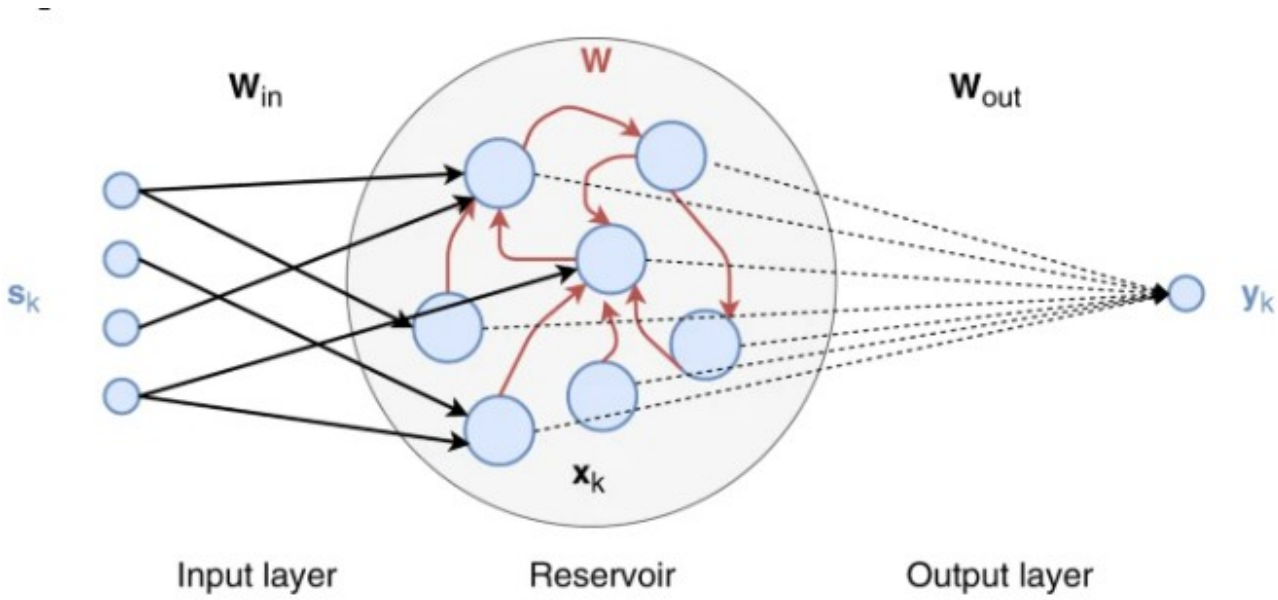
Wikipedia



Toward navigation ability for autonomous mobile robots with learning from demonstration paradigm: A view of hierarchical temporal memory [24]

ESN :

Les Echo State Networks (ESN) sont une classe de réseaux de neurones récurrents (RNN) introduite pour la première fois en 2002 par Herbert Jaeger [25].
L'architecture des ESN se distingue par la présence d'une couche de neurones récurrents, appelée réservoir, où les neurones sont connectés entre eux de manière aléatoire.



https://www.nature.com/articles/s41598-019-50158-4

Ces connexions sont pondérées par des poids qui restent fixes une fois initialisés de plus les connexion entre neurones ne possède pas de biais contrairement aux autre réseaux RNN. Un ESN peut être défini comme suis :

Calcul de l'état interne du réseau à chaque pas de temps k :

$xk=f(W \cdot xk-1+Win \cdot uk)$

Génération de la sortie à partir de l'état interne :

$yk=Wout \cdot xk$

Où :

xk est l'état du réseau au pas de temps k,

f est la fonction d'activation (généralement la fonction tangente hyperbolique),

W est la matrice de connexion interne du réservoir,

Win est la matrice de connexion des entrées vers le réservoir,

uk est le signal d'entrée au temps k,

yk est la sortie générée par le réseau au temps k,

Wout est la matrice de poids des connexions de sortie.

Lors de l'exécution du réseau, les données d'entrée traversent la couche d'entrée, où elles sont modifiées par les poids correspondants. Ensuite, ces données entrent dans le réservoir où elles interagissent avec les neurones récurrents. Dans les Echo State Networks (ESN), le nombre de connexions dans le réservoir est généralement bien inférieur au nombre total de connexions possibles entre les neurones. En pratique, seule une petite fraction des connexions possibles est activée. Cette caractéristique permet de réduire la complexité du réseau tout en maintenant sa capacité à capturer les dynamiques des données. Typiquement, entre 1% et 10% des connexions dans le réservoir sont actives, ce qui est souvent défini comme la densité de connexion du réseau. Les activations des neurones du réservoir sont combinées pour former une sortie finale en sommant les activations pondérées de chaque neurone du réservoir après avoir été calculées en appliquant une fonction non linéaire à la somme pondérée de leurs entrées. Les poids de la couche de sortie, qui sont ajustés lors de l'entraînement, déterminent l'importance de chaque activation du réservoir dans la sortie finale. Les ESN sont conçus de manière à ne pas impliquer de connexion de retour de sortie, ce qui signifie que les activations des neurones ne sont pas réinjectées dans le réseau pour influencer les activations futures. Cela différencie les ESN des autres types de RNN, car ils n'utilisent pas la rétropropagation du gradient à travers le temps.

CNN (3D, 1D, Conv Dilaté, TCN) :

CNN

L'introduction des Convolutional Neural Networks (CNN) en 1989 par Yann Le Cun et al. a marqué une révolution dans l'analyse de données 2D, telles que les images [43] ou encore les serie temporelles [44]. Ces réseaux ont permis une extraction efficace des caractéristiques spatiales à partir des images en utilisant des opérations de convolution. Les opérations de convolution consistent à appliquer des filtres (également appelés noyaux) à travers une image ou une autre forme de données 2D, produisant ainsi des cartes de caractéristiques qui mettent en évidence les motifs et les structures présents dans l'image. Mathématiquement, la convolution peut être exprimée comme suit :

$$C[i,j]=(I*K)[i,j]=\sum_m\sum_n I[i-m,j-n]\cdot K[m,n]$$

Ici, C[i,j] représente la valeur du pixel à la position (i,j) dans l'image de sortie, I est l'image d'entrée, K est le noyau de convolution, et $*$ indique l'opération de convolution.

En parallèle, les opérations de pooling sont utilisées pour réduire la dimensionnalité des cartes de caractéristiques générées par les couches de convolution. Cela permet de réduire la taille des données tout en conservant les informations les plus

importantes sur les motifs présents dans l'image. Typiquement, le pooling peut être effectué en utilisant des opérations telles que le max pooling ou le average pooling, où une fenêtre de pooling se déplace à travers la carte de caractéristiques et extrait la valeur maximale ou moyenne dans chaque région. Mathématiquement, le max pooling peut être représenté comme suit :

$$P[i,j]=\max_{m,n}(C[i\cdot s+m, j\cdot s+n])$$

Ici, P[i,j] représente la valeur de la sortie du pooling à la position (i,j), C est la carte de caractéristiques en entrée, s est le facteur de sous-échantillonnage, et (i·s+m,j·s+n) sont les indices dans la fenêtre de pooling. Ensemble, les opérations de convolution et de pooling des CNN permettent une représentation efficace des caractéristiques spatiales des données, ce qui en fait des outils puissants pour l'analyse de données 2D.

CNN + RNN / CONVRNN

Les architectures Convolutional GRU (ConvGRU) et Convolutional LSTM (CNN + LSTM) sont des variantes importantes utilisées pour modéliser des séries temporelles complexes. Dans l'architecture Convolutional LSTM, chaque image temporelle est traitée par des couches convolutives successives afin de générer un vecteur de caractéristiques qui capture les informations spatiales importantes. Ce vecteur de caractéristiques est ensuite traité par un réseau LSTM (Long Short-Term Memory) pour capturer les dépendances temporelles à long terme dans les données. En revanche, l'architecture ConvLSTM remplace les multiplications matricielles internes du LSTM par des calculs de convolution, ce qui lui permet de mieux modéliser les relations spatiales dans les séries temporelles. Ces architectures sont fréquemment utilisées dans des domaines tels que la reconnaissance d'actions [30, 45], de la détection d'anomalies [46, 51] la détection d'actions violentes [47, 48, 49, 50], prédiction de météorologique [35, 36] ou dans le domaine de la santé [37]. Bien que les modèles Convolutional GRU soient moins cités dans la littérature, ils sont généralement plus performants que les Convolutional LSTM et nécessitent également moins de puissance de calcul.

Convolution 1D / 3D

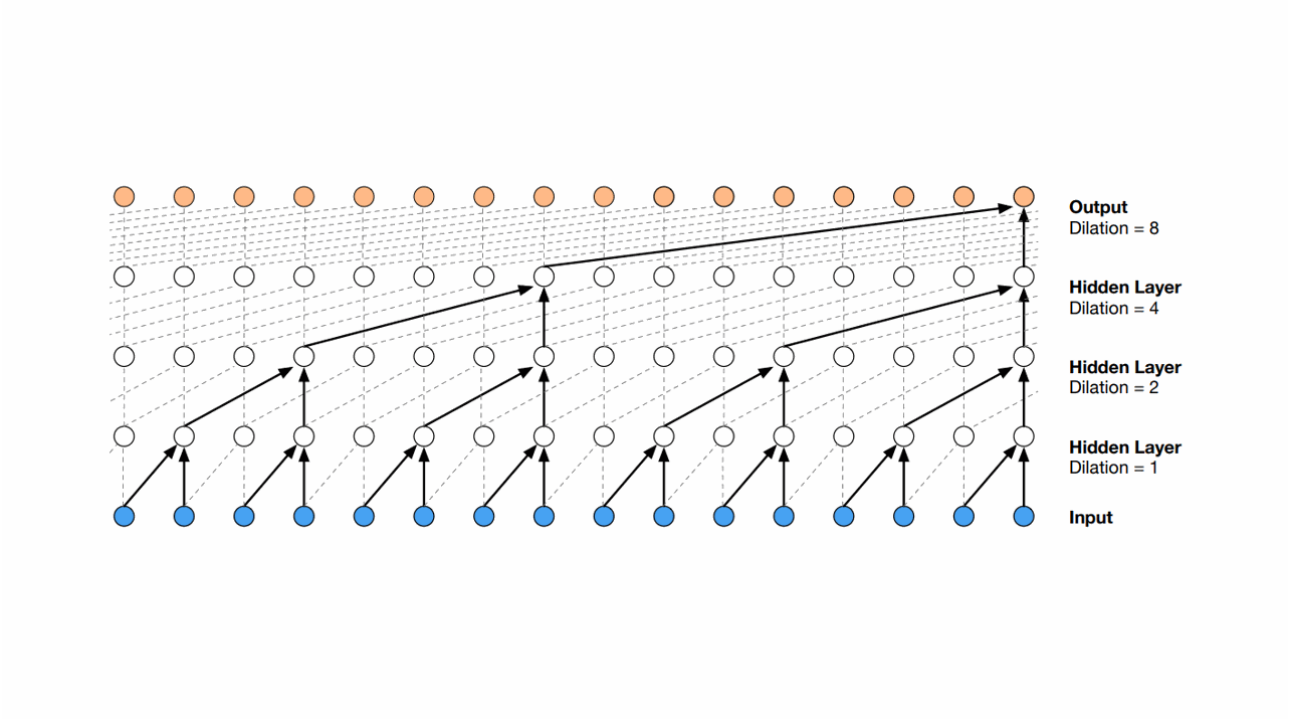
Les réseaux à convolution ont été étendus à différents types de données, notamment les données 1D telles que le texte ou les séries numériques, ainsi que les données 3D comme les vidéos. Une architecture notable dans ce domaine est le modèle à base de convolutions 3D (C3D) introduit par Ji et al.[52] pour la reconnaissance d'actions dans les vidéos. Contrairement aux approches de type Convolutional LSTM / GRU, qui réduisent chaque image en un vecteur de caractéristiques 1D, les convolutions 3D conservent les informations liées à la profondeur et aux changements inter-images. Ces convolutions se déplacent dans trois directions (x, y, z) pour produire une carte d'activation 3D, permettant d'analyser les différences entre les images successives et de capturer les informations relatives au mouvement. Les convolutions 3D ont été

largement utilisées dans divers domaines, tels que l'estimation de la pose humaine dans les images ou les vidéos, la reconnaissance d'actions [52, 53, 54] et la segmentation d'images médicales.

TCN

En parallèle, les Temporal Convolutional Networks (TCN) constituent une autre famille de modèles de réseaux de neurones convolutifs spécialement conçus pour modéliser les séquences temporelles.

Proposés par Colin Lea et al. en 2016 dans leur article intitulé "Temporal Convolutional Networks for Action Segmentation and Detection" [26], ces réseaux ont été développés pour capturer efficacement les dépendances temporelles dans les données séquentielles. Contrairement aux RNN traditionnels, qui utilisent des connexions récurrentes pour capturer la dépendance temporelle, les TCN utilisent des convolutions temporelles pour extraire les motifs temporels. Les filtres dilatés sont une composante clé des TCN, permettant de capturer des motifs temporels à différentes échelles. Ces filtres peuvent être causaux (en ne prenant en compte que les valeurs passées de la séquence) ou non causaux (en considérant à la fois les valeurs passées et futures de la séquence), selon les besoins de la tâche à accomplir. Les TCN ont été appliqués avec succès à des tâches telles que la segmentation et la détection d'actions dans les vidéos. Les résultats expérimentaux ont montré que les TCN sont capables de capturer des motifs temporels complexes et de reconnaître des actions même en présence de variations temporelles.



https://arxiv.org/abs/1609.03499

Les convolution dilatées, également appelées dilated convolutions, sont une extension des convolutions traditionnelles largement utilisées dans les réseaux de neurones convolutifs pour le traitement des séries temporelles. Elles permettent aux réseaux de capturer des motifs temporels à différentes échelles, ce qui en fait un outil puissant pour l'analyse de séquences temporelles complexes. Dans les Dilated Convolutional Networks (DCN), les filtres dilatés sont utilisés pour augmenter le champ réceptif sans augmenter le nombre de paramètres du modèle, ce qui est particulièrement utile pour la modélisation des séquences temporelles longues avec des dépendances à long terme. Les DCN ont été appliqués avec succès à diverses tâches de séries temporelles, telles que la prévision de la consommation énergétique, la prédiction des prix des actions et la reconnaissance de motifs dans les données biomédicales. L'une des architectures les plus connues est le réseau WaveNet, une architecture de réseau neuronal convolutif (CNN) développée par DeepMind, une filiale de Google, en 2016 [38]. Elle est principalement employée pour manipuler des données audio, notamment pour générer des signaux sonores tels que la parole ou la musique. Cependant, elle peut également être adaptée à d'autres types de séquences temporelles [39, 40, 41, 42].

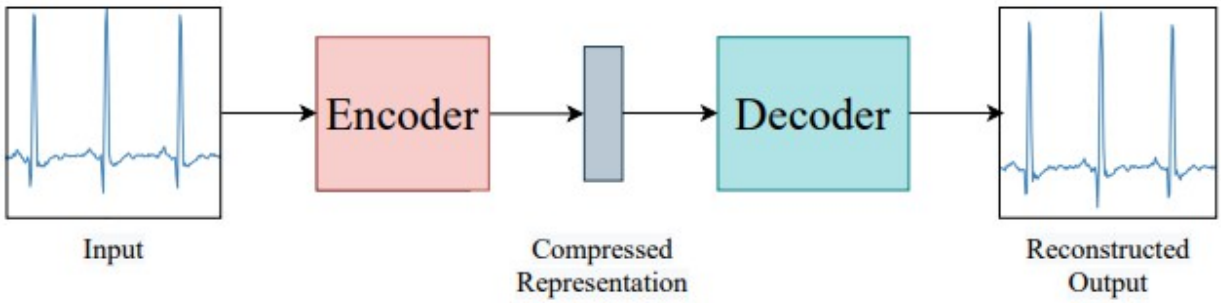
Capsule Network

Les capsule networks représentent une évolution des réseaux de neurones convolutionnels (CNN) visant à répondre à leurs limitations dues au pooling, permettant ainsi une modélisation plus précise des relations hiérarchiques dans les données. Proposés par Sara Sabour et al. en 2017 dans leur article intitulé "Dynamic Routing Between Capsules" [27], ces réseaux introduisent des "capsules" comme unités de base, permettant de représenter des entités plus complexes en préservant les informations spatiales relatives entre les caractéristiques extraites. Contrairement aux CNN traditionnels qui se contentent de renvoyer l'activation entre une carte et l'image pour indiquer la présence d'une caractéristique, les capsules retournent un vecteur contenant plusieurs informations telles que la caractéristique détectée, sa position et sa taille. Dans ce processus, chaque capsule de la couche précédente envoie son "vote" à toutes les capsules de la couche suivante, représentant ainsi les informations sur la présence et les attributs des entités dans l'image. Ces votes sont ensuite pondérés en fonction de leur accord avec les prédictions de chaque capsule de la couche suivante sur l'entité correspondante. Les poids des votes sont ajustés par rétropropagation du gradient pour renforcer les connexions entre les capsules qui sont d'accord sur les entités présentes et réduire les connexions avec celles qui ne le sont pas. Ce processus itératif de "routing by agreement" permet aux capsules de détecter et de représenter de manière robuste des entités hiérarchiques dans les données tout en préservant les informations sur leurs attributs spatiaux et structurels. Les résultats expérimentaux ont démontré que cette méthode offre des performances satisfaisantes dans la reconnaissance des émotions humaines à partir des signaux EEG [55] ou

encore la detection de fake pour des données vidéos [56].

AUTO-Encodeurs (AE / VAE):

La détection d'anomalies dans les séries temporelles est une tâche complexe en raison de la diversité des anomalies et de l'absence de données étiquetées dans de nombreux cas. Dans ces situations, les approches non supervisées, telles que les auto-encodeurs, se démarquent comme des solutions efficaces. Un auto-encodeur est composé d'un encodeur et d'un décodeur, où les données d'entrée sont encodées puis reconstruites par le décodeur. L'anomalie est détectée lorsqu'une donnée est difficile à reconstruire.



Architecture d'un auto encodeur [60]

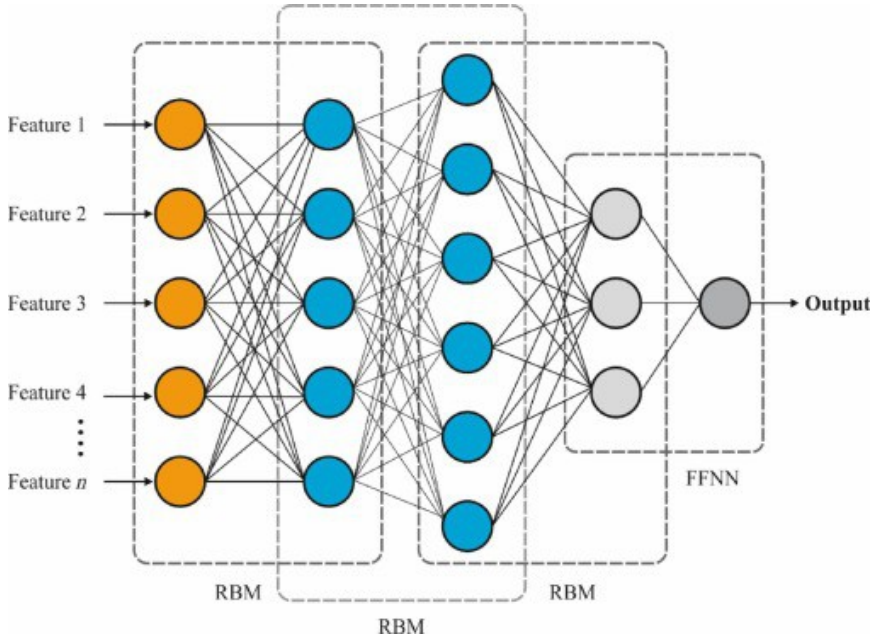
Les Variational Autoencoders (VAE) représentent une extension des auto-encodeurs classiques, développée par Diederik P Kingma et Max Welling en 2013 et présentée dans leur article intitulé "Auto-Encoding Variational Bayes" [28] ils introduisent une dimension probabiliste dans leur mécanisme d'encodage. Contrairement aux auto-encodeurs classiques qui génèrent une représentation latente unique pour chaque entrée, les VAE génèrent une distribution de probabilité dans l'espace latent. Cette propriété probabiliste permet aux VAE de générer de nouvelles données en échantillonnant aléatoirement dans cet espace latent, offrant ainsi une capacité de génération de données nouvelles et originales.

Dans le contexte de la détection d'anomalies, les auto-encodeurs, qu'ils soient traditionnels ou variationnels, sont souvent utilisés avec des architectures de convolution (CNN) 1D ou 2D pour traiter les séries temporelles. Cependant, la représentation compressée générée par l'encodeur peut entraîner une perte d'informations pertinentes. Pour surmonter cette limitation, une variante des auto-encodeurs utilise des LSTM convolutionnels, combinant ainsi les avantages des réseaux de neurones récurrents (RNN) et des auto-encodeurs pour reconstruire les données temporelles antérieures. Cette approche permet de capturer les dépendances temporelles complexes et de reconstruire de manière plus précise les données, améliorant ainsi les performances de détection d'anomalies [57, 58, 73]

Deep Belief Network :

Les Réseaux de Croyances Profondes (DBN) constituent une architecture de réseau

de neurones introduite en 2006 par Geoffrey Hinton, visant à réaliser de l'apprentissage non supervisé. Les DBN exploitent des couches de Restricted Boltzmann Machines (RBM) connectées dans les deux sens, où chaque neurone d'une couche est lié à tous les neurones de la couche suivante. Dans une DBM, chaque couche peut être interprétée comme un ensemble de variables cachées, où chaque variable cachée est connectée à toutes les variables observées de la couche précédente, mais pas aux autres variables cachées de la même couche. Cette organisation permet aux DBN d'apprendre efficacement des représentations des données non étiquetées, ce qui les rend particulièrement adaptés à l'extraction de caractéristiques significatives à partir de données complexes telles que des images, du texte ou des séries temporelles [59].



https://www.sciencedirect.com/topics/engineering/deep-belief-network

Pour entraîner les DBN, plusieurs calculs sont utilisés. Tout d'abord, le calcul de l'énergie d'un état donné de la RBM est crucial. L'énergie E(v,h|θ) est définie comme la somme des produits scalaires des états des neurones visibles et cachés, pondérée par les poids des connexions, et ajustée par les biais des neurones.

$$E(v,h|\theta)=-\sum_{i=1}^{n}\sum_{j=1}^{m}v_i h_j W_{ij}-\sum_{i=1}^{n}a_i v_i-\sum_{j=1}^{m}b_j h_j$$

où v est l'état de la couche visible, *h*h est l'état de la couche cachée, θ représente les paramètres du modèle (poids Wij, biais ai et bj), et n et m sont le nombre de neurones dans les couches visibles et cachées respectivement. Cette énergie est utilisée pour évaluer la probabilité des états des neurones à l'aide de la distribution de Boltzmann

Ensuite, lors de l'apprentissage non supervisé des DBN, les probabilités conditionnelles des états des neurones visibles et cachés sont calculées à l'aide du Gibbs Sampling. Par exemple, la probabilité qu'un neurone caché hj soit actif (c'est-à-

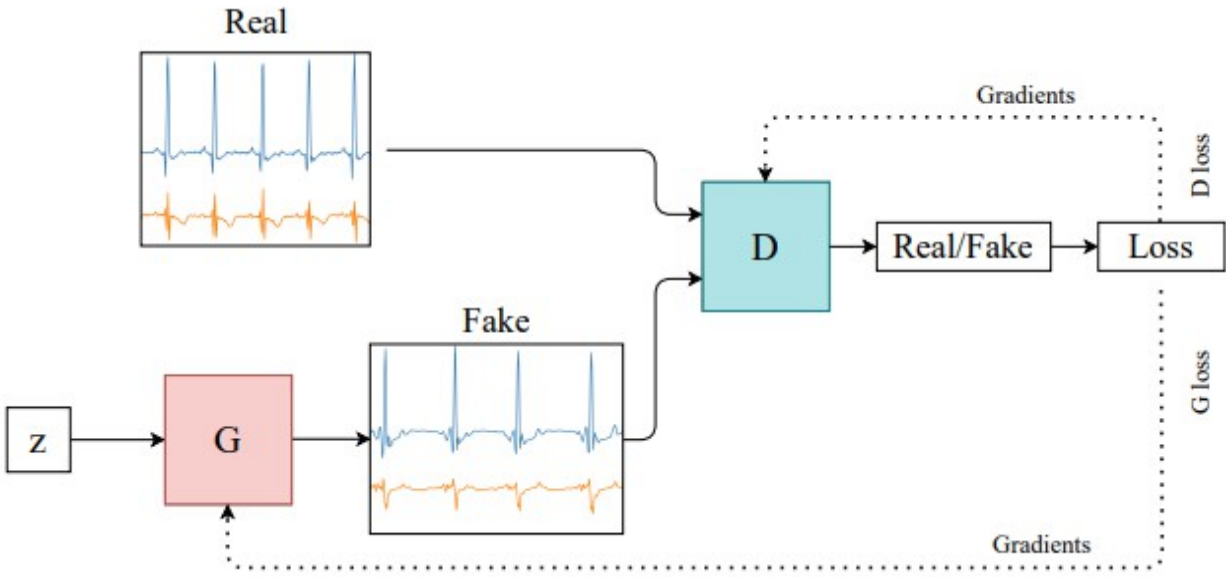
dire qu'il prenne la valeur 1) étant donné un état visible v est calculée à l'aide de la fonction sigmoid : P(hj=1|v)=sigmoid(bj+∑iviWij) De manière similaire, la probabilité qu'un neurone visible vi soit actif étant donné un état caché *h*h est calculée de la même manière. Enfin, lors de l'apprentissage supervisé (fine-tuning) des DBN, les gradients des poids du réseau sont calculés par rétropropagation du gradient. Ces gradients sont utilisés pour mettre à jour les poids du réseau afin de minimiser une fonction de perte définie, par exemple, la fonction de coût dans le cas de la classification.

GAN :

Initialement proposés par Goodfellow et al. (2014), les GAN se composent de deux sous-modèles : un générateur et un discriminateur (voir figure 1.12). Le rôle du générateur est de créer des données ressemblant à celles du jeu de données d'origine, tandis que le discriminateur cherche à distinguer les données authentiques des données générées. Au cours de l'apprentissage, les deux modèles sont améliorés par une optimisation alternée jusqu'à ce que le discriminateur ne puisse plus différencier les données synthétiques des données authentiques. La fonction de perte utilisée pour l'entraînement du générateur et du discriminateur dans un GAN est définie par l'équation suivante :

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p(X)}[log\ D(X_{t-w+1:t})] + \mathbb{E}_{z\sim p(Z)}[log(1 - D(Z_{t-w+1:t}))]$$

Dans cette équation, V(D,G) est la fonction de valeur utilisée dans le jeu minimax, où D est le discriminateur et G est le générateur. L'objectif est de minimiser la fonction de perte pour le générateur G tout en maximisant la fonction de perte pour le discriminateur D, ce qui permet d'atteindre un équilibre entre les deux composants du GAN. Les GAN offrent ainsi la possibilité de former un modèle sur un jeu de données pour ensuite générer synthétiquement de nouvelles données, puis d'utiliser le discriminateur pour identifier les données anormales. Leur capacité à générer des données similaires à celles du monde réel les rend également utiles dans les domaines où les données sont rares ou incomplètes.



Architecture d'un GAN [60]

Ce genre d'architecture est utilisé dans de nombreuses problématiques, telles que la réalisation de la data augmentation, la génération de données manquantes, la détection d'anomalies et le débruitage de données. Une taxonomie présentant les réseaux GAN ainsi que leurs diverses variantes, notamment GAN, Sequence GAN (SeqGAN), Quant GAN, Continuous RNN-GAN (C-RNN-GAN), Recurrent Conditional GAN (RCGAN), Sequentially Coupled GAN (SC-GAN), Noise Reduction GAN (NR-GAN), Time GAN, Conditional Sig-Wasserstein GAN (SigCWGAN), Decision Aware Time series conditional GAN (DAT-CGAN), et Synthetic biomedical Signals GAN (SynSigGAN), appliquée aux séries temporelles continues ou discrètes, a été proposée par Eoin Brophy et al. [60]

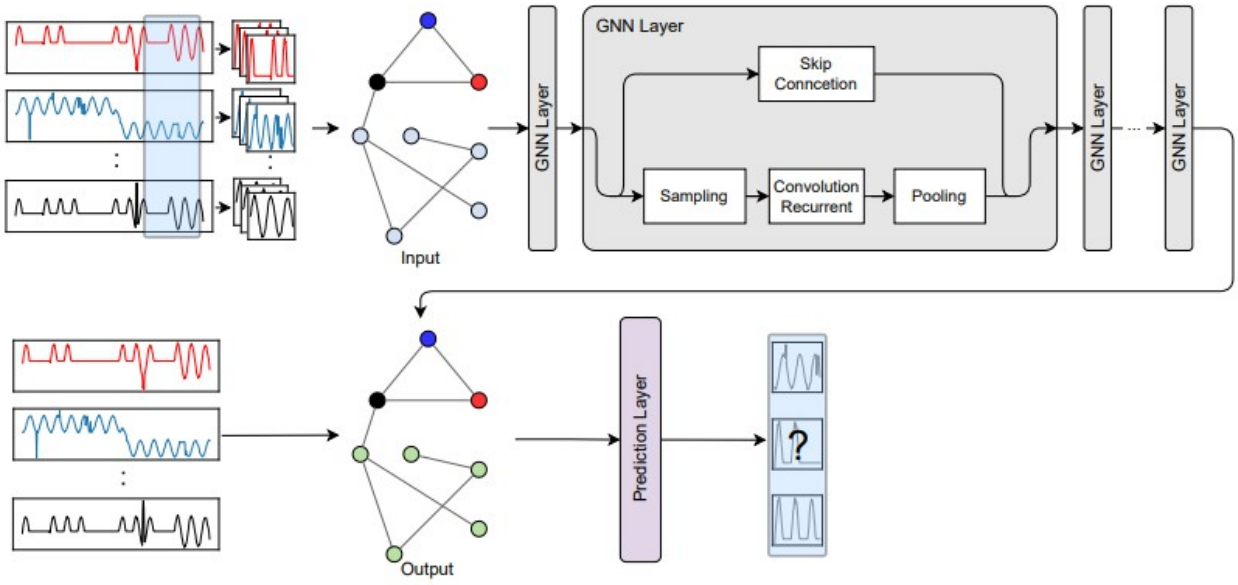
Pour une étude complète referez vous aux travaux de E Brophy et al. [76]

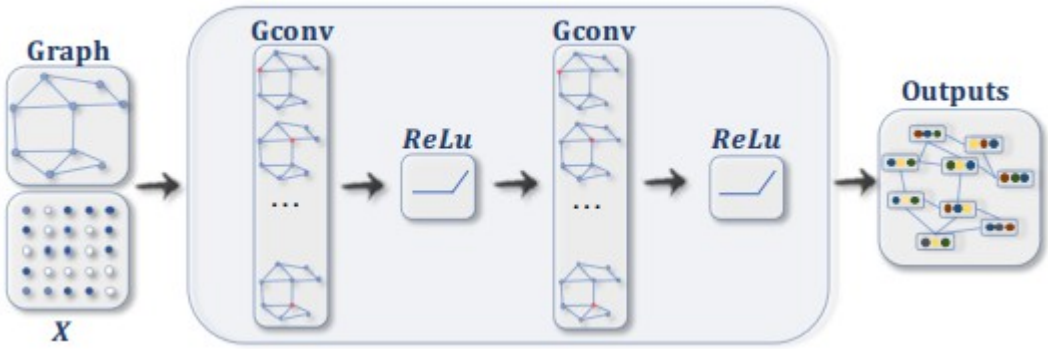
GNN :

Les Graph Neural Networks (GNN) sont une classe de modèles d'apprentissage automatique qui traitent des données structurées sous forme de graphes. Ils ont gagné en popularité ces dernières années en raison de leur capacité à capturer les relations complexes entre les entités dans divers domaines tels que la chimie, la biologie, la recommandation et l'analyse de réseau social. Les premiers travaux sur les GNN remontent à la fin des années 2000, mais leur adoption à grande échelle a eu lieu plus récemment, avec des avancées significatives dans la conception de modèles et les méthodes d'entraînement.

Le fonctionnement des GNN repose sur une architecture itérative où les représentations des nœuds dans le graphe sont mises à jour en fonction des représentations de leurs voisins. Tout d'abord, les graphes sont encodés sous forme de

matrices indiquant les connexions entre chaque nœud, ce qui constitue une représentation d'embedding. Ensuite, les couches de convolution graphique (Gconv / GNN layer) réalisent trois étapes clés : le passage de messages, au cours duquel les nœuds échangent des informations avec leurs voisins ; l'agrégation de ces messages pour obtenir une vision globale du voisinage ; et enfin, la mise à jour des représentations des nœuds en utilisant les informations agrégées. Cette approche permet aux GNN de capturer les structures de voisinage complexes et de réaliser des prédictions sur les données de graphe.





Les GNN sont utilisés pour résoudre une variété de problèmes, tels que la classification de nœuds, la prédiction de liens, la recommandation d'articles et la détection d'anomalies dans les réseaux. Leur capacité à modéliser les relations entre les entités fait d'eux des outils puissants pour analyser des données structurées et non structurées.

Il existe plusieurs variantes de GNN, chacune avec ses propres caractéristiques et avantages. Par exemple, les Graph Convolutional Networks (GCN) utilisent des couches de convolution pour capturer les motifs locaux dans le graphe, tandis que les Graph Recurrent Networks (GRN) utilisent des couches récurrentes pour modéliser les dépendances temporelles. D'autres variantes incluent les Graph Attention

Networks (GAT), qui utilisent des mécanismes d'attention pour pondérer les contributions des voisins lors du passage de messages, et les Graph Autoencoders (GAE), qui apprennent des représentations compressées des graphes. Chaque variante est adaptée à des types spécifiques de données et de problèmes, offrant ainsi une gamme d'approches flexibles pour la modélisation des graphes.

Pour plus de detail sur ces diverse architecture référer vous au travaux de JOHN A. MILLER et al. [60] ainsi que ceux de Zonghan Wu et al [63]

ELM :

Les ELM (Extreme Learning Machines) sont une forme particulière de réseaux neuronaux à propagation avant, qui se distinguent par leur architecture simple dotée d'une seule couche cachée. Conçue en 2006 par G. Huang, cette méthode présente une caractéristique essentielle : elle ne repose pas sur la rétropropagation du gradient pour ajuster ses poids. Contrairement aux approches traditionnelles basées sur le gradient, les ELM initialisent aléatoirement les poids et les biais de la couche cachée, qui restent constants pendant l'apprentissage. Les fonctions d'activation non linéaires dans cette couche garantissent la complexité du modèle. Au lieu de la rétropropagation du gradient, les ELM utilisent la méthode de Moore-Penrose pour déterminer leurs poids.

$$X^+ = (X^T X)^{-1} X^T$$

Cette méthode implique de calculer la transposée de la matrice de sortie de la couche cachée, de la multiplier par la matrice d'origine, puis de calculer l'inverse du résultat et de le multiplier à nouveau par la transposée de la matrice d'origine. Cette approche permet aux ELM d'apprendre efficacement les relations entre les entrées et les sorties sans nécessiter d'itérations pour ajuster les poids.

Les Extreme Learning Machines (ELM) se révèlent être des candidats prometteurs pour le traitement des séries temporelles, en particulier pour l'apprentissage continu. La version OS-ELM, développée par Huang et Liang [65] ainsi que Zhao et Wang [66], offre des solutions efficaces dans ce domaine. De plus, les variantes pondérées de ces modèles ELM, telles que celles proposées par Zong et Huang [67], permettent de gérer les jeux de données déséquilibrés, améliorant ainsi la performance des modèles dans des scénarios réels. Par ailleurs, certaines variantes des ELM offrent la possibilité de réaliser de l'apprentissage non supervisé, élargissant ainsi leur champ d'application dans le traitement des séries temporelles. Pour plus d'information sur ce modele et ces variantes référez vous aux travaux de Jian Wang, et al [64]

NODE / Recurent NODE :

TODO Mettre plus de reference ou l'article officiel suffit ?

Les Neural Ordinary Differential Equations (NODEs) sont une approche qui combine les concepts de réseaux de neurones artificiels et d'équations différentielles

ordinaires. Cette méthode a été introduite pour la première fois en 2018 par les chercheurs Ricky T. Q. Chen, Yulia Rubanova et Jeffrey Pennington dans leur article intitulé "Neural Ordinary Differential Equations". L'idée clé derrière les NODEs est de modéliser la propagation vers l'avant d'un réseau de neurones comme un problème d'intégration d'équations différentielles, offrant ainsi une représentation continue et flexible de la dynamique du système.

Mathématiquement, un NODE peut être décrit comme suit. On considère un réseau de neurones avec une entrée x et une sortie y. Au lieu de couches discrètes de neurones ou le calcul du gradient est effectué de manière itérative, couche par couche, un NODE définit une fonction d'encodage h(x, t), qui transforme l'entrée x en un espace latent à un temps donné t. L'évolution de cet espace latent est ensuite modélisée à l'aide d'une équation différentielle ordinaire (ODE) :

$dy/dt = f(y, t; \theta)$

Ici, y représente l'état du système, t est le temps, et θ sont les paramètres de l'ODE qui sont appris à partir des données. La fonction f, qui peut être un réseau de neurones profond, décrit la dynamique du système. La sortie du réseau est obtenue en résolvant numériquement l'ODE, ce qui donne y(t).

L'un des types de NODEs les plus intéressants est le NODE récurrent, qui étend le concept aux données séquentielles. Ces modèles, appelés Recurrent Neural Ordinary Differential Equations (RNODEs), peuvent traiter des entrées séquentielles en intégrant l'état du système au fil du temps. L'équation ODE pour un RNODE peut être formulée comme suit :

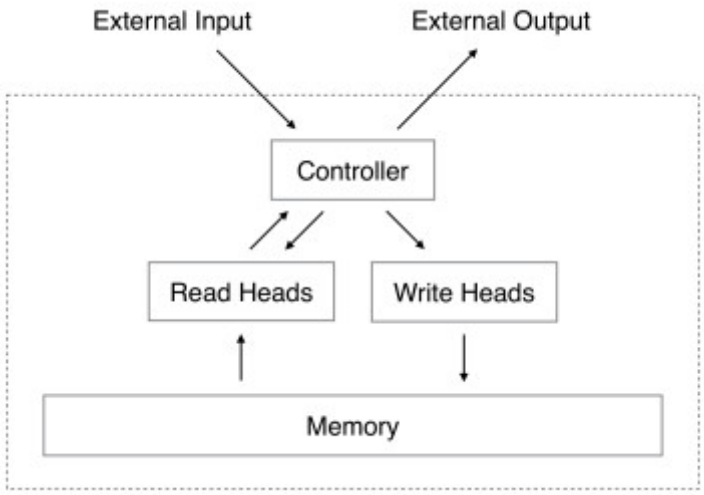
$dy/dt = f(y, x\_t, t; \theta)$

Ici, x_t représente l'entrée du réseau à l'instant t, et l'état y dépend à la fois de l'entrée actuelle et des états précédents. Les RNODEs sont particulièrement adaptés aux tâches impliquant des séries chronologiques ou du traitement du langage naturel, où l'ordre et la dépendance temporelle des données sont importants.

Les NODEs ont trouvé diverses applications, notamment dans la modélisation de systèmes dynamiques, la génération d'images, la prédiction de séries chronologiques et la modélisation de données de séries chronologiques.

Neural Turing Machines (NTM)

Les Neural Turing Machines (NTM) et les Differentiable Neural Computers (DNC) sont deux types d'architectures de réseau neuronal profond inspirées du concept de la machine de Turing, une machine théorique capable d'exécuter n'importe quel algorithme. Les NTMs, présentés pour la première fois en 2014 par Graves et al., ont une architecture similaire à celle d'un réseau neuronal standard, mais avec une mémoire externe explicitement gérée. Le Neural Turing Machine (NTM) est constitué de quatre éléments principaux : le contrôleur, un module d'adressage, un module d'écriture et un module de lecture, ainsi que la mémoire externe une matrice 2D fonctionant comme une table de hachage.



Représentation simplifier trouvable dans l'article officiel

Dans l'architecture d'un Neural Turing Machine (NTM), le contrôleur, qui agit comme un réseau de neurones récurrents, reçoit à la fois les données d'entrée et les données lues de la mémoire. Les sorties générées par le contrôleur sont transmises au module d'écriture et d'adressage qui forme les poids **w** pour déterminer quels emplacements de la mémoire doivent être lus ou écrits.

Ensuite, le module d'écriture, en se basant sur les sorties du contrôleur et la matrice de la mémoire externe, décide quels éléments de la mémoire doivent être effacés et écrits, suivant des instructions préalablement communiquées par le contrôleur. Cette opération d'écriture est décomposée en deux parties : une effacement suivi d'un ajout. Étant donné un poids **w(t)** émis par une tête d'écriture à l'instant **t**, ainsi qu'un vecteur d'effacement **e(t)** dont tous les éléments **M** se situent dans l'intervalle (0, 1), les vecteurs de mémoire **M(t-1)(i)** de l'instant précédent sont modifiés comme suit :

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right],$$

où 1 est un vecteur ligne de tous les 1, et la multiplication contre l'emplacement de la mémoire agit point par point. Par conséquent, les éléments d'un emplacement de mémoire sont réinitialisés à zéro uniquement si le poids à l'emplacement et l'élément d'effacement sont tous deux égaux à un ; si le poids ou l'effacement est zéro, la mémoire reste inchangée.

Chaque tête d'écriture produit également un vecteur d'ajout de longueur **M** à l'instant **t**, qui est ajouté à la mémoire après que l'étape d'effacement a été effectuée :

$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Par la suite, le module de lecture utilise les poids **w** pour déterminer quelles données doivent être lues dans la mémoire. Les données lues **r** sont calculées à chaque instant **t** en prenant une somme pondérée des éléments de la mémoire **m**, où les poids **w(i)** indiquent l'importance de chaque élément dans la lecture. Cette formule est exprimée comme suit :

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$$

Les sorties du module d'écriture forment la nouvelle mémoire, tandis que les sorties du module de lecture deviennent les nouvelles données fournies au contrôleur pour l'itération suivante.

Differentiable Neural Computers (DNC)

Les DNCs, ou ordinateurs neuronaux différentiables, ont été introduits en 2016 comme une extension des NTMs avec des mécanismes d'attention mémoire et temporelle supplémentaires, leur permettant de stocker et de gérer l'information de manière plus sophistiquée. Contrairement aux NTMs, qui se reposent uniquement sur des réseaux neuronaux pour effectuer des opérations symboliques, les DNCs intègrent une mémoire différentielle qui peut être mise à jour lors de l'apprentissage. Cette mémoire différentielle peut être considérée comme une extension de la mémoire externe des NTMs, mais avec des fonctionnalités supplémentaires pour gérer l'information de manière plus dynamique et complexe.

Mathématiquement, la mémoire différentielle dans un DNC peut être représentée par une matrice externe **M** de taille **N×W**, où **N** est le nombre d'emplacements mémoire et **W** est la dimensionnalité de chaque emplacement. Lors de l'opération de lecture, une tête de lecture utilise un mécanisme d'attention différentiable pour sélectionner les emplacements mémoire pertinents. Cela peut être réalisé en utilisant une clé de recherche **k** qui est comparée au contenu de chaque emplacement mémoire pour calculer une distribution de similarité **s**. Cette distribution de similarité est ensuite utilisée pour pondérer les emplacements mémoire, produisant un vecteur de lecture **r** qui est une somme pondérée des emplacements mémoire. Cette opération peut être exprimée mathématiquement comme suit :

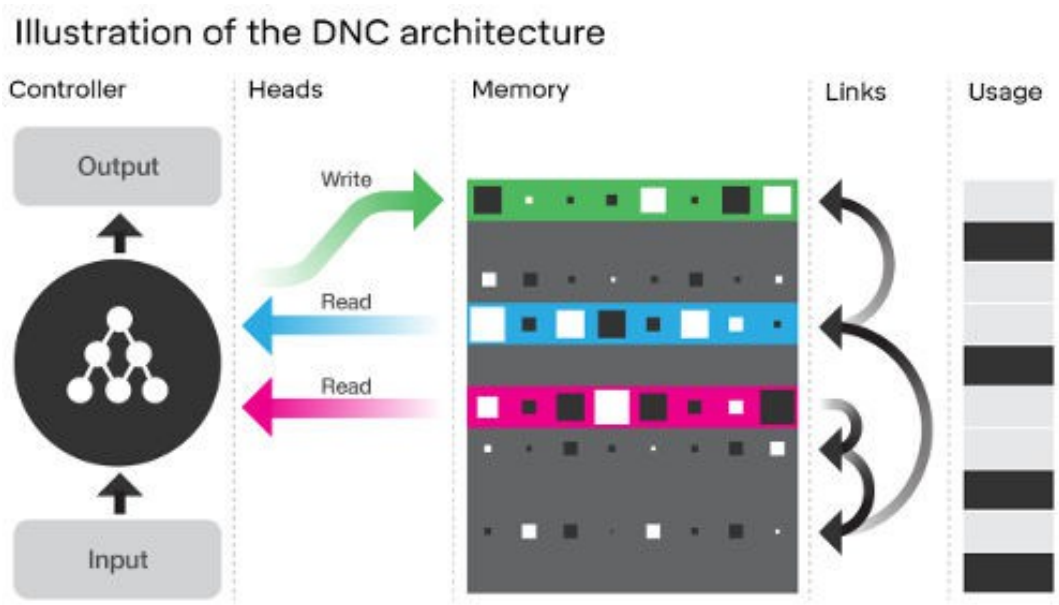
$$r = \sum_{i=1}^{N} s(i) \cdot M(i)$$

Dans un DNC, les mécanismes d'attention permettent également la gestion des liens temporels entre les emplacements mémoire. Une matrice de liens temporels **L** de

taille **N×N** est utilisée pour enregistrer les transitions entre les emplacements mémoire consécutivement écrits. Cette matrice permet au DNC de retrouver séquentiellement les séquences dans l'ordre dans lequel elles ont été écrites, même lorsque les écritures consécutives ne se produisent pas dans des étapes temporelles adjacentes. Si **L(t-1)** est la matrice de liens à l'instant **t-1**, alors la matrice de liens à l'instant **t** peut être mise à jour comme suit :

$$L(t) = (1 - w(t) - w(t)^T) \cdot L(t-1) + w(t) \cdot p(t)^T$$

où **p(t)** est le précédent poids d'écriture et **w(t)** est le poids d'écriture actuel. Cette formule permet de mettre à jour les liens entre les emplacements mémoire en fonction des poids d'écriture.



Pointeur network :

Les réseaux de pointeurs, également connus sous le nom de Pointer Networks, ont été introduits par Oriol Vinyals, Meire Fortunato et Navdeep Jaitly en 2015. Ils ont été présentés dans l'article intitulé "Pointer Networks" lors de la conférence Neural Information Processing Systems (NeurIPS) en 2015. Ces réseaux, peuvent être considérés comme des modèles séquence-à-séquence (seq2seq) dotés d'un mécanisme d'attention. Ils se distinguent des modèles seq2seq classiques par leur structure et leur mode de fonctionnement. Un réseau de pointeurs se compose de trois éléments clés :

L'encodeur, généralement un réseau de neurones récurrent (RNN) comme un LSTM ou un GRU, traite chaque mot de la séquence d'entrée un par un. Pour chaque mot, il reçoit un vecteur d'embedding représentant le mot lui-même ainsi que sa position dans la séquence. En utilisant cette information, l'encodeur encode chaque mot dans un espace latent, prenant en compte à la fois le mot et son contexte dans la séquence.

En résultat, il capture les relations et les dépendances entre les mots, produisant ainsi une représentation vectorielle de l'ensemble de la séquence.

Le décodeur utilise l'état final de l'encodeur comme point de départ. Il applique ensuite un mécanisme d'attention pour pondérer l'importance de chaque mot encodé en fonction du contexte actuel. Contrairement aux modèles seq2seq classiques, le décodeur ne génère pas une distribution de probabilité sur les mots encodés. Au lieu de cela, il utilise le mécanisme d'attention pour sélectionner un mot spécifique de la séquence encodée comme pointeur. Ce pointeur représente la position du mot choisi dans l'embedding, indiquant ainsi quel mot a été sélectionné.

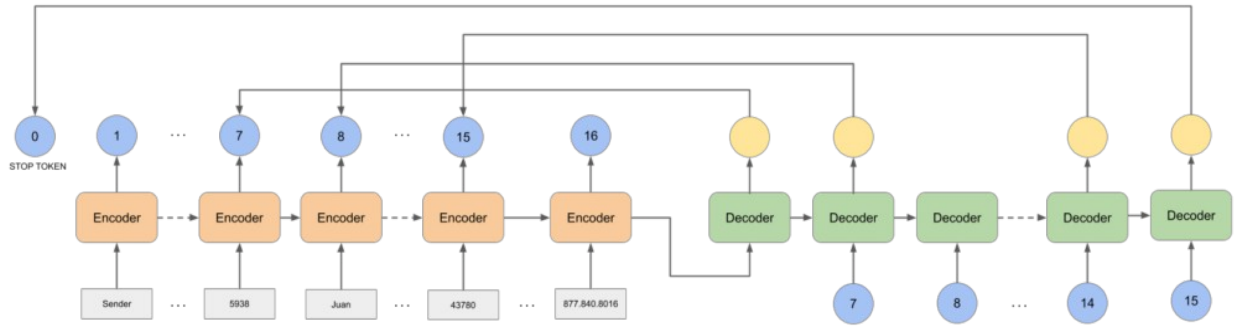
•Le mécanisme d'attention permet au décodeur de se concentrer sur les parties les plus pertinentes de la séquence encodée lors de la génération de la séquence de sortie. Pour résoudre les problèmes d'optimisation combinatoire où la taille du dictionnaire de sortie dépend du nombre d'éléments dans la séquence d'entrée, ce mécanisme d'attention est proposé comme suit :

$$u_{ij} = v^T \tanh(W_1 e_j + W_2 d_i) \quad \text{pour } j \in (1, ..., n)$$
$$p(C_i | C_1, ..., C_{i-1}, P) = \text{softmax}(u_i)$$

uij représente l'importance de chaque élément de l'entrée pour prédire le prochain élément de sortie. Le vecteur **v** quantifie l'importance globale de chaque élément de l'entrée pour la prédiction, et il est utilisé pour pondérer l'importance des éléments individuels. Les matrices de poids **W1** et **W2** sont apprises par le modèle pour ajuster l'importance des éléments de l'entrée. Les vecteurs **ej** et **di** représentent respectivement les états de l'encodeur et du décodeur à un certain moment donné, contenant des informations sur les éléments de l'entrée et de la sortie jusqu'à ce moment-là. La fonction **tanh** est utilisée pour introduire une non-linéarité dans le modèle, en comprimant les informations entre -1 et 1.
p(Ci | C1, ..., Ci−1, P) est la probabilité que le prochain élément de sortie soit **Ci**, étant donné les éléments précédents de la sortie (de C1 à Ci−1) ainsi que l'entrée **P**. **Softmax** est utilisé pour normaliser le vecteur **ui** (de longueur **n**) en une distribution de sortie sur le dictionnaire d'entrées. Dans ce contexte, l'état de l'encodeur **ej** n'est pas utilisé pour transmettre des informations supplémentaires au décodeur. Au lieu de cela, **uij** est utilisé comme des pointeurs vers les éléments de l'entrée, indiquant quels éléments sont les plus pertinents pour prédire **Ci**. De même, pour conditionner sur **Ci−1**, P**Ci−1** le dernier élément prédit est utilisé comme entrée.

Le décodeur utilise ensuite le mot sélectionné comme base pour générer le mot suivant de la séquence de sortie. Ce processus est répété itérativement jusqu'à ce que la séquence de sortie souhaitée soit générée.

Les réseaux de pointeurs sont particulièrement utiles pour des tâches telles que le résumé automatique, la traduction automatique et la réponse aux questions. Leur capacité à se concentrer sur les parties les plus pertinentes de la séquence d'entrée leur permet de générer des sorties plus précises et cohérentes.

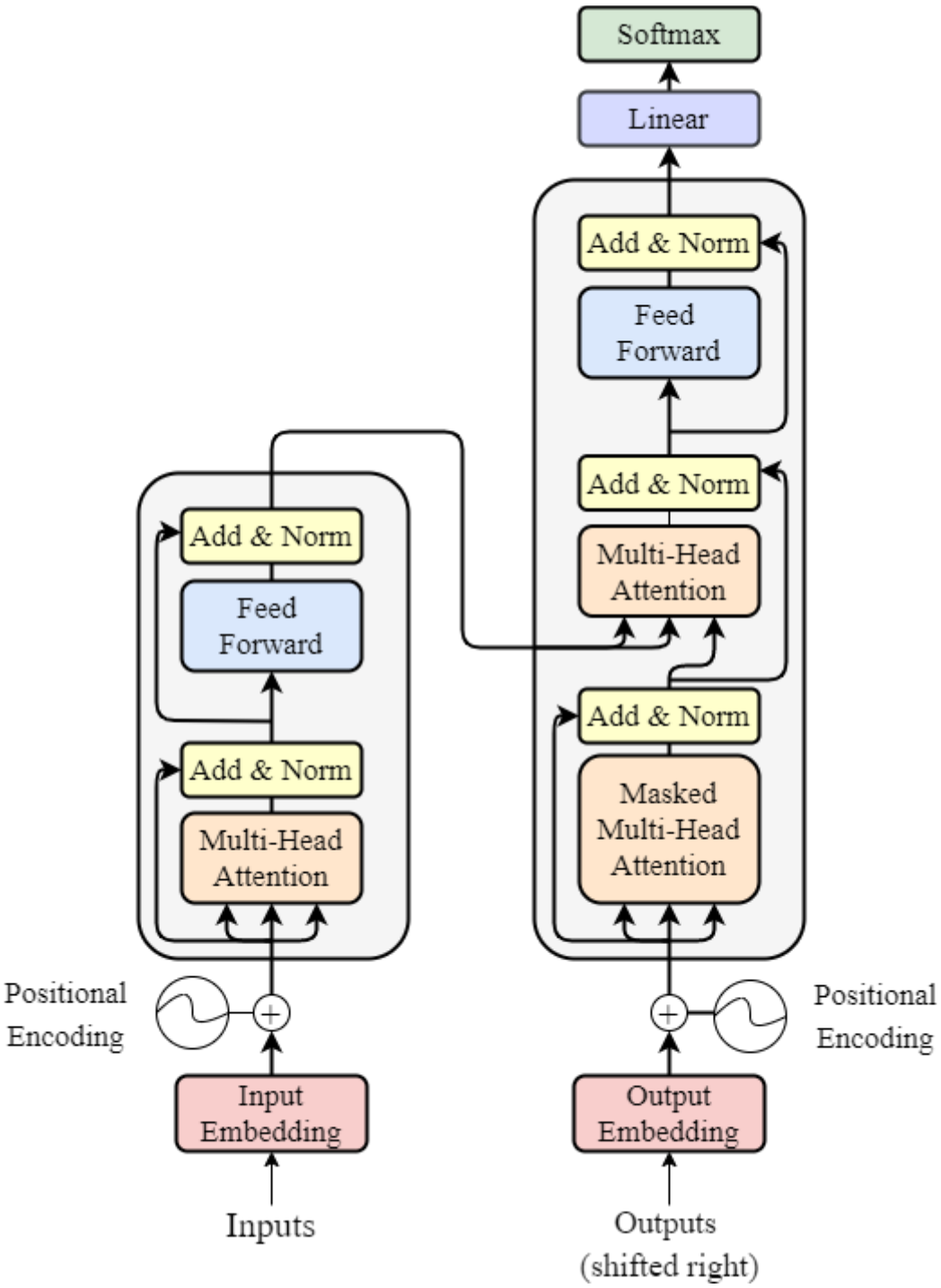


https://www.hyperscience.com/blog/the-power-of-pointer-networks/

Transformer :

Les Transformers, introduits par Vaswani et al. en 2017, via l'article "Attention is all you need" ont révolutionné le domaine du traitement du langage naturel (NLP) et de la modélisation séquentielle. Ils se distinguent des architectures traditionnelles par leur utilisation de mécanismes d'attention, abandonnant les unités récurrentes comme les LSTM ou les GRU. Cette approche a permis d'obtenir des résultats de pointe dans une large gamme de tâches, notamment la traduction automatique, le résumé automatique, la réponse aux questions et la génération de texte.

L'architecture d'un Transformer repose sur un empilement d'encodeurs et de décodeurs. Chaque encodeur comprend des couches d'attention multi-têtes, qui permettent au modèle de se concentrer simultanément sur différentes parties de la séquence d'entrée, capturant ainsi les relations à longue distance. Ces couches sont suivies de couches de feed-forward, contenant des réseaux de neurones à propagation avant, ainsi que de connexions résiduelles et de normalisation de couche pour faciliter l'entraînement en profondeur. Le décodeur, quant à lui, comprend des couches d'attention multi-têtes similaires à celles de l'encodeur, des couches d'attention croisée pour accéder aux informations de l'encodeur, des couches de feed-forward et des connexions résiduelles avec normalisation de couche pour une stabilité accrue.



Un modèle Transformer traite une séquence d'entrée, généralement composée de mots ou de jetons, pour en extraire des informations et produire une sortie. Le processus se déroule en plusieurs étapes clés :

1. Préparation de l'entrée :
•Tokenisation : La séquence d'entrée est découpée en jetons, qui sont les unités de base que le modèle va traiter.
•Encodage des jetons : Chaque jeton est converti en un vecteur de taille fixe, représentant ses caractéristiques sémantiques.

•Codage positionnel : Comme les Transformers n'ont pas de mémoire interne, il est nécessaire de spécifier la position relative de chaque jeton dans la séquence. Cela peut se faire par des techniques comme le codage absolu ou relatif.

2. Encodage :
•Couches d'encodeur : La séquence d'entrée passe par un empilement de couches d'encodeur. Chaque couche d'encodeur est composée de deux sous-couches principales :
•Couche d'attention multi-têtes : Cette couche permet au modèle de se concentrer sur différentes parties de la séquence d'entrée en calculant une somme pondérée des vecteurs de jetons. Les poids sont déterminés par la similarité entre les jetons, permettant au modèle de capturer les relations à longue distance.
•Couche de réseau neuronal à feed-forward : Cette couche applique des transformations non linéaires aux vecteurs de sortie de la couche d'attention multi-têtes, permettant au modèle d'extraire des informations plus complexes.
•Normalisation et connexions résiduelles : Des mécanismes de normalisation et de connexions résiduelles sont utilisés pour stabiliser l'apprentissage et améliorer les performances du modèle.

3. Décodage (pour les tâches de génération de texte) :
•Couches de décodeur : Si la tâche implique la génération de texte, le modèle utilise un empilement de couches de décodeur. Chaque couche de décodeur est similaire aux couches d'encodeur, mais avec des mécanismes d'attention supplémentaires :
•Attention propre : Permet au modèle de se concentrer sur les parties pertinentes de la séquence générée jusqu'à présent.
•Attention croisée : Permet au modèle d'accéder aux informations de la séquence d'entrée encodée lors de la génération de chaque jeton de sortie.
•Génération de sortie : A chaque étape de décodage, le modèle prédit le jeton suivant de la séquence de sortie en se basant sur les sorties des couches d'attention et du réseau neuronal à feed-forward.

4. Sortie :
•Têtes de sortie : Les dernières couches du modèle, appelées têtes de sortie, sont spécifiques à la tâche. Elles transforment les représentations internes du modèle en une sortie adaptée à la tâche, comme la prédiction d'une classe dans une tâche de classification ou la génération d'une séquence de mots dans une tâche de traduction.

L'auto-attention est le mécanisme clé qui distingue les Transformers des autres architectures d'apprentissage profond. Par defaut l'attention est calculer comme suivant Étant donné une matrice de requêtes Q, une matrice de clés K et une matrice de valeurs V, l'attention est calculée comme suit : attention(Q, K, V) = softmax(QKT/ √dK)V.

Il existe plusieurs type types d'attention utilisés dans les Transformers :

1. Attention scalaire : L'attention scalaire calcule un score de similarité entre deux

vecteurs, généralement à l'aide d'un produit scalaire. Ce score est ensuite utilisé pour pondérer l'importance d'un vecteur par rapport à l'autre. Les vecteurs avec des scores d'attention plus élevés ont une influence plus importante sur la sortie.

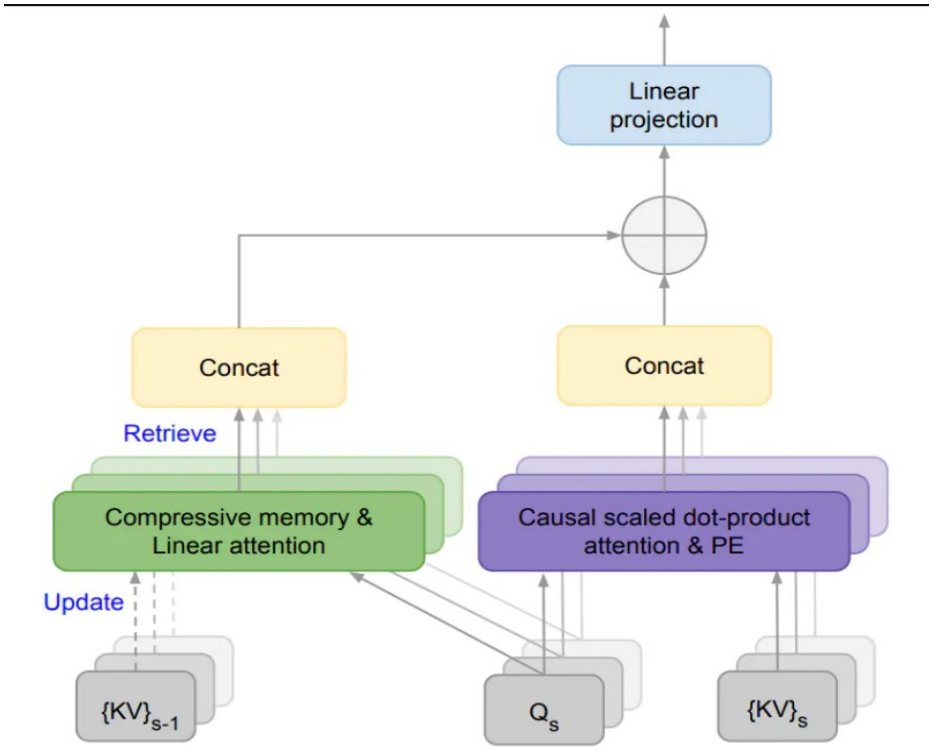
2. Attention additive : L'attention additive utilise une fonction d'alignement, généralement une couche de réseau neuronal à feed-forward, pour calculer les scores d'attention. La fonction prend deux vecteurs en entrée et produit un score d'attention qui capture la similarité ou la relation entre eux.
3. Attention multi-têtes : L'attention multi-têtes est une extension de l'auto-attention. Il divise la couche d'auto-attention en plusieurs « têtes », chacune apprenant une représentation d'attention distincte. Cela permet au modèle de capturer différents aspects ou perspectives des interactions entre les éléments de la séquence. Les représentations des différentes têtes sont ensuite concaténées et transformées pour produire la sortie finale.
4. Attention pondérée : Ce type d'attention implique d'attribuer des poids différents aux entrées en fonction de leur importance. Les poids peuvent être appris ou déterminés par des fonctions spécifiques, mettant ainsi l'accent sur les éléments les plus pertinents de la séquence.
5. Attention hiérarchique : L'attention hiérarchique traite des structures de données complexes ou hiérarchiques en appliquant l'attention à différents niveaux de granularité. Il peut capturer des dépendances à différents niveaux, tels que des mots, des phrases ou des paragraphes dans une hiérarchie de texte.
6. Attention axée sur l'information : Ce type d'attention se concentre sur la capture de dépendances informatives entre les éléments d'entrée. Il met l'accent sur les informations pertinentes pour une tâche donnée, ignorant les détails moins importants. Cela peut aider le modèle à se concentrer sur les aspects les plus pertinents des données.
7. Attention dynamique : L'attention dynamique implique d'adapter les poids d'attention en fonction du contexte ou de l'état actuel du modèle. Les poids d'attention ne sont pas statiques, mais évoluent dynamiquement au fur et à mesure que le modèle traite les données, permettant ainsi des représentations plus flexibles et adaptatives.
8. Attention auto-adaptative : L'attention auto-adaptative permet au modèle d'apprendre automatiquement les poids d'attention au lieu de les calculer à l'aide de fonctions fixes prédéfinies. Le modèle peut ainsi apprendre les poids d'attention les plus appropriés pour une tâche ou un ensemble de données spécifique.
9. Auto-attention : L'auto-attention, également connue sous le nom d'auto-attention auto-régressive ou d'auto-attention intra-séquence, permet à chaque élément d'une séquence d'interagir et d'attirer l'attention sur tous les autres éléments. Il calcule une représentation pondérée de la séquence, capturant les interactions et les dépendances entre les éléments. L'auto-attention est au cœur

des architectures Transformer et permet au modèle de traiter efficacement des séquences longues en capturant des dépendances à longue distance.

L'attention infinie récemment proposée par Google AI dans leur article "Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention"[68] vise à surmonter la limitation de taille fixe des fenêtres d'attention dans les modèles Transformer classiques. Il permet aux modèles de traiter des séquences d'entrée extrêmement longues avec une attention sur tout le contexte, d'où le nom « infini ».

L'Infini-attention incorpore une mémoire compressive dans le mécanisme d'attention classique et intègre à la fois une attention locale masquée et des mécanismes d'attention linéaire à long terme dans un seul bloc de Transformateur.

•**La mémoire compressive** : permet de stocker et de récupérer efficacement les informations les plus importantes du contexte.
•**L'attention locale masquée** : permet au modèle de se concentrer sur les parties les plus pertinentes du contexte, en fonction de la tâche en cours.
•**L'attention linéaire à long terme** : permet au modèle de prendre en compte des informations à long terme, même si elles sont éloignées dans le texte.



Mecanisme d'attention infini [68]

{KV}s−1 et {KV}s représentent respectivement les clés et les valeurs d'attention

pour les segments d'entrée actuels et précédents, tandis que Qs représente les requêtes d'attention. PE désigne les embeddings de position.

Les anciens états KV de l'attention sont stocker dans la mémoire compressive, au lieu de les supprimer comme dans le mécanisme d'attention standard. Ainsi, chaque couche d'attention des Infini-Transformers possède à la fois des états compressifs globaux et des états fins locaux. Au lieu de calculer de nouvelles entrées mémoire pour la mémoire compressive, les états de requête, de clé et de valeur (Q, K et V) issus du calcul de l'attention par produit scalaire sont réutilisé. Le partage et la réutilisation des états entre l'attention par produit scalaire et la mémoire compressive permettent non seulement une adaptation efficace à long contexte mais accélèrent également l'entraînement et l'inférence. Ensuite les valeurs de la mémoire sont récupérées en utilisant les états de requête d'attention lors du traitement des séquences suivantes. Pour calculer la sortie contextuelle finale, l'Infini-attention agrège les valeurs récupérées de la mémoire à long terme et les contextes d'attention locaux.

Vision transformer

Après avoir fait leurs preuve dans le traitement du langage naturel les transformer et leurs mecanisme d'attention se sont vue adapter au domaine de la vision par ordinateur pour donné naissance au Vision transformer. Le principe de base des Vision Transformers reste fondamentalement le même que celui des modèles Transformer appliqués au traitement du langage naturel. Tout d'abord, l'image est découpée en patchs, qui sont ensuite transformés en embeddings. Chaque patch se voit également attribuer une position dans l'image pour conserver les informations spatiales. Ensuite, ces embeddings sont soumis au Vision Transformer, qui applique des opérations d'attention pour capturer les relations entre les différents patchs, permettant ainsi au modèle de comprendre le contenu global de l'image. Cette approche permet aux Vision Transformers de rivaliser avec d'autres architectures de pointe en vision par ordinateur, offrant des performances de pointe dans des tâches telles que la classification d'images, la détection d'objets et la segmentation sémantique, leurs permettant également d'etre appliquer sur des série temporelle composé d'image tel que des données vidéos. Pour plus d’informations concernant les différents types de vision transformers disponibles à ce jour, il faut se référer à l’étude de P. Wang [69].

LLM :

Dans leur article [61], John A. Miller et ses collègues abordent un autre type de modèle dérivé des transformers, qui peut être utilisé pour le traitement des séries temporelles. Ces modèles sont appelé "modèle fondamentaux" et repose sur l'utilisation de LLM (Large Language Model) en tant que structure de base pour les architectures de deep learning.

D'après eux il existerait quatre approches pour créer des modèles fondamentaux applicable aux séries temporelles

1. Utiliser la puissance d'un modèle de langue large existant. Cela impliquerait de convertir des segments ou des patchs de séries temporelles en mots, d'utiliser ceux-ci pour générer les mots qui suivent, puis de les convertir à nouveau en séries temporelles (c'est-à-dire les prévisions). La base de ce fonctionnement serait l'existence de motifs universels dans les deux séquences (mots et segments de séries temporelles). Cependant, sans précaution, les séries temporelles converties en une séquence de mots risquent de produire des phrases sans signification. Il en va de même lorsque les mots de sortie sont convertis en prévisions de séries temporelles. Le réglage final d'un modèle de langue large à l'aide de données de séries temporelles peut améliorer leurs capacités de prévision.
   Parmi cette catégorie on retrouve des modèles telque : GPT4TS [77], LLM4TS [78] TEMPO [79], PromptCast [80]

2. Construire un modèle fondamental polyvalent pour les séries temporelles à partir de zéro en utilisant un grand nombre d'ensembles de données de séries temporelles. Cela nécessiterait un effort considérable pour collecter et prétraiter le grand volume de séries temporelles. Un calcul haute performance serait également nécessaire pour un entraînement approfondi. Bien que la formation exhaustive dans le domaine des séries temporelles soit généralement considérée comme moins exigeante que dans le domaine du langage.
   Catégorie dont font partie les modèles : TimeGPT [81] TimeCLR [82] PreDct [83], Lag-Llama [84], TimesNet [85]

3. Construire un modèle fondamental spécialisé pour les séries temporelles à partir de zéro. Cette alternative est plus gérable en termes de volume de données de formation nécessaire et des exigences en ressources de formation. De plus, on ne sait pas s'il existe une universalité exploitante à travers les domaines de séries temporelles. Un modèle fondamental entraîné sur des données de marché boursier serait-il utile pour la prédiction de pandémies ou d'autre domaine ?
   C'est le cas du modèle : AutoMixer [86]

4. Créer un modèle fondamental multimodal contenant des données textuelles et des séries temporelles. Toutes ces données devront être synchronisés en fonction des horodatages et en utilisant des techniques telles que l'alignement de séries temporelles par temps dynamique (DTW) pour l'alignement de séries temporelles.
   Les modèle concerné par cette catégorie sont : UniTime [87], Time-LLM [88]

John A. Miller et ses collègues fournissent un tableau synthétique de tous ces

modèles dans le tableau X.

Table 5. Foundation Models for Time Series

Model Type	Model	Short Description	Backbone	Date
2	TimeCLR [136, 137]	Contrastive Learning pre-training	Transformer	Jun 2022
2	TimesNet [129]	2D variation modeling	TimeBlock	Oct 2022
1	GPT4TS [153]	LLM with patch token input	GPT-2	May 2023
1	LLM4TS [14]	Temporal Encoding with LLM	GPT-2	Aug 2023
4	UniTime [71]	Input domain instructions + time series	GPT-2	Oct 2023
2	TimeGPT [26]	Added linear layer for forecasting	Transformer	Oct 2023
4	Time-LLM [44]	Input context via prompt prefix	LLaMA	Oct 2023
2	PreDct [20]	Patch-based Decoder-only	Transformer	Oct 2023
2	Lag-Llama [100]	Lag-based Decoder-only	LLaMA	Oct 2023
3	AutoMixer [90]	Adds AutoEncoder to TSMixer	TSMixer	Oct 2023
1	TEMPO [12]	Pre-trained transformer + statistical analysis	GPT-2	Oct 2023
1	PromptCast [134]	Text-like prompts for time series with LLM	GPT-3.5	Dec 2023

Tableau recap des modèle LLM disponible pour lanalyse de serie temporelles [61]

Pour plus de detail sur certaine technologie aborder dans cette article referez vous à l'article [60, 61].

Sources :

Num ref	titre	lien	bibtex
1	Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation	https://arxiv.org/abs/1406.1078	@article{cho2014learning, title={Learning phrase representations using RNN encoder-decoder for statistical machine translation}, author={Cho, Kyunghyun and Van Merri{\"e}nboer,

			Bart and Gulcehre, Caglar and Bahdanau, Dzmitry and Bougares, Fethi and Schwenk, Holger and Bengio, Yoshua}, journal={arXiv preprint arXiv:1406.1078}, year={2014} }
2	Sequence to Sequence Learning with Neural Networks	**https://arxiv.org/abs/1409.3215**	@article{sutskever2014sequence, title={Sequence to sequence learning with neural networks}, author={Sutskever, Ilya and Vinyals, Oriol and Le, Quoc V}, journal={Advances in neural information processing systems}, volume={27}, year={2014} }
3	Neural Machine Translation by Jointly Learning to Align and Translate	https://arxiv.org/abs/1409.0473	@article{bahdanau2014neural, title={Neural machine translation by jointly learning to align and translate},

			author={Bahdanau, Dzmitry and Cho, Kyunghyun and Bengio, Yoshua}, journal={arXiv preprint arXiv:1409.0473}, year={2014} }
4	HYBRID SPEECH RECOGNITION WITH DEEP BIDIRECTIONAL LSTM	https://www.cs.toronto.edu/~graves/asru_2013.pdf	@inproceedings{graves2013hybrid, title={Hybrid speech recognition with deep bidirectional LSTM}, author={Graves, Alex and Jaitly, Navdeep and Mohamed, Abdel-rahman}, booktitle={2013 IEEE workshop on automatic speech recognition and understanding}, pages={273--278}, year={2013}, organization={IEEE } }
5	Show, Attend and Tell: Neural Image Caption Generation with Visual Attention	https://proceedings.mlr.press/v37/xuc15.pdf	@inproceedings{xu2015show, title={Show, attend and tell: Neural image caption generation with visual

			attention}, author={Xu, Kelvin and Ba, Jimmy and Kiros, Ryan and Cho, Kyunghyun and Courville, Aaron and Salakhudinov, Ruslan and Zemel, Rich and Bengio, Yoshua}, booktitle={International conference on machine learning}, pages={2048--2057}, year={2015}, organization={PMLR} }
6	Deep Visual-Semantic Alignments for Generating Image Descriptions	https://cs.stanford.edu/people/karpathy/cvpr2015.pdf	@inproceedings{karpathy2015deep, title={Deep visual-semantic alignments for generating image descriptions}, author={Karpathy, Andrej and Fei-Fei, Li}, booktitle={Proceedings of the IEEE conference on computer vision and pattern recognition}, pages={3128--3137}, year={2015}

			}
7	Phrase-based Image Captioning	https://proceedings.mlr.press/v37/lebret15.pdf	@inproceedings{lebret2015phrase, title={Phrase-based image captioning}, author={Lebret, R{\'e}mi and Pinheiro, Pedro and Collobert, Ronan}, booktitle={International conference on machine learning}, pages={2085--2094}, year={2015}, organization={PMLR} }
8	**Long-term Recurrent Convolutional Networks for Visual Recognition and Description**	https://arxiv.org/abs/1411.4389	@inproceedings{donahue2015long, title={Long-term recurrent convolutional networks for visual recognition and description}, author={Donahue, Jeffrey and Anne Hendricks, Lisa and Guadarrama, Sergio and Rohrbach, Marcus and Venugopalan, Subhashini and Saenko, Kate and Darrell, Trevor},

			booktitle={Proceedings of the IEEE conference on computer vision and pattern recognition}, pages={2625--2634}, year={2015} }
9	Unsupervised Learning of Video Representations using LSTMs	https://proceedings.mlr.press/v37/srivastava15.html	@inproceedings{srivastava2015unsupervised, title={Unsupervised learning of video representations using lstms}, author={Srivastava, Nitish and Mansimov, Elman and Salakhudinov, Ruslan}, booktitle={International conference on machine learning}, pages={843--852}, year={2015}, organization={PMLR} }
10	**Convolutional LSTM Network: A Machine Learning Approach for**	https://arxiv.org/abs/1506.04214	@article{shi2015convolutional, title={Convolutional LSTM network: A

	Precipitation Nowcasting		machine learning approach for precipitation nowcasting},
			  author={Shi, Xingjian and Chen, Zhourong and Wang, Hao and Yeung, Dit-Yan and Wong, Wai-Kin and Woo, Wang-chun},
			
			journal={Advances in neural information processing systems},
			  volume={28},
			  year={2015}
			}

11	J. L. Elman . « Finding Structure in Time ». In : *Cogn. Sci.* 14 (1990), p. 179-
	211 (cf. p. 36).
		https://www.sciencedirect.com/science/article/abs/pii/0364021390090002E	@article{elman1990finding,
			  title={Finding structure in time},
			  author={Elman, Jeffrey L},
			
			journal={Cognitive science},
			  volume={14},
			  number={2},
			  pages={179--211},
			  year={1990},
			  publisher={Wiley Online Library}
			}

12	M. I. Jordan . « Serial Order : A Parallel Distributed Processing	https://www.sciencedirect.com/science/article/abs/pii/S0166411597801112	@incollection{jordan1997serial, title={Serial order: A parallel

	Approach ». In : *Advances in psychology* 121 (1997), p. 471-495 (cf. p. 36).		distributed processing approach}, author={Jordan, Michael I}, booktitle={Advances in psychology}, volume={121}, pages={471--495}, year={1997}, publisher={Elsevier} }
13	S. Hochreiter et J. Schmidhuber . « Long Short-Term Memory ». In : *Neural Computation* 9 (1997), p. 1735-1780 (cf. p. 36).	https://ieeexplore.ieee.org/abstract/document/6795963	@article{hochreiter1997long, title={Long short-term memory}, author={Hochreiter, Sepp and Schmidhuber, J{\"u}rgen}, journal={Neural computation}, volume={9}, number={8}, pages={1735--1780}, year={1997}, publisher={MIT press} }
14	F. A. Gers , J. Schmidhuber et F. Cummins . « Learning to Forget : Continual	https://ieeexplore.ieee.org/abstract/document/6789445	@article{gers2000learning, title={Learning to forget: Continual prediction with

	Prediction with LSTM ». In : *Neural Computation* 12 (2000), p. 2451-2471 (cf. p. 36).		LSTM}, author={Gers, Felix A and Schmidhuber, J{\"u}rgen and Cummins, Fred}, journal={Neural computation}, volume={12}, number={10}, pages={2451--2471}, year={2000}, publisher={MIT press} }
15	**Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**	https://arxiv.org/abs/1406.1078	@article{cho2014learning, title={Learning phrase representations using RNN encoder-decoder for statistical machine translation}, author={Cho, Kyunghyun and Van Merri{\"e}nboer, Bart and Gulcehre, Caglar and Bahdanau, Dzmitry and Bougares, Fethi and Schwenk, Holger and Bengio, Yoshua}, journal={arXiv preprint arXiv:1406.1078}, year={2014} }

16	**Minimal Gated Unit for Recurrent Neural Networks**	https://arxiv.org/abs/1603.09420	@article{zhou2016minimal, title={Minimal gated unit for recurrent neural networks}, author={Zhou, Guo-Bing and Wu, Jianxin and Zhang, Chen-Lin and Zhou, Zhi-Hua}, journal={International Journal of Automation and Computing}, volume={13}, number={3}, pages={226--234}, year={2016}, publisher={Springer} }
17	**Highway Speed Prediction Using Gated Recurrent Unit Neural Networks**	https://www.mdpi.com/2076-3417/11/7/3059	@article{jeong2021highway, title={Highway speed prediction using gated recurrent unit neural networks}, author={Jeong, Myeong-Hun and Lee, Tae-Young and Jeon, Seung-Bae and Youm, Minkyo}, journal={Applied Sciences}, volume={11}, number={7},

			pages={3059}, year={2021}, publisher={MDPI} }
18	Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network	https://arxiv.org/ftp/arxiv/papers/1703/1703.09752.pdf	@inproceedings{bontemps2016collective, title={Collective anomaly detection based on long short-term memory recurrent neural networks}, author={Bontemps, Lo{\"\i}c and Cao, Van Loi and McDermott, James and Le-Khac, Nhien-An}, booktitle={Future Data and Security Engineering: Third International Conference, FDSE 2016, Can Tho City, Vietnam, November 23-25, 2016, Proceedings 3}, pages={141--152}, year={2016}, organization={Springer} }
19	**Anomaly detection in ECG time**	https://ieeexplore.ieee.org/document/7344872	@inproceedings{chauhan2015anomaly, title={Anomaly

	signals via deep long short-term memory networks		detection in ECG time signals via deep long short-term memory networks}, author={Chauhan, Sucheta and Vig, Lovekesh}, booktitle={2015 IEEE international conference on data science and advanced analytics (DSAA)}, pages={1--7}, year={2015}, organization={IEEE} }
20	Towards End-to-End Speech Recognition with Recurrent Neural Networks	https://proceedings.mlr.press/v32/graves14.pdf	@inproceedings{graves2014towards, title={Towards end-to-end speech recognition with recurrent neural networks}, author={Graves, Alex and Jaitly, Navdeep}, booktitle={International conference on machine learning}, pages={1764--1772}, year={2014}, organization={PMLR} }

21	Bidirectional Long Short-Term Memory Networks for Relation Classification	https://aclanthology.org/Y15-1009.pdf	@inproceedings{zhang2015bidirectional, title={Bidirectional long short-term memory networks for relation classification}, author={Zhang, Shu and Zheng, Dequan and Hu, Xinchen and Yang, Ming}, booktitle={Proceedings of the 29th Pacific Asia conference on language, information and computation}, pages={73--78}, year={2015} }
22			@book{hawkins2007intelligence, title={On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines}, author={Hawkins, Jeff and Blakeslee, Sandra}, year={2007}, publisher={Macmill

			an} }
23	**Hierarchical Temporal Memory method for time-series-based anomaly detection**	https://www.sciencedirect.com/science/article/abs/pii/S0925231217313887	@article{wu2018hierarchical, title={Hierarchical temporal memory method for time-series-based anomaly detection}, author={Wu, Jia and Zeng, Weiru and Yan, Fei}, journal={Neurocomputing}, volume={273}, pages={535--546}, year={2018}, publisher={Elsevier} }
24	**Toward navigation ability for autonomous mobile robots with learning from demonstration paradigm: A view of hierarchical temporal memory**	https://journals.sagepub.com/doi/full/10.1177/1729881418777939	@article{zhang2018toward, title={Toward navigation ability for autonomous mobile robots with learning from demonstration paradigm: A view of hierarchical temporal memory}, author={Zhang, Xinzheng and Zhang, Jianfen and Zhong, Junpei}, journal={International Journal of

			Advanced Robotic Systems}, volume={15}, number={3}, pages={1729881418777939}, year={2018}, publisher={SAGE Publications Sage UK: London, England} }
25	A tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach	https://www.ai.rug.nl/minds/uploads/ESNTutorialRev.pdf	@article{jaeger2002tutorial, title={Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach}, author={Jaeger, Herbert}, year={2002}, publisher={GMD-Forschungszentrum Informationstechnik Bonn} }
26	**Temporal Convolutional Networks for Action Segmentation and Detection**	https://arxiv.org/abs/1611.05267	@inproceedings{lea2017temporal, title={Temporal convolutional networks for action segmentation and detection}, author={Lea, Colin and Flynn, Michael D and

			Vidal, Rene and Reiter, Austin and Hager, Gregory D}, booktitle={proceedings of the IEEE Conference on Computer Vision and Pattern Recognition}, pages={156--165}, year={2017} }
27	**Dynamic Routing Between Capsules**	https://arxiv.org/abs/1710.09829	@article{sabour2017dynamic, title={Dynamic routing between capsules}, author={Sabour, Sara and Frosst, Nicholas and Hinton, Geoffrey E}, journal={Advances in neural information processing systems}, volume={30}, year={2017} }
28	**Auto-Encoding Variational Bayes**	https://arxiv.org/abs/1312.6114	@article{kingma2013auto, title={Auto-encoding variational bayes}, author={Kingma, Diederik P and

			Welling, Max}, journal={arXiv preprint arXiv:1312.6114}, year={2013} }
30	**Multi-input CNN-GRU based human activity recognition using wearable sensors** Nidhi Dua, Shiva Nand Singh Vijay Bhaskar Semwal	https://link.springer.com/article/10.1007/s00607-021-00928-8	@article{dua2021multi, title={Multi-input CNN-GRU based human activity recognition using wearable sensors}, author={Dua, Nidhi and Singh, Shiva Nand and Semwal, Vijay Bhaskar}, journal={Computing}, volume={103}, number={7}, pages={1461--1478}, year={2021}, publisher={Springer} }
31	Sima Siami-Namini et al. A Comparison of ARIMA and LSTM in Forecasting Time Series	**https://par.nsf.gov/servlets/purl/10186768#:~:text=The%20study%20shows%20that%20LSTM,indicating%20the%20superiority%20of%20LSTM**.	

32	Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," arXiv preprint arXiv:1508.01991, 2015.		
33	E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, "Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks," in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pp. 2164–2168, IEEE, 2014.		
34	Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in Fifteenth Annual Conference of the International Speech Communication Association, 2014		
35	**Water Level Prediction Model**	https://ieeexplore.ieee.org/abstract/docu	

	Based on GRU and CNN Mingyang Pan; Hainan Zhou; Jiayi Cao; **Yisai Liu**; **Jiangling Hao**; Shaoxi Li; **Chi-Hua Chen**	ment/9044367	
36	DEEPRAIN: CONVLSTM NETWORK FOR PRECIPITATION PREDICTION USING MULTICHANNEL RADAR DATA Seongchan Kim1 , Seungkyun Hong1,2 , Minsu Joh1,3 , Sa-kwang Song	https://arxiv.org/pdf/1711.02316.pdf	
37	CNN-RNN Based Intelligent Recommendation for Online Medical Pre-Diagnosis Support Xiaokang Zhou; Yue Li; Wei Liang	https://ieeexplore.ieee.org/document/9093981	
38	**WaveNet: A Generative Model for Raw Audio** Aaron van den Oord	https://arxiv.org/abs/1609.03499	
39	LSTM, WaveNet, and 2D CNN for nonlinear time history prediction of seismic responses Chunxiao Ning, Yazhou Xie, Lijun Sun	https://www.sciencedirect.com/science/article/abs/pii/S0141029623004972	
40	Sales forecasting	https://arxiv.org/pdf	

	using WaveNet within the framework of the Kaggle competition Glib Kechyn, Lucius Yu, Yangguang Zang, Svyatoslav Kechyn	/1803.04037.pdf	
41	Dilated Convolutional Neural Networks for Time Series Forecasting Anastasia Borovykh	https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3272962	
42	Modeling Temporal Patterns with Dilated Convolutions for Time-Series Forecasting Yangfan Li_	https://dl.acm.org/doi/abs/10.1145/3453724	
43	Gradient-based learning applied to document recognition	https://ieeexplore.ieee.org/abstract/document/726791	@article{lecun1998gradient, title={Gradient-based learning applied to document recognition}, author={LeCun, Yann and Bottou, L{\'e}on and Bengio, Yoshua and Haffner, Patrick}, journal={Proceedings of the IEEE}, volume={86}, number={11}, pages={2278--2324}, year={1998}, publisher={Ieee}

			}
44	**An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling**	https://arxiv.org/abs/1803.01271	
45	Convolutional LSTM network: A machine learning approach for precipitation nowcasting	https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html	@article{shi2015convolutional, title={Convolutional LSTM network: A machine learning approach for precipitation nowcasting}, author={Shi, Xingjian and Chen, Zhourong and Wang, Hao and Yeung, Dit-Yan and Wong, Wai-Kin and Woo, Wang-chun}, journal={Advances in neural information processing systems}, volume={28}, year={2015} }
46	**A motion-aware**	https://link.springer.	@article{majd2019

	ConvLSTM network for action recognition	com/article/10.1007/s10489-018-1395-8	motion, title={A motion-aware ConvLSTM network for action recognition}, author={Majd, Mahshid and Safabakhsh, Reza}, journal={Applied Intelligence}, volume={49}, pages={2515--2521}, year={2019}, publisher={Springer} }
47	**A New Approach for Abnormal Human Activities Recognition Based on ConvLSTM Architecture**	https://www.mdpi.com/1424-8220/22/8/2946	@article{vrskova2022new, title={A new approach for abnormal human activities recognition based on ConvLSTM architecture}, author={Vrskova, Roberta and Hudec, Robert and Kamencay, Patrik and Sykora, Peter}, journal={Sensors}, volume={22}, number={8}, pages={2946}, year={2022}, publisher={MDPI} }

48	**Violent Behavioral Activity Classification Using Artificial Neural Network**	https://ieeexplore.ieee.org/abstract/document/9229532	@inproceedings{vrskova2020violent, title={Violent Behavioral Activity Classification Using Artificial Neural Network}, author={Vrskova, Roberta and Hudec, Robert and Sykora, Peter and Kamencay, Patrik and Benco, Miroslav}, booktitle={2020 New Trends in Signal Processing (NTSP)}, pages={1--5}, year={2020}, organization={IEEE} }
49	**A Temporal Fusion Approach for Video Classification with Convolutional and LSTM Neural Networks Applied to Violence Detection**	https://journal.iberamia.org/index.php/intartif/article/view/573	@article{de2021temporal, title={A temporal fusion approach for video classification with convolutional and LSTM neural networks applied to violence detection}, author={de Oliveira Lima, Jean Phelipe and Figueiredo, Carlos Maur{\'\i}cio Ser{\'o}dio}, journal={Inteligenci

			a Artificial}, volume={24}, number={67}, pages={40--50}, year={2021} }
50	**Mobile Neural Architecture Search Network and Convolutional Long Short-Term Memory-Based Deep Features Toward Detecting Violence from Video**	https://link.springer.com/article/10.1007/s13369-021-05589-5	@article{jahlan2021mobile, title={Mobile neural architecture search network and convolutional long short-term memory-based deep features toward detecting violence from video}, author={Jahlan, Heyam M Bin and Elrefaei, Lamiaa A}, journal={Arabian Journal for Science and Engineering}, volume={46}, number={9}, pages={8549--8563}, year={2021}, publisher={Springer} }
51	**Enhancing Anomaly Detection in Videos using a Combined YOLO and a VGG GRU Approach**	https://ieeexplore.ieee.org/abstract/document/10479307	@inproceedings{porier2023enhancing, title={Enhancing Anomaly Detection in Videos using a Combined YOLO and a VGG GRU

			Approach}, author={Poirier, Fabien and Jaziri, Rakia and Srour, Camille and Bernard, Gilles}, booktitle={2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)}, pages={1--6}, year={2023}, organization={IEEE} }
52	**3D Convolutional Neural Networks for Human Action Recognition**	https://ieeexplore.ieee.org/abstract/document/6165309	@article{ji20123d, title={3D convolutional neural networks for human action recognition}, author={Ji, Shuiwang and Xu, Wei and Yang, Ming and Yu, Kai}, journal={IEEE transactions on pattern analysis and machine intelligence}, volume={35}, number={1}, pages={221--231}, year={2012}, publisher={IEEE} }

53	**Human Activity Classification Using the 3DCNN Architecture**	https://www.mdpi.com/2076-3417/12/2/931	@article{vrskova2022human, title={Human activity classification using the 3DCNN architecture}, author={Vrskova, Roberta and Hudec, Robert and Kamencay, Patrik and Sykora, Peter}, journal={Applied Sciences}, volume={12}, number={2}, pages={931}, year={2022}, publisher={MDPI} }
54	Learning Spatiotemporal Features With 3D Convolutional Networks	https://openaccess.thecvf.com/content_iccv_2015/html/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.html	@inproceedings{tran2015learning, title={Learning spatiotemporal features with 3d convolutional networks}, author={Tran, Du and Bourdev, Lubomir and Fergus, Rob and Torresani, Lorenzo and Paluri, Manohar}, booktitle={Proceedings of the IEEE international conference on computer vision},

			pages={4489--4497}, year={2015} }
55	**Emotion Recognition from Multiband EEG Signals Using CapsNet** **Hao Chao *,Liang Dong,Yongli Liu** and**Baoyun Lu**	https://www.mdpi.com/1424-8220/19/9/2212	
56	**Use of a Capsule Network to Detect Fake Images and Videos** Huy H. Nguyen, **Junichi Yamagishi**, Isao Echizen	https://arxiv.org/abs/1910.12467	
57	**Abnormal Event Detection in Videos Using Spatiotemporal Autoencoder**	https://link.springer.com/chapter/10.1007/978-3-319-59081-3_23	@inproceedings{chong2017abnormal, title={Abnormal event detection in videos using spatiotemporal autoencoder}, author={Chong, Yong Shean and

			Tay, Yong Haur}, booktitle={Advances in Neural Networks-ISNN 2017: 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21--26, 2017, Proceedings, Part II 14}, pages={189--196}, year={2017}, organization={Springer} }
58	**CNN and LSTM based Encoder-Decoder for Anomaly Detection in Multivariate Time Series**	https://ieeexplore.ieee.org/abstract/document/9587207	@article{malhotra2016lstm, title={LSTM-based encoder-decoder for multi-sensor anomaly detection}, author={Malhotra, Pankaj and Ramakrishnan, Anusha and Anand, Gaurangi and Vig, Lovekesh and Agarwal, Puneet and Shroff, Gautam}, journal={arXiv preprint arXiv:1607.00148}, year={2016}

			}
59	Time series forecasting using a deep belief network with restricted Boltzmann machines	https://www.sciencedirect.com/science/article/abs/pii/S0925231213007388	@article{kuremoto2014time, title={Time series forecasting using a deep belief network with restricted Boltzmann machines}, author={Kuremoto, Takashi and Kimura, Shinsuke and Kobayashi, Kunikazu and Obayashi, Masanao}, journal={Neurocomputing}, volume={137}, pages={47--56}, year={2014}, publisher={Elsevier} }
60	**Generative adversarial networks in time series: A survey and taxonomy**	https://arxiv.org/abs/2107.11098	@article{brophy2021generative, title={Generative adversarial networks in time series: A survey and taxonomy}, author={Brophy, Eoin and Wang, Zhengwei and She, Qi and Ward, Tomas}, journal={arXiv preprint

			arXiv:2107.11098}, year={2021} }
61	**A Survey of Deep Learning and Foundation Models for Time Series Forecasting**	https://arxiv.org/pdf/2401.13912	@article{miller2024survey, title={A survey of deep learning and foundation models for time series forecasting}, author={Miller, John A and Aldosari, Mohammed and Saeed, Farah and Barna, Nasid Habib and Rana, Subas and Arpinar, I Budak and Liu, Ninghao}, journal={arXiv preprint arXiv:2401.13912}, year={2024} }
62	**Deep Learning for Time Series Anomaly Detection: A Survey**	https://arxiv.org/pdf/2211.05244	@article{darban2022deep, title={Deep learning for time series anomaly detection: A survey}, author={Darban, Zahra Zamanzadeh and Webb, Geoffrey I and Pan, Shirui and Aggarwal, Charu C and Salehi, Mahsa}, journal={arXiv

			preprint arXiv:2211.05244}, year={2022} }
63	**A Comprehensive Survey on Graph Neural Networks**	https://arxiv.org/abs/1901.00596	@article{wu2020comprehensive, title={A comprehensive survey on graph neural networks}, author={Wu, Zonghan and Pan, Shirui and Chen, Fengwen and Long, Guodong and Zhang, Chengqi and Philip, S Yu}, journal={IEEE transactions on neural networks and learning systems}, volume={32}, number={1}, pages={4--24}, year={2020}, publisher={IEEE} }
64	**A review on extreme learning machine**	https://link.springer.com/article/10.1007/s11042-021-11007-7	@article{wang2022review, title={A review on extreme learning machine}, author={Wang, Jian and Lu, Siyuan and Wang, Shui-Hua and Zhang, Yu-Dong}, journal={Multimedia Tools and

			Applications}, volume={81}, number={29}, pages={41611--41660}, year={2022}, publisher={Springer} }
65	**On-line sequential extreme learning machine**.	https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2ebdfa3852e4ad68a0cfde9f0f69b95953d69178	@article{huang2005line, title={On-line sequential extreme learning machine.}, author={Huang, Guang-Bin and Liang, Nan-Ying and Rong, Hai-Jun and Saratchandran, Paramasivan and Sundararajan, Narasimhan}, journal={Computational Intelligence}, volume={2005}, pages={232--237}, year={2005}, publisher={Citeseer} }
66	Online sequential extreme learning machine with forgetting mechanism	https://www.sciencedirect.com/science/article/abs/pii/S0925231212000690	@article{zhao2012online, title={Online sequential extreme learning machine with forgetting

			mechanism},
			  author={Zhao, Jianwei and Wang, Zhihui and Park, Dong Sun},
			
			journal={Neurocomputing},
			  volume={87},
			  pages={79--89},
			  year={2012},
			
			publisher={Elsevier}
			}

67	Weighted extreme learning machine for imbalance learning	https://www.sciencedirect.com/science/article/abs/pii/S0925231212006479	@article{zong2013weighted,
			  title={Weighted extreme learning machine for imbalance learning},
			  author={Zong, Weiwei and Huang, Guang-Bin and Chen, Yiqiang},
			
			journal={Neurocomputing},
			  volume={101},
			  pages={229--242},
			  year={2013},
			
			publisher={Elsevier}
			}

68	Leave No Context Behind: Efficient Infinite Context	https://arxiv.org/pdf/2404.07143.pdf	@article{munkhdalai2024leave, title={Leave No

	Transformers with Infini-attention		Context Behind: Efficient Infinite Context Transformers with Infini-attention}, author={Munkhdalai, Tsendsuren and Faruqui, Manaal and Gopal, Siddharth}, journal={arXiv preprint arXiv:2404.07143}, year={2024} }
69		https://github.com/lucidrains/vit-pytorch	
70	Sima Siami-Namini et al. The Performance of LSTM and BiLSTM in Forecasting Time Series	https://par.nsf.gov/servlets/purl/10186554	
71	Framewise phoneme classification with bidirectional LSTM and other neural network architectures	https://www.sciencedirect.com/science/article/abs/pii/S0893608005001206	@article{graves2005framewise, title={Framewise phoneme classification with bidirectional LSTM and other neural network architectures}, author={Graves, Alex and Schmidhuber, J{\"u}rgen},

			journal={Neural networks}, volume={18}, number={5-6}, pages={602--610}, year={2005}, publisher={Elsevier} }
72	**Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition**	https://link.springer.com/chapter/10.1007/11550907_126	@inproceedings{graves2005bidirectional, title={Bidirectional LSTM networks for improved phoneme classification and recognition}, author={Graves, Alex and Fern{\'a}ndez, Santiago and Schmidhuber, J{\"u}rgen}, booktitle={International conference on artificial neural networks}, pages={799--804}, year={2005}, organization={Springer} }
73	Temporal	https://www.sciencedirect.com/science/a	@article{thill2021temporal,

	convolutional autoencoder for unsupervised anomaly detection in time series	rticle/pii/S1568494621006724?casa_token=SA5XhF8sncoAAAAA:0s7tLOswrFSAW_8cedrbmFq9XprsfJrdC2kBWov8p9XvJsQ4uJg7O0uE6itvG8G1KkDkk_BlRkMQ	title={Temporal convolutional autoencoder for unsupervised anomaly detection in time series}, author={Thill, Markus and Konen, Wolfgang and Wang, Hao and B{\"a}ck, Thomas}, journal={Applied Soft Computing}, volume={112}, pages={107751}, year={2021}, publisher={Elsevier} }
74	A comparative study of HTM and other neural network models for online sequence learning with streaming data	https://scholar.google.com/scholar?hl=fr&as_sdt=0%2C5&q=A+comparative+study+of+HTM+and+other+neural+network+models+for+online+sequence+learning+with+streaming+data&btnG=#d=gs_cit&t=1715079089080&u=%2Fscholar%3Fq%3Dinfo%3AjAI0rkyYo4YJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Dfr	@inproceedings{cui2016comparative, title={A comparative study of HTM and other neural network models for online sequence learning with streaming data}, author={Cui, Yuwei and Surpur, Chetan and Ahmad, Subutai and Hawkins, Jeff}, booktitle={2016 International joint conference on neural networks (IJCNN)}, pages={1530--1538},

			year={2016}, organization={IEEE } }
75	Using LSTM and GRU neural network methods for traffic flow prediction	https://www.researchgate.net/profile/Li-Li-86/publication/312402649_Using_LSTM_and_GRU_neural_network_methods_for_traffic_flow_prediction/links/5c20d38d299bf12be3971696/Using-LSTM-and-GRU-neural-network-methods-for-traffic-flow-prediction.pdf	@inproceedings{fu2016using, title={Using LSTM and GRU neural network methods for traffic flow prediction}, author={Fu, Rui and Zhang, Zuo and Li, Li}, booktitle={2016 31st Youth academic annual conference of Chinese association of automation (YAC)}, pages={324--328}, year={2016}, organization={IEEE } }
76	**Generative adversarial networks in time series: A survey and taxonomy**	https://arxiv.org/abs/2107.11098	@article{brophy2021generative, title={Generative adversarial networks in time series: A survey and taxonomy}, author={Brophy, Eoin and Wang, Zhengwei and She, Qi and Ward,

			Tomas}, journal={arXiv preprint arXiv:2107.11098}, year={2021} }
77	One Fits All: Power General Time Series Analysis by Pretrained LM	https://proceedings.neurips.cc/paper_files/paper/2023/hash/86c17de05579cde52025f9984e6e2ebb-Abstract-Conference.html	@article{zhou2024one, title={One fits all: Power general time series analysis by pretrained lm}, author={Zhou, Tian and Niu, Peisong and Sun, Liang and Jin, Rong and others}, journal={Advances in neural information processing systems}, volume={36}, year={2024} }
78	LLM4TS: Aligning Pre-Trained LLMs as Data-Efficient Time-Series Forecasters	https://arxiv.org/abs/2308.08469	@article{chang2023llm4ts, title={Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms}, author={Chang, Ching and Peng, Wen-Chih and Chen, Tien-Fu}, journal={arXiv preprint

			arXiv:2308.08469}, year={2023} }
79	TEMPO: Prompt-based Generative Pre-trained Transformer for Time Series Forecasting	https://arxiv.org/abs/2310.04948	@article{cao2023tempo, title={Tempo: Prompt-based generative pre-trained transformer for time series forecasting}, author={Cao, Defu and Jia, Furong and Arik, Sercan O and Pfister, Tomas and Zheng, Yixiang and Ye, Wen and Liu, Yan}, journal={arXiv preprint arXiv:2310.04948}, year={2023} }
80	PromptCast: A New Prompt-Based Learning Paradigm for Time Series Forecasting	https://ieeexplore.ieee.org/abstract/document/10356715	@article{xue2023promptcast, title={Promptcast: A new prompt-based learning paradigm for time series forecasting}, author={Xue, Hao and Salim, Flora D}, journal={IEEE Transactions on Knowledge and Data Engineering}, year={2023}, publisher={IEEE}

			}
81	TimeGPT-1	https://arxiv.org/abs/2310.03589	@article{garza2023timegpt, title={TimeGPT-1}, author={Garza, Azul and Mergenthaler-Canseco, Max}, journal={arXiv preprint arXiv:2310.03589}, year={2023} }
82	**Toward a Foundation Model for Time Series Data**	https://dl.acm.org/doi/abs/10.1145/3583780.3615155	@inproceedings{yeh2023toward, title={Toward a foundation model for time series data}, author={Yeh, Chin-Chia Michael and Dai, Xin and Chen, Huiyuan and Zheng, Yan and Fan, Yujie and Der, Audrey and Lai, Vivian and Zhuang, Zhongfang and Wang, Junpeng and Wang, Liang and others}, booktitle={Proceedings of the 32nd ACM International Conference on Information and Knowledge Management},

			pages={4400--4404}, year={2023} }
83	A decoder-only foundation model for time-series forecasting	https://arxiv.org/abs/2310.10688	@article{das2023decoder, title={A decoder-only foundation model for time-series forecasting}, author={Das, Abhimanyu and Kong, Weihao and Sen, Rajat and Zhou, Yichen}, journal={arXiv preprint arXiv:2310.10688}, year={2023} }
84	Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting	https://arxiv.org/abs/2310.08278	@article{rasul2023lag, title={Lag-llama: Towards foundation models for time series forecasting}, author={Rasul, Kashif and Ashok, Arjun and Williams, Andrew Robert and Khorasani, Arian and Adamopoulos, George and Bhagwatkar, Rishika and Bilo{\v{s}}, Marin and Ghonia, Hena and Hassen, Nadhir Vincent and Schneider,

			Anderson and others}, journal={arXiv preprint arXiv:2310.08278}, year={2023} }
85	TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis	https://openreview.net/forum?id=ju_Uqw384Oq	@inproceedings{wu2022timesnet, title={Timesnet: Temporal 2d-variation modeling for general time series analysis}, author={Wu, Haixu and Hu, Tengge and Liu, Yong and Zhou, Hang and Wang, Jianmin and Long, Mingsheng}, booktitle={The eleventh international conference on learning representations}, year={2022} }
86	AutoMixer for Improved Multivariate Time-Series Forecasting on Business and IT Observability Data	https://arxiv.org/abs/2310.20280	@article{palaskar2023automixer, title={AutoMixer for Improved Multivariate Time-Series Forecasting on BizITOps Data}, author={Palaskar, Santosh and Ekambaram, Vijay and Jati, Arindam

			and Gantayat, Neelamadhav and Saha, Avirup and Nagar, Seema and Nguyen, Nam H and Dayama, Pankaj and Sindhgatta, Renuka and Mohapatra, Prateeti and others}, journal={arXiv preprint arXiv:2310.20280}, year={2023} }
87	UniTime: A Language-Empowered Unified Model for Cross-Domain Time Series Forecasting	https://arxiv.org/abs/2310.09751	@article{liu2023unitime, title={Unitime: A language-empowered unified model for cross-domain time series forecasting}, author={Liu, Xu and Hu, Junfeng and Li, Yuan and Diao, Shizhe and Liang, Yuxuan and Hooi, Bryan and Zimmermann, Roger}, journal={arXiv preprint arXiv:2310.09751}, year={2023} }
88	Time-LLM: Time Series Forecasting by Reprogramming	https://arxiv.org/abs/2310.01728	@article{jin2023time, title={Time-llm: Time series

	Large Language Models		forecasting by reprogramming large language models},
			  author={Jin, Ming and Wang, Shiyu and Ma, Lintao and Chu, Zhixuan and Zhang, James Y and Shi, Xiaoming and Chen, Pin-Yu and Liang, Yuxuan and Li, Yuan-Fang and Pan, Shirui and others},
			  journal={arXiv preprint arXiv:2310.01728},
			  year={2023}
			}

Info en +:

Data Augmentation techniques in time series domain: a survey and taxonomy

https://link.springer.com/article/10.1007/s00521-023-08459-3

Info article 75 :

GRU > LSTM / GRU plus petit et plus rapide à entrainer

Info article MGU :

MGU == GRU / Plus petit et plus rapide à entrainer

Info provenant de l'article 61 :

L'efficacité des LLMs est discutée dans [120]. Plusieurs modèles/architectures de base ont été envisagés pour les modèles fondamentaux dans la classification de séries temporelles [137]. Cet article a comparé les architectures LSTM, ResNet, GRU et

Transformers, laissant entrevoir le plus grand potentiel avec l'architecture Transformer. Pour les séries temporelles multivariées, les modèles avec une attention ciblée ou clairsemée ont montré une plus grande précision (erreurs plus petites) que les transformers utilisant une attention complète.