

Titre de la thèse (sur plusieurs lignes si nécessaire)

Traduction du titre de la thèse (sur plusieurs lignes si nécessaire)

Thèse de doctorat de l'université Paris-Saclay et de l'université XXX (si cotutelle - sinon enlever cette seconde partie)

École doctorale n° d'accréditation, dénomination et sigle
Spécialité de doctorat : voir annexe
Graduate School : voir annexe. Référent : voir annexe

Thèse préparée dans la (ou les) unité(s) de recherche **Nom(s)** (voir annexe), sous la direction de **Prénom NOM**, titre du directeur ou de la directrice de thèse, la co-direction de **Prénom NOM**, titre du co-directeur ou de la co-directrice de thèse, le co-encadrement de **Prénom NOM**, titre, du co-encadrant ou de la co-encadrante ou la co-supervision de **Prénom NOM**, titre, du tuteur ou de la tutrice (en cas de partenariat industriel)

Thèse soutenue à Paris-Saclay, le JJ mois AAAA, par

Prénom NOM

Composition du jury

Membres du jury avec voix délibérative

Prénom NOM

Titre, Affiliation

Président ou Présidente

Rapporteur & Examinateur / trice

Rapporteur & Examinateur / trice

Examinateur ou Examinatrice

Examinateur ou Examinatrice

Titre : titre (en français).....

Mots clés : 3 à 6 mots clefs (version en français)

Résumé : Lorem ipsum dolor sit amet, consec-tetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, no-nummy eget, consectetuer id, vulputate a, ma-gna. Donec vehicula augue eu neque. Pellentes-que habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasel-lus eu tellus sit amet tortor gravida placerat. In-teger sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bi-bendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Cu-rabitur auctor semper nulla. Donec varius orci

eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non ju-sto. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum so-ciis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tinci-dunt urna. Nulla ullamcorper vestibulum tur-pis. Pellentesque cursus luctus mauris.

Title : titre (en anglais).....

Keywords : 3 à 6 mots clefs (version en anglais)

Abstract : Lorem ipsum dolor sit amet, consec-tetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, no-nummy eget, consectetuer id, vulputate a, ma-gna. Donec vehicula augue eu neque. Pellentes-que habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasel-lus eu tellus sit amet tortor gravida placerat. In-teger sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bi-bendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Cu-rabitur auctor semper nulla. Donec varius orci

eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non ju-sto. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum so-ciis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tinci-dunt urna. Nulla ullamcorper vestibulum tur-pis. Pellentesque cursus luctus mauris.

Table des matières

1	Introduction	9
1.1	Contexte historique et technique de la guerre électronique	9
1.1.1	Genèse d'une course technologique	9
1.1.2	La dynamique fréquentielle (Années 1950)	9
1.1.3	La cohérence, le Doppler et l'analyse fine (Années 1960-1970)	10
1.1.4	Complexité moderne et rôle de l'ELINT	10
1.2	Du signal à l'information : le rôle du capteur ESM et du renseignement	10
1.2.1	De l'onde à la donnée : intercepter et trier	10
1.2.2	De l'alerte tactique à l'exploitation stratégique	11
1.3	Le besoin de simulation et la problématique d'accélération	11
1.3.1	L'impératif de la simulation numérique	11
1.3.2	Problématique : le goulot d'étranglement temporel	12
1.4	Organisation du manuscrit	12
2	Problématique	17
2.1	Introduction	17
2.2	Description de l'intégration du capteur de Mesures de Soutien Électronique et son fonctionnement	17
2.2.1	Contexte opérationnel : La maîtrise du spectre électromagnétique	17
2.2.2	Chaîne de traitement de l'information	17
2.2.3	Architecture et fonctionnement du capteur	18
2.3	Description de l'environnement virtuel	22
2.3.1	Motivations	22
2.3.2	Architecture et fonctionnement de l'environnement virtuel	23
2.4	Identification du goulot d'étranglement et limites opérationnelles	24
2.4.1	L'impératif de temps réel et le coût de la simulation	24
2.4.2	Analyse de la complexité et localisation du verrou	25
2.5	Formalisation et complexité du problème d'apprentissage	25
2.5.1	Nature des données et définition formelle de la tâche	25
2.5.2	Une double problématique : Traitement de séquence et Génération	26
2.6	Introduction	29
2.7	IA générative	29
2.8	Méthodes pour le traitement de séquence	29
2.9	Les améliorations	30
3	État de l'art	31
3.1	Introduction : Environnements Numériques et typologie des apports de l'IA	31
3.1.1	Cadre Conceptuel : Environnement Virtuel et Jumeau Numérique	31

3.1.2	Principes mathématiques de l'Apprentissage Profond	33
3.1.3	L'IA pour la constitution géométrique et visuelle de l'environnement	36
3.1.4	L'IA pour l'accélération et la modélisation des phénomènes physiques	37
3.1.5	L'IA au service de l'interactivité et de l'adaptation décisionnelle	38
3.1.6	Ancrage dans la problématique	39
3.2	IA générative	39
3.2.1	L'approche probabiliste explicite : Les VAE (2013)	40
3.2.2	La révolution antagoniste : Les GAN (2014)	40
3.2.3	La génération par raffinement : Les Modèles de Diffusion (2020)	41
3.2.4	Le paradigme séquentiel et l'Autorégression	41
3.2.5	Ancrage dans la problématique	42
3.3	Méthodes de traitement : Séquences et structures spatiales	42
3.3.1	Typologie des données : De la causalité temporelle à la topologie spatiale	42
3.3.2	Réseaux de convolution	43
3.3.3	Réseaux de neurones récurrents et Espaces d'Etats (RNN et SSM)	48
3.3.4	L'architecture Transformer	52
3.3.5	Ancrage dans la problématique	61
4	Capacité de discernement des modèles	63
4.1	Génération des données et protocole de construction	63
4.1.1	Règle d'ordonnancement implicite	64
4.1.2	Définition de la difficulté et invariances	64
4.1.3	Algorithme de génération sous contrainte	66
4.2	Stratégie d'apprentissage et architectures neuronales	66
4.2.1	Fonction de coût normalisée	66
4.2.2	Métrique d'évaluation : La Justesse Positionnelle	67
4.2.3	Apprentissage progressif	68
4.2.4	Architectures évaluées	68
4.3	Résultats expérimentaux et analyse	69
4.3.1	Protocole de visualisation	69
4.3.2	Évaluation du modèle de référence	70
4.3.3	Évaluation de la méthodologie proposée	71
4.3.4	Synthèse et comparaison des architectures	72
5	Le cœur de ma thèse ici	75
6	Dépliage du problème : une approche événementielle	77
6.1	Motivation : Du traitement de séquence à la modélisation dynamique	77
6.1.1	Le goulot d'étranglement des architectures Seq2Seq	77
6.1.2	Le DCI comme système dynamique réactif	78
6.2	Formalisation du problème "Flux-à-flux"	78
6.2.1	Contraintes architecturales et choix de conception	78
6.3	Protocole de transformation des données	79

6.3.1	Illustration du mécanisme	79
6.4	Validation de l'approche sur architectures récurrentes	80
6.4.1	Les architectures	80
6.4.2	Résultats expérimentaux	81
6.4.3	Conclusion	82
6.5	Extension des essais : Généralisation à d'autres architectures	82
6.5.1	Préparation des données et ingénierie des caractéristiques	82
6.5.2	Sélection des architectures	84
6.5.3	Fonction de perte et métriques d'évaluation	86
6.5.4	Résultats et Analyse Comparative	87
6.5.5	Conclusion sur les extensions	89

Introduction (plan)

Le chapitre introduction comprendra les éléments suivants :

- Introduction du domaine de la guerre électronique
- Les grandes lignes du rôle du capteur ESM
- Explications de l'utilité de l'environnement numérique
- La problématique d'accélération
- Introduction succincte sur le domaine de l'IA
- Le plan de notre approche

1 - Introduction

1.1 . Contexte historique et technique de la guerre électronique

L'histoire de la guerre moderne est indissociable de la maîtrise du spectre électromagnétique. Dès lors que le radar est devenu le capteur principal des armées, assurant la surveillance aérienne et le guidage des armes, il est devenu simultanément une cible prioritaire. En émettant pour détecter, ces systèmes révèlent leur position et s'exposent à des contre-mesures. Cette vulnérabilité intrinsèque a donné naissance à la Guerre Électronique (GE), définie comme l'ensemble des actions visant à exploiter, réduire ou empêcher l'utilisation hostile du spectre électromagnétique tout en protégeant son utilisation par son propre camp.

La synthèse historique et technique qui suit, décrivant l'évolution conjointe des radars et des systèmes de brouillage, est adaptée de l'ouvrage *Fundamentals of electronic warfare* [1].

1.1.1 . Genèse d'une course technologique

L'émergence de la GE répond à une "dialectique du combat" entre les systèmes offensifs et défensifs.

Les premières générations de radars, tels que les systèmes d'alerte ou de conduite de tir de la Seconde Guerre Mondiale (par exemple le Würzburg allemand), se caractérisaient par leur fréquence porteuse et leur période de répétition fixes. Les premières missions d'écoute aéroportées, connues sous le nom de vols Ferret et posant les fondations du Renseignement Électronique (Electronic Intelligence - ELINT), interceptèrent les signaux adverses pour en extraire la longueur d'onde et la cadence. Ce renseignement permit de concevoir des contre-mesures adaptées : lors du raid sur Hambourg en juillet 1943, le largage massif de bandelettes métalliques, taillées exactement à la demi-longueur d'onde des radars allemands, satura mécaniquement les écrans de défense. Ce brouillage passif doubla le taux de survivabilité des bombardiers alliés pendant de nombreux mois, forçant le radar à évoluer en retour.

1.1.2 . La dynamique fréquentielle (Années 1950)

Pour s'extraire de ce brouillage rudimentaire ciblé sur une fréquence connue, les radars ont été munis d'émetteurs pouvant modifier la fréquence de leur porteuse. Sur le plan offensif, le brouillage de barrage est apparu pour inonder une large bande spectrale, afin de couvrir toutes les fréquences de saut possibles. Sur le plan du renseignement, cette agilité a marqué la fin de l'écoute sur fréquence fixe : les intercepteurs ont dû évoluer vers des récepteurs à large bande instantanée, introduisant la notion de mesure de soutien électronique (Electronic Support Measures - ESM).

1.1.3 . La cohérence, le Doppler et l'analyse fine (Années 1960-1970)

L'avènement des radars cohérents a marqué un tournant technologique. En réalisant un filtrage Doppler, le radar est devenu capable d'annuler les échos fixes, rendant les nuages de bandelettes obsolètes. La GE a alors basculé dans l'ère du traitement fin du signal. Pour le brouillage, il a fallu développer des techniques de déception capables d'intercepter le signal, de le mettre en mémoire et de le réémettre avec des altérations précises de phase ou de retard pour créer de fausses cinématiques (leurrage en distance ou en vitesse). Pour la fonction ESM, la mission n'était plus simplement de mesurer l'enveloppe temporelle d'une impulsion mais aussi d'extraire et caractériser la modulation intra-impulsion (phase, fréquence) pour identifier la signature électromagnétique du radar adverse.

1.1.4 . Complexité moderne et rôle de l'ELINT

La phase actuelle se caractérise par l'utilisation d'antennes à balayage électronique et de formes d'onde discrètes, rendant les signaux difficiles à distinguer du bruit de fond. Dans ce contexte, la mission de Renseignement Électronique est double : détecter les émissions adverses pour permettre leur neutralisation, et analyser l'environnement de brouillage pour optimiser la protection de ses propres systèmes.

1.2 . Du signal à l'information : le rôle du capteur ESM et du renseignement

Si l'évolution des architectures radar a façonné la complexité de l'environnement électromagnétique, c'est au capteur de Mesures de Soutien Électronique (ESM) qu'incombe la tâche de le décrypter. Contrairement au radar qui est un système actif, l'ESM opère de manière strictement passive pour écouter le spectre sans révéler la position du porteur. Son objectif est d'intercepter les émissions adverses pour transformer un flux analogique en données numériques structurées, ouvrant la voie à une exploitation tactique dans l'immédiat, puis une exploitation stratégique à plus long terme.

1.2.1 . De l'onde à la donnée : intercepter et trier

Dans la réalité opérationnelle, une antenne de réception ne capte pas des signaux radar proprement isolés. Elle baigne en permanence dans un champ électromagnétique, où se superposent le bruit de fond naturel, les communications et les émissions simultanées de différents radars. Le premier rôle de la chaîne ESM est donc de scruter ce flux analogique pour extraire du bruit les brèves variations d'énergie correspondant à des impulsions incidentes.

Lorsqu'une impulsion est ainsi détectée, elle est caractérisée numériquement sous la forme d'un vecteur compact appelé PDW (Pulse Descriptor Word). Un PDW résume la signature physique de l'impulsion à travers quelques paramètres clés :

- son temps d'arrivée (Time of arrival - ToA)
- sa durée (Longueur d'impulsion - LI)

- sa fréquence (Fréquence - Freq)
- son amplitude (Level - Lvl)
- sa direction d'arrivée (Direction of Arrival - DoA)

Le flux continu de PDW mélangés est transmis à une unité de traitement chargée du désentrelacement. Cet algorithme cherche des corrélations fréquentielles dans le flux de PDW sur de courtes fenêtres temporelles, il sépare et regroupe les impulsions appartenant à un même radar pour former des regroupements élémentaires appelés *plots*. Le flux de plots ainsi généré est transmis à une deuxième unité de traitement chargée du pistage. Cette fois à long terme mais sur un même principe de corrélation, les plots sont regroupés en différentes *pistes* qui décrivent le comportement global de chaque émetteur (comme sa cadence d'observation ou le type de balayage de son antenne).

1.2.2 . De l'alerte tactique à l'exploitation stratégique

Une fois ces pistes reconstruites, la donnée est exploitée selon deux échelles de temps distinctes.

En vol, le système opère à un niveau purement tactique. Le capteur ESM compare en temps réel les paramètres des pistes avec une base de données embarquée, communément appelée bibliothèque de menaces, pour identifier la classe du radar adverse. L'objectif ici est d'alerter l'équipage d'un accrochage potentiel et de déclencher automatiquement les contre-mesures adéquates, qu'il s'agisse de manœuvres évasives ou d'actions de brouillage.

Les données ESM les plus complexes ou celles correspondant à des émissions inconnues de la bibliothèque sont enregistrées pour être analysées au sol. Des algorithmes lourds et des analystes spécialisés dépouillent ces enregistrements pour découvrir de nouveaux modes opératoires ou extraire des signatures matérielles uniques permettant d'identifier un émetteur spécifique. C'est cette boucle de renseignement différé qui permet de mettre à jour l'Ordre de Bataille Électronique et de reprogrammer les bibliothèques de toute la flotte, garantissant l'adaptation continue des forces face à l'évolution de la menace.

1.3 . Le besoin de simulation et la problématique d'accélération

La chaîne de traitement de la Guerre Électronique, depuis la détection de l'impulsion par le capteur ESM jusqu'à l'identification de la menace, s'appuie sur une succession de différents algorithmes : désentrelacement, pistage, classification. Garantir la fiabilité opérationnelle de ces algorithmes exige de les tester intensivement lors des phases de développement et de validation.

1.3.1 . L'impératif de la simulation numérique

Idéalement, ces algorithmes devraient être évalués sur des données réelles issues d'essais en vol. Néanmoins, les campagnes de vol se heurtent à des contraintes logistiques et financières majeures, tout en n'offrant qu'une représentativité limitée face à l'infinité des scénarios

tactiques possibles. De plus, lors d'un vol réel, la "vérité terrain" absolue, c'est-à-dire la position et l'activité exactes de tous les émetteurs de la zone, n'est jamais parfaitement connue. Cette incertitude empêche d'évaluer objectivement si les algorithmes ont fonctionné comme voulu.

Pour s'affranchir de ces limites, le recours à la simulation est souvent la solution privilégiée, et c'est celle employée dans notre cas d'étude. Grâce à un environnement virtuel auquel on fournit un scénario tactique décrivant parfaitement la vérité terrain (la cinématique et les modes de chaque radar), l'ensemble de la chaîne de l'information est modélisé : l'émission des impulsions par les radars adverses, les effets de propagation, l'impact des antennes, jusqu'au processus interne de détection du capteur ESM. À l'issue de cette simulation, un flux de PDW est généré, tel qu'il aurait été produit par un véritable équipement matériel. L'avantage de cette approche réside dans la maîtrise absolue de la vérité terrain. Elle permet de boucler le cycle de validation en confrontant directement au scénario initial les résultats reconstruits par les algorithmes situés en aval de la génération des PDW.

1.3.2 . Problématique : le goulot d'étranglement temporel

Bien que cet environnement virtuel offre une capacité de génération de données maîtrisée et en grande quantité, sa mécanique de modélisation s'avère particulièrement lente. Cette lenteur devient problématique lors de la simulation de scénarios complexes impliquant un grand nombre de radars. De surcroît, elle interdit toute exécution en temps réel, un cas d'usage pourtant indispensable pour la formation des opérateurs sur des simulateurs interactifs, où l'humain doit pouvoir modifier la situation tactique en direct, par exemple en initiant des manœuvres d'évitement.

Nos travaux de recherche s'inscrivent précisément dans ce contexte. L'objectif de cette thèse est d'identifier le goulot d'étranglement de l'environnement virtuel et d'étudier comment l'intelligence artificielle peut s'y substituer pour accélérer la génération des données en conservant la même représentativité. Le chapitre suivant s'attachera à détailler l'architecture interne de cet environnement virtuel afin de formaliser techniquement cette problématique.

1.4 . Organisation du manuscrit

Pour répondre à cette problématique d'accélération par l'intelligence artificielle, nos travaux s'articulent autour d'une analyse progressive, allant de la compréhension physique du système jusqu'à l'élaboration d'architectures neuronales avancées. Ce manuscrit est ainsi structuré en cinq chapitres principaux :

Le Chapitre 2 pose le cadre formel et technique de notre problématique. Il détaille le principe de fonctionnement du capteur ESM réel et de l'environnement virtuel permettant de simuler son comportement. L'isolement précis du le goulot d'étranglement de l'environnement virtuel nous permet de traduire le besoin d'accélération en un problème mathématique d'apprentissage automatique, d'en exposer les difficultés intrinsèques, et de définir les axes de recherche qui guideront nos contributions.

Le Chapitre 3 est consacré à l'état de l'art. Il s'ouvre sur une clarification terminologique essentielle distinguant la notion d'environnement virtuel de celle, souvent employée à tort, de "jumeau numérique". Nous introduirons ensuite les concepts fondamentaux de l'intelligence artificielle, de l'apprentissage automatique et de l'apprentissage profond, qui constituent la colonne vertébrale de nos solutions. La suite du chapitre dresse un panorama de l'IA appliquée à la simulation puis cible successivement deux domaines qui se recoupent au cœur de nos travaux : l'IA générative et les architectures dédiées au traitement de séquences.

Le Chapitre 4 présente notre première contribution méthodologique, dont les résultats ont fait l'objet d'une publication à la conférence EUSIPCO. Ce chapitre introduit les fondations de notre approche d'apprentissage : nous proposerons une méthode de pondération de l'erreur ainsi qu'une stratégie d'apprentissage spécifique qui serviront de socle pour la suite de nos recherches.

Le Chapitre 5 expose la contribution principale de cette thèse. Nous y démontrerons comment les architectures de type *Transformers* sont particulièrement adaptées pour générer des séquences d'impulsions. Pour surmonter les limites de ces modèles face à des flux de données continus, nous introduirons un principe de fenêtrage dynamique, permettant de traiter et de générer des séquences extrêmement longues, répondant ainsi aux exigences des scénarios problématiques pour notre environnement numérique.

Enfin, le Chapitre 6 explore une approche alternative et complémentaire basée sur le "dépliement" des séquences. L'objectif de cette contribution est de repenser la représentation de la donnée en entrée pour se rapprocher de la réalité des traitements de Détection et Caractérisation des Impulsions (DCI). Cette nouvelle modélisation du problème permet de réintégrer dans la course des architectures qui s'avéraient jusqu'alors inadaptées, ouvrant ainsi de nouvelles perspectives.

Problématique (plan)

Le chapitre sur la problématique contiendra les éléments suivants :

- Description du fonctionnement du capteur ESM dans l'environnement (dans la limite de ce qu'on peut dire), intégré dans la chaîne algorithmique de traitement de l'information.
- Description du fonctionnement de l'environnement virtuel, avec l'explication des modélisations de chaque traitement.
- Spécification du goulot d'étranglement et commentaire sur les données I/O.
- Commentaire sur le complexité du problème pour l'apprentissage automatique : double problématique génération et traitement de séquence.
- **Penser à ajouter des références de traitement avec ce principe de mesureur (brevet?) pour montrer qu'on ne révèle pas des secrets**

2 - Problématique

2.1 . Introduction

L'objectif central de ces travaux de recherche est d'accélérer par intelligence artificielle la simulation d'un environnement virtuel (VE - Virtual Environment) modélisant l'interaction entre des impulsions RADAR et un capteur de Mesures de Soutien Électronique (ESM - Electronic Support Measures). Ce chapitre s'attache à définir précisément le périmètre et les enjeux de cette problématique. Dans un premier temps, nous détaillerons le fonctionnement opérationnel du capteur ESM et son intégration au sein de la chaîne de traitement de l'information tactique. Cette analyse contextuelle permettra de justifier la nécessité de disposer d'un VE de simulation capable de reproduire fidèlement le comportement du capteur pour des besoins de validation. Nous exposerons ensuite l'architecture de cet VE afin d'identifier les verrous technologiques qui limitent actuellement ses performances, en localisant spécifiquement le goulot d'étranglement computationnel. Cette analyse conduira finalement à la formalisation du problème d'accélération sous l'angle de l'apprentissage automatique, en mettant en évidence sa double nature de traitement de séquence et de génération conditionnelle.

2.2 . Description de l'intégration du capteur de Mesures de Soutien Électronique et son fonctionnement

2.2.1 . Contexte opérationnel : La maîtrise du spectre électromagnétique

Dans le cadre d'un scénario de guerre électronique (GE), la survie de l'aéronef dépend de sa capacité à percevoir et comprendre son environnement électromagnétique (EM). L'appareil évolue dans un espace abondant d'émissions provenant de RADAR adverses ou civils, au sol ou aéroportés, cherchant eux-mêmes à détecter leur cible. Les caractéristiques techniques de ces signaux, telles que la fréquence, la largeur d'impulsion ou la période de répétition, constituent une signature unique permettant d'identifier l'émetteur et d'en déduire ses intentions tactiques (veille, poursuite, engagement). La mission des capteurs de ESM, illustrée dans la partie droite de la figure 2.1, est alors d'assurer une écoute passive et discrète du spectre pour détecter, de caractériser et de localiser ces menaces potentielles. Ils fournissent ainsi les données critiques à la décision stratégique et aux contre-mesures.

2.2.2 . Chaîne de traitement de l'information

L'intégration du capteur s'inscrit dans une architecture de traitement séquentielle visant à transformer un signal physique brut en renseignement tactique exploitable. Le champ EM incident est initialement capté par le capteur ESM, qui opère la première conversion fondamentale : il détecte les impulsions radar et construit en temps réel des descripteurs numériques, les PDW (Pulse Description Word - Mot de Description d'Impulsion). Chaque PDW synthétise les

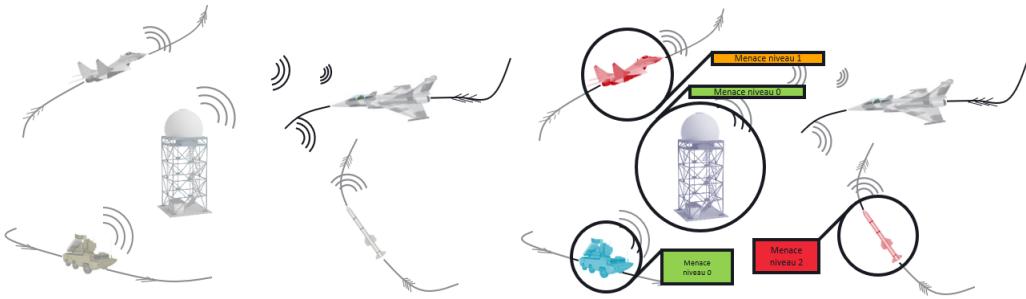


Figure 2.1 – Visualisation de l’apport tactique. Le système transforme un environnement EM dense (gauche) en une situation tactique intelligible (droite), permettant la localisation et la classification des plateformes présentes.

paramètres mesurés de l’impulsion : Date d’arrivée (ToA), Largeur d’impulsion (LI), Fréquence, Niveau de puissance et Direction d’arrivée (DoA). Ce flux continu de PDW alimente ensuite les algorithmes de traitement de haut niveau. Une étape de désentrelacement regroupe d’abord les impulsions par émetteur sur un horizon temporel court, isolant les trains d’impulsions cohérents. Ces regroupements élémentaires sont ensuite consolidés par un processus de pistage (tracking) qui suit l’évolution des émetteurs sur le long terme pour en caractériser la cinématique et le mode de fonctionnement. Finalement, ces pistes enrichies sont confrontées à des bases de données de signatures pour identifier formellement le système d’arme associé et évaluer le niveau de menace immédiat.

La figure 2.2 présente la chaîne de traitement d’un capteur réel (haut) ainsi que deux approches de simulation correspondant à un VE (milieu) et un VE avec IA (bas).

2.2.3 . Architecture et fonctionnement du capteur

Le fonctionnement interne du capteur ESM ne se limite pas à une conversion analogique-numérique transparente ; il constitue une chaîne de traitements physiques et logiques qui conditionne structurellement la qualité des données produites. Les traitements décrits ci-après, et illustré de manière simplifié au milieu de la figure 2.2, constituent le socle architectural de la plupart des récepteurs numériques modernes, bien que des variantes d’implémentation, dictées par les contraintes matérielles des systèmes temps réel, puissent exister selon les constructeurs.

Le traitement débute par la conversion du champ EM incident en signal électrique analogique. L’objectif est ensuite d’extraire, sur des fenêtres temporelles successives, les raies spectrales significatives. La mission de surveillance de bandes passantes instantanées de plusieurs gigahertz (typiquement 2 – 18GHz) impose l’usage de bancs de convertisseurs analogique-numérique (au moins 3) fonctionnant en parallèle à des cadences inférieures à la fréquence de Nyquist (ex : 1GHz, 1.2GHz, 1.4GHz.) [2]. Ce choix architectural induit un repliement spectral

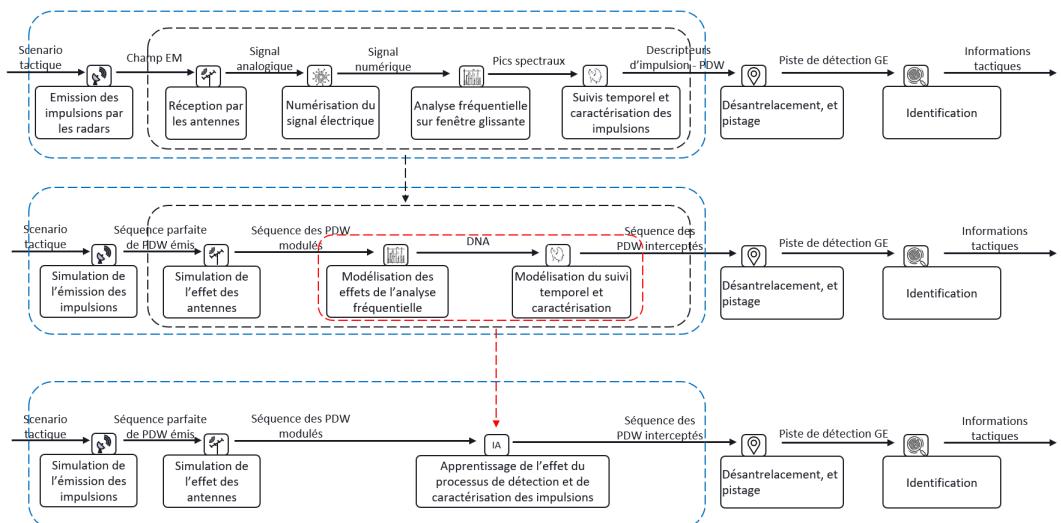


Figure 2.2 – Illustration des chaînes de transformation des informations pour un capteur réel (haut), un VE (milieu) et un VE avec IA (bas). **Haut :** La vérité terrain dicte l'émission d'impulsions, le signal physique est intercepté et les PDW sont déterminées par le capteur (en pointillés noirs). Ces informations sont agrégées (désentrelacement, pistage) et analysées (identification) pour produire un renseignement exploitable sur la vérité terrain initiale. **Milieu :** Un VE (périmètre en pointillés bleus) modélise la physique, à partir de l'émission des impulsions jusqu'à la caractérisation des PDW. La simulation des blocs est comportementale. **Bas :** Le goulot d'étranglement du VE (en pointillés rouges) est remplacé par un module d'IA. L'enchaînement des blocs de traitement constitue encore un VE mais avec IA.

systématique sur chaque voie d'acquisition, mais permet la détermination de la fréquence réelle grâce à la résolution d'un système de congruences entre les différentes voies repliées, principe connu sous le nom de théorème des restes chinois [3], [4]. Cependant, sur chaque canal d'acquisition, des phénomènes de masquage peuvent intervenir, liés à la saturation des convertisseurs, aux harmoniques et/ou à l'intermodulation provoquée par des impulsions de haute énergie, ou à la proximité fréquentielle entre signaux repliés. De plus, la résolution d'ambiguïté s'appuie généralement sur une sélection restreinte des N pics spectraux les plus énergétiques par canal, liée aux contraintes matérielles. La conjonction du bruit thermique sur des canaux critiques et de cette sélection limitative peut conduire à l'échec de la levée d'ambiguïté, entraînant la perte d'impulsions sur la fenêtre temporelle considérée. Les fréquences non-ambiguës identifiées, associées leur énergie, sont qualifiées de détections non-ambiguës (DNA).

Une fois les DNA identifiées sur la fenêtre d'analyse courante, elles sont transmises collectivement au système de suivi temporel. Ce module reçoit ainsi un paquet de détections simultanées qu'il doit associer aux ressources mémoires actives, qualifiées de "pistes" [5]. Le processus d'association est séquentiel et hiérarchisé : les DNA sont traitées une à une, par ordre de priorité croissante selon leur amplitude, afin de privilégier le suivi des signaux les plus énergétiques. Pour chaque détection candidate, le système recherche une correspondance parmi les pistes actives sur la base de critères de tolérance prédéfinis (proximité fréquentielle, cohérence de niveau). Une contrainte stricte d'unicité s'applique alors : une piste ne peut être mise à jour qu'une seule fois par fenêtre temporelle. En cas de corrélation valide avec une piste disponible (non encore mise à jour sur ce cycle), celle-ci intègre les nouveaux paramètres et prolonge sa durée de vie. Si aucune association n'est possible, ou si la piste candidate a déjà été servie par une détection prioritaire, une nouvelle piste est ouverte pour le signal, sous réserve qu'une ressource mémoire soit libre. La clôture d'une piste et la génération du PDW final [6] s'opèrent finalement lorsqu'elle n'a pas reçu de mise à jour durant une période seuil ou lorsque l'impulsion dépasse sa limite de durée opérationnelle.

La limitation matérielle du nombre de ces mémoires, conjuguée à leur logique d'allocation, engendre des artefacts de segmentation spécifiques. Premièrement, une contrainte de latence maximale impose de segmenter artificiellement les impulsions très longues ou continues pour assurer des mises à jour périodiques, générant une série de PDW contigus. Deuxièmement, les échecs de résolution d'ambiguïté peuvent provoquer une rupture de suivi prématurée. Si le masquage est transitoire, la piste reprend après une interruption, scindant l'impulsion en plusieurs entités. Si le masquage persiste jusqu'à la fin de l'émission, le suivi s'arrête définitivement, tronquant la fin du signal. Troisièmement, la saturation des ressources mémoires en environnement dense impacte directement la détection : si aucune piste n'est disponible à l'apparition du signal, il sera ignoré sur l'instant. Cela conduit soit à une acquisition tardive dès la libération d'une ressource, amputant alors le début du signal, soit à une perte totale de l'impulsion si aucune ressource ne se libère à temps. À l'inverse de ces phénomènes de fragmentation ou de perte, la résolution temporelle finie des bancs de filtres peut conduire à l'amalgame de deux impulsions brèves et rapprochées en un seul descripteur. Ainsi, la séquence de PDW produite ne doit pas être considérée comme une simple mesure dégradée de la réalité, mais comme une reconstruction interprétée, portant intrinsèquement la signature des limitations fréquentielles

et des heuristiques de gestion de ressources du capteur.

Cette problématique d'interprétation est illustrée par la figure 2.3, issue d'une campagne de mesures réelles. Si cette visualisation permet d'appréhender la structure macroscopique des données interceptées — notamment la parcimonie des impulsions et leur organisation en trains caractéristiques du fonctionnement radar — elle révèle surtout les limites de l'analyse sur données réelles. Dans ce contexte opérationnel, la seule réalité observable est celle restituée par le capteur ESM; il n'existe aucune "vérité terrain" absolue permettant de certifier si un train d'impulsions est intégral ou s'il résulte de la fragmentation d'une émission plus longue. Cette incertitude quant à la signature du traitement interne rend délicate la validation formelle des algorithmes de désentrelacement et d'identification, soulignant la nécessité de disposer de données maîtrisées. À l'inverse, la figure 2.4 exploite des impulsions générées par un environnement numérique pour objectiver ces biais. Elle permet d'établir une comparaison entre l'état des informations conceptuelles transitant avant la modélisation de la DCI, considérées comme une quasi-vérité terrain, et leur état après caractérisation par le capteur, c'est-à-dire les PDW interceptés. Cette visualisation exploite une représentation dans le plan temps-fréquence pour mettre en exergue les phénomènes d'altération de l'information liés au traitement numérique, rendant les effets du traitement visibles et descriptibles.

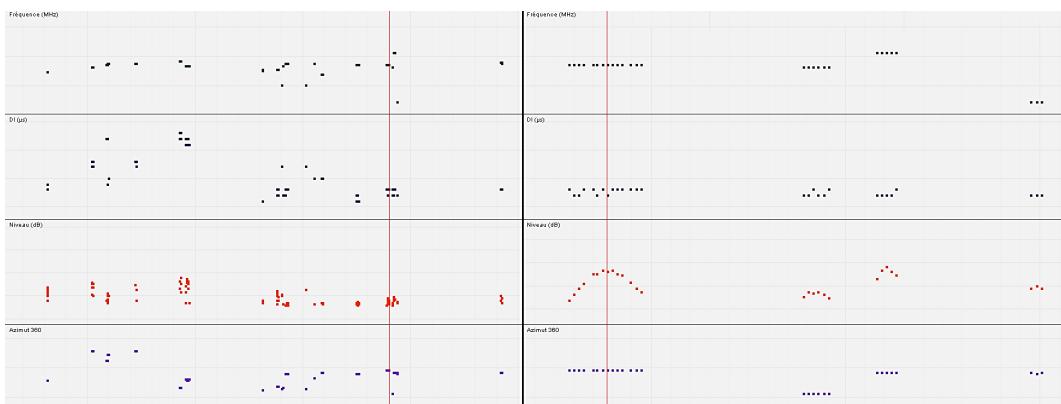


Figure 2.3 – Visualisation d'un flux d'impulsions intercepté par un capteur ESM lors d'un campagne d'essai, mettant en évidence la structure temporelle par paquets des émissions RADAR. Les quatre sous-graphiques superposés détaillent l'évolution des paramètres scalaires de chaque PDW en fonction de sa date d'arrivée. Le marqueur vertical rouge matérialise la correspondance temporelle entre la vue globale (gauche) et la vue locale (droite). Cette dernière, correspondant à un facteur d agrandissement de 25, permet d observer la structure fine du signal et révèle sur l'axe de l'amplitude la modulation caractéristique du lobe de balayage d'antenne du RADAR émetteur.

2.3 . Description de l'environnement virtuel

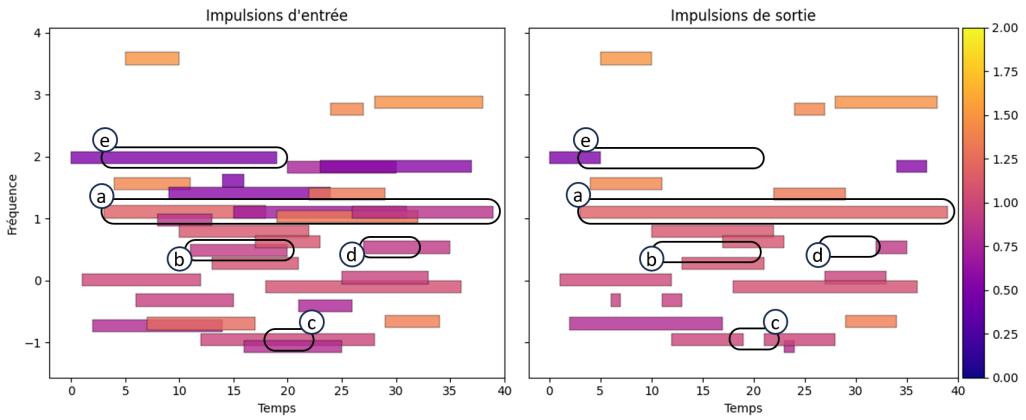


Figure 2.4 – Représentation dans un plan temps-fréquence des informations conceptuelles transitant dans la chaîne de traitement, visualisées à l'étape précédent (gauche) et succédant (droite) à la modélisation du processus de DCI (Détection et Caractérisation des Impulsions). La couleur de chaque rectangle encode l'amplitude du PDW. Les annotations identifient les dégradations structurelles induites par la chaîne de traitement : (a) Fusion de signaux distincts, (b) Disparition d'impulsion (non-détection), (c) Scission artificielle d'une impulsion continue, (d) Troncature avant, (e) Troncature arrière

2.3.1 . Motivations

La phase d'Intégration, Vérification, Validation et Qualification (IVVQ) des algorithmes de GE, tels que le désentrelacement, le pistage et l'identification, exige de disposer de jeux de données rigoureusement contrôlés. Pour valider la chaîne de traitement, il est nécessaire de confronter les données d'entrée des algorithmes - le flux de PDW en sortie du capteur ESM - aux données de sortie attendues, c'est-à-dire la situation tactique restituée. Or, l'obtention de ces données par des essais en vol réels se heurte à des contraintes majeures. En effet, la connaissance exacte et exhaustive de la situation tactique (la position et l'activité de tous les émetteurs environnants) est souvent impossible à garantir, empêchant d'établir une "vérité terrain" fiable pour qualifier les algorithmes. De plus, la réalisation d'essais en vol dédiés représente un défi logistique et financier considérable. La constitution d'un scénario réaliste implique le déploiement de moyens conséquents, tels que des avions plastrons ou des stations radars au sol, dont la disponibilité est limitée. Par ailleurs, ces essais physiques ne permettent de couvrir qu'un spectre restreint de configurations géométriques et EM, limitant la diversité des données récoltées. Face à ces obstacles, le recours à un simulateur d'impulsions s'impose comme la solution de référence. Comme l'illustre la partie centrale de la figure 2.2, l'environnement virtuel (encadré en pointillés bleus) s'insère en amont de la chaîne de traitement de l'information pour fermer la boucle de validation. En générant, à partir d'une vérité terrain parfaitement maîtrisée (le scénario tactique), le flux de PDW que les capteurs auraient intercepté, il permet d'alimenter les algorithmes de post-traitement et de confronter directement la situation tactique reconstruite au scénario initial. Cette approche autorise la simulation d'une quantité massive de données et la construction

de scénarios d'une complexité arbitraire, incluant des cas limites difficilement reproductibles en vol, en assurant une maîtrise totale des paramètres d'entrée. Cette capacité de tests intensifs est indispensable pour analyser finement la réaction de la chaîne algorithmique et procéder aux itérations nécessaires à son amélioration.

2.3.2 . Architecture et fonctionnement de l'environnement virtuel

L'environnement virtuel est structuré en quatre modules fonctionnels séquentiels. Il repose sur une approche de modélisation comportementale, dont l'objectif n'est pas de reproduire le traitement du signal échantillon par échantillon, mais de simuler l'effet macroscopique des traitements physiques et logiques sur le flux d'impulsions incident, afin de prédire les PDW que le capteur aurait effectivement générés. Les quatre modules présentés ci-après sont schématisés dans la partie centrale de la figure 2.2, dans l'environnement numérique (encadré en pointillés bleus).

Le premier module assure la génération de la vérité terrain EM. Il ingère les fichiers de description cinématique (trajectoires du porteur et des radars environnants) ainsi que les séquences d'émission théoriques de chaque radar. À partir de ces données, il construit une liste chronologique d'impulsions émises, triées par date d'arrivée (ToA), où chaque impulsion est décrite par un PDW idéal contenant ses caractéristiques physiques natives.

Le second module modélise la chaîne de propagation et de réception analogique. Sa fonction est double : premièrement, il applique l'équation du bilan de liaison pour déterminer l'amplitude du signal arrivant au voisinage du porteur, en tenant compte de la position relative émetteur-récepteur et de la direction d'émission. Deuxièmement, il simule la fonction de transfert des antennes réceptrices. En exploitant les diagrammes de gain et l'angle d'incidence du signal, ce module calcule l'atténuation ou l'amplification subie par le signal électrique en sortie d'antenne. La sortie de ce bloc est une liste de PDW dont l'amplitude a été ajustée pour refléter la puissance réellement disponible à l'entrée du récepteur numérique.

Les deux derniers modules simulent le cœur du traitement numérique : la détection spectrale et la caractérisation temporelle. Pour s'abstraire de la simulation coûteuse du temps continu, ces modules opèrent sur une échelle temporelle discrétisée nommée "palier". Un palier définit un intervalle de temps au cours duquel l'environnement EM est considéré comme stationnaire : il est délimité par l'apparition ou la disparition d'une impulsion quelconque. Cette hypothèse de stationnarité permet de considérer que les fréquences détectables par le capteur restent constantes sur toute la durée du palier, autorisant un calcul unique par intervalle.

Le troisième module se charge de la modélisation spectrale à l'échelle de ce palier. Il reproduit les effets de l'architecture sous-Nyquist en calculant, pour chaque impulsion présente, ses fréquences repliées sur les différents canaux d'acquisition. En comparant les niveaux relatifs des signaux concurrents et en appliquant les seuils de sensibilité matériels, le module détermine la visibilité de chaque impulsion. Une impulsion est considérée comme "détectée" — c'est-à-dire que son ambiguïté fréquentielle aurait été levée avec succès — si elle reste visible

et non masquée sur au moins trois canaux simultanément, elle est alors qualifiée de DNA.

Enfin, le quatrième module reproduit le mécanisme de suivi temporel. Son fonctionnement mime la logique interne du capteur décrite précédemment, mais la cadence de mise à jour est ici dictée par les paliers et non par les fenêtres d'échantillonnage. À chaque palier, les DNA sont comparées aux pistes actives. En cas de corrélation, la piste est maintenue; sinon, une nouvelle piste est allouée sous réserve de disponibilité mémoire. Ce module gère également les clôtures de pistes : si une piste n'est pas mise à jour pendant une durée critique, elle est fermée. De même, pour simuler la segmentation des signaux continus ou longs, une logique de coupure forcée est implémentée : lorsqu'une piste dépasse une durée maximale d'ouverture, elle est close (génération d'un PDW), puis immédiatement rouverte pour poursuivre le suivi, reproduisant ainsi fidèlement les artefacts de segmentation du capteur réel.

En conclusion, l'environnement virtuel articule une chaîne de transformation cohérente qui mime le cycle de vie complet de l'information EM. En élaborant successivement une séquence de PDW idéaux issue de la vérité terrain, en leur appliquant les modulations radiométriques propres à la chaîne d'acquisition, puis en soumettant ce flux aux logiques de sélection spectrale et de suivi temporel du récepteur, le simulateur parvient à reproduire la séquence de PDW qu'un capteur aurait effectivement interceptée pour un scénario donné. Bien que cette modélisation comportementale constitue par essence une approximation par rapport à une simulation physique du signal au niveau de l'échantillon, elle offre un compromis entre la précision des phénomènes reproduits et la charge de calcul. La représentativité des artefacts générés — incluant les effets de masquage, de fragmentation et de saturation — s'avère suffisante pour garantir la pertinence des données synthétiques, permettant ainsi de répondre aux exigences de diversité et de volume nécessaires à la validation robuste des algorithmes de traitement de l'information sans dépendre exclusivement des essais en vol. Ces artefacts peuvent être visualisés dans la partie droite de la figure 2.4.

2.4 . Identification du goulot d'étranglement et limites opérationnelles

2.4.1 . L'impératif de temps réel et le coût de la simulation

Si l'architecture modulaire de l'environnement virtuel garantit une fidélité satisfaisante des données produites, son exploitation opérationnelle se heurte à une contrainte majeure de performance temporelle. Actuellement, la simulation détaillée des traitements du capteur présente un coefficient d'expansion temporel moyen de l'ordre de 100 : la simulation d'une seule seconde de scénario nécessite environ cent secondes de calcul. Cette latence prohibe l'utilisation du simulateur pour des applications nécessitant une interaction en temps réel, telles que la formation des pilotes et des opérateurs de GE. Par ailleurs, même dans le cadre de la validation algorithmique hors ligne, ce coût calculatoire devient un obstacle à la génération massive de données. La validation statistique robuste des algorithmes de désentrelacement ou d'identification exige de couvrir des milliers de variations de scénarios, une tâche qui, avec le coefficient

d'expansion actuel, nécessiterait des temps de calcul incompatibles avec les cycles de développement industriels, particulièrement pour les scénarios denses.

2.4.2 . Analyse de la complexité et localisation du verrou

L'analyse de profilage de l'environnement virtuel permet de localiser précisément l'origine de cette latence au niveau des troisième et quatrième modules fonctionnels, responsables de la détection spectrale, du pistage et de la caractérisation des impulsions. Une distinction structurelle fondamentale sépare ces modules des étages amont. Les modules 1 et 2 (génération et propagation) appliquent des transformations physiques indépendantes sur chaque impulsion; ils se prêtent donc naturellement à une parallélisation sur CPU voire GPU. À l'inverse, les modules 3 et 4 opèrent intrinsèquement de manière séquentielle : l'état du système à l'instant t (les pistes actives, les masquages en cours) dépend de l'histoire du traitement, empêchant toute parallélisation temporelle simple. De surcroît, le nombre de paliers temporels à traiter croît linéairement avec la densité d'impulsions du scénario. Or, au sein de chaque palier, le module de détection doit effectuer des comparaisons croisées entre toutes les impulsions présentes pour résoudre les masquages et les ambiguïtés, induisant une complexité quadratique par rapport au nombre d'impulsions locales. Cette combinaison d'une structure séquentielle rigide et d'une complexité locale quadratique crée un goulot d'étranglement dès que la densité du scénario augmente. Ce verrou, situé au cœur de la logique de traitement du capteur, constitue l'obstacle technologique qu'il est nécessaire de lever. L'objectif de la thèse est donc d'accélérer l'environnement virtuel en substituant ces deux modules, que nous désignerons conjointement comme le système de détection et de caractérisation des impulsions (DCI), par un modèle d'intelligence artificielle optimisé. Cette approche est schématisée par les parties centrale et inférieure de la figure 2.2 : le goulot d'étranglement, mis en évidence par l'encadré en pointillés rouges, est remplacé par un module d'apprentissage automatique, définissant ainsi la nouvelle chaîne de traitement de l'information.

2.5 . Formalisation et complexité du problème d'apprentissage

2.5.1 . Nature des données et définition formelle de la tâche

L'objectif est de substituer au DCI un modèle appris capable d'approximer sa dynamique de fonctionnement. Du point de vue des données, cette tâche se formalise comme un problème de transduction de séquence s'opérant dans un espace continu. L'entrée du modèle, notée X , est la séquence de PDW "modulés" issue du module de propagation (module 2). Elle représente l'information physique brute incidente aux bornes des convertisseurs analogique-numérique. La cible à prédire, notée Y , est la séquence de PDW "capteur" (sortie du module 4), correspondant aux impulsions effectivement construites et publiées par le système. Ces flux de données sont illustrés par la figure 2.4, qui en propose une visualisation temps-fréquence où l'amplitude est codée par la couleur. Dans ce formalisme, les désignations "données d'entrée" et "données de sortie" se définissent par rapport aux bornes du goulot d'étranglement identifié précédemment (le DCI) : la mission de l'IA consiste à reproduire la fonction de génération permettant de

construire la séquence de sortie Y conditionnellement à l'observation de la séquence d'entrée X .

Ces données présentent deux caractéristiques structurelles qui font leurs particularités. Premièrement, elles appartiennent à un espace continu multidimensionnel. Chaque élément constitutif des séquences X et Y est un vecteur de valeurs réelles (Date, Fréquence, Largeur, Amplitude, Azimut) définissant les propriétés physiques de l'impulsion. Deuxièmement, les séquences se caractérisent par une cardinalité variable et asynchrone. La topologie temporelle diffère entre l'entrée et la sortie en raison des mécanismes internes du capteur (masquages, fusions, scissions). Ainsi, la longueur N de la séquence d'entrée et la longueur M de la séquence de sortie sont variables et ne respectent pas de règle de proportionnalité stricte ou de synchronisation élément par élément.

2.5.2 . Une double problématique : Traitement de séquence et Génération

Cette caractérisation ancre le problème à l'intersection de deux paradigmes de l'apprentissage profond, induisant une spécificité qui le distingue des applications classiques.

D'une part, il s'agit fondamentalement d'un problème de traitement de séquence (Sequence Processing), imposé par la nature variable et décorrélée des dimensions temporelles en jeu. Contrairement à des tâches de régression à taille fixe, le modèle doit traiter une série d'entrée de longueur arbitraire N pour produire une série de sortie de longueur M , sans qu'il existe de règle de proportionnalité simple entre N et M . Cette structure constraint à l'utilisation de mécanismes capables de mapper une séquence de taille variable vers une représentation latente fixe ou dynamique. De plus, la tâche se trouve complexifiée par la nature continue des données. Contrairement aux tâches de Traitement du Langage Naturel (NLP) où les séquences sont formées de mots issus d'un dictionnaire fini, les PDW exigent une précision numérique fine sur plusieurs variables continues simultanément (Fréquence, Largeur, Amplitude, etc.). Toute tentative de discréétisation de cet espace pour le ramener à un problème de classification fini se heurterait à la *malédiction de la dimensionnalité* [7] : pour un descripteur composé de seulement cinq variables physiques (Fréquence, Largeur, Amplitude, Azimut, Date), une grille de résolution modeste de cent intervalles par dimension engendrerait un espace d'états de 100^5 cellules, soit dix milliards de classes potentielles. Une telle cardinalité, supérieure de plusieurs ordres de grandeur aux vocabulaires des modèles de langage massifs (50000 classes), rend la méthode intraitable. L'espace de travail est ainsi mathématiquement condamné à rester continu.

D'autre part, il s'agit d'un problème de génération conditionnelle. Le modèle ne doit pas simplement filtrer ou altérer les impulsions incidentes, mais construire intégralement une séquence de sortie dont la structure événementielle diffère fondamentalement de l'entrée. La logique de suivi temporel du capteur brise la relation bijective : un événement physique unique peut se trouver traduit par une succession de plusieurs PDW distincts, ou inversement, plusieurs événements peuvent être fusionnés. Le modèle doit donc apprendre à générer une nouvelle série de descripteurs qui constituent une interprétation synthétique de la réalité physique.

Dans cette reconstruction, la séquence de sortie devient une entité structurellement autonome plutôt qu'une version simplement dégradée de l'entrée.

Cette dualité, combinée à l'exigence de précision, écarte les solutions sur étagère et nécessite la conception d'une architecture capable de marier les mécanismes propres à la compréhension de contexte et les capacités de régression fine nécessaires à la reconstruction physique du signal.

État de l'art : PLAN (à supprimer après rédaction)

Le chapitre sur l'état de l'art se découpe en 4 parties.

2.6 . Introduction

Cette section aborde les aspects suivants :

- Environnements numériques
- Injection 1 : génération et modélisation de l'environnement
- Injection 2 : simulation de phénomènes physiques
- Injection 3 : adaptation et interaction

2.7 . IA générative

Cette section présente le domaine de l'IA générative. Notre problème peut y être naïvement associé mais en réalité quasiment aucune des méthodes ne sera applicable. Les aspects présentés sont :

- Le concept d'IA générative. En notant que n'importe quelle fonction génère une sortie à partir d'une entrée et que la dériver de tout appeler IA générative est tentante.
- Les VAE et spécialement VAE conditionnels
- Les GAN et spécialement GAN conditionnels
- Les modèles de diffusion et spécialement ceux conditionnels
- Les modèles de langage et GPT

2.8 . Méthodes pour le traitement de séquence

Cette section présente les architectures connues pour leurs capacités à traiter des séquences, de leurs formes les plus simples aux formes les plus complexes. Par ordre d'apparition :

- Le concept de séquence : notion de proximité dans un ensemble. Série temporelle, image, texte.
- Réseau de convolution :
 - Histoire de son apparition : dans l'image
 - Comment la convolution interagit avec la séquence
 - La convolution dans l'image (vue comme une séquence)
 - La convolution dans le texte
 - La convolution ailleurs
- Réseau de neurones récurrents :
 - Histoire de son apparition
 - Comment un RNN interagit avec la séquence
 - Variante SSM

- RNN dans le texte
- RNN dans les systèmes temporels (chaine de Markov)
- Transformer :
 - Histoire de son apparition : dans le langage
 - Comment le Transformer interagit avec la séquence
 - Transformer dans le texte (traduction, GPT, ...)
 - Transformer dans les systèmes temporels (chaine de Markov)
 - Transformer dans l'image
 - Transformer ailleurs (généralisation)

2.9 . Les améliorations

Cette section met en avant les difficultés liées à l'apprentissage automatique, entre complexité calculatoire, mémorielle et instabilité en entraînement. À cette occasion, nous montrons les propositions existantes visant à résoudre ces problèmes. Par ordre d'apparition :

- Compréhension des architectures : Mechanistic Interpretability
- Présentation des soucis de performances
- Présentation des solutions aux soucis de performances
 - Positional Encoding
 - Certains mécanismes d'attention
 - Pre-Training
 - Embedding et tokenization
- Présentation des soucis de stabilité
- Présentation des solutions aux soucis de stabilité :
 - Layer-norm
 - Initialisation
 - Structure (hyper-paramètre de manière générale)
- Présentation des soucis d'efficacité et leurs solutions
 - Complexité mémoire et calcul : mécanisme d'attention
 - Vitesse d'entraînement : MAMBA?

3 - État de l'art

Ce chapitre présente les concepts et méthodes fondamentaux sur lesquels s'appuie cette thèse. Nous commencerons par définir la notion d'Environnement Numérique, avant de poser les principes mathématiques de l'apprentissage profond qui nous permettront d'introduire les différents apports de l'IA à la simulation. Nous exposerons ensuite les approches d'IA générative, pour enfin nous concentrer sur les méthodes de traitement de séquence, en détaillant particulièrement l'architecture Transformer. Nous conclurons par une analyse des défis inhérents à l'entraînement de ces modèles profonds, en passant en revue les solutions techniques existantes pour garantir leur stabilité, leur efficacité calculatoire et leur interprétabilité.

3.1 . Introduction : Environnements Numériques et typologie des apports de l'IA

L'utilisation de représentations virtuelles pour l'analyse et l'optimisation des systèmes physiques s'est aujourd'hui largement généralisée. Dans ce contexte, les termes de « jumeau numérique » et d'« environnement numérique » sont souvent employés de manière interchangeable. Cette première sous-section a pour objectif de démêler cette confusion sémantique. Afin d'introduire rigoureusement les outils capables de modéliser et d'animer ces environnements virtuels, nous formaliserons ensuite les principes mathématiques de l'apprentissage profond. Enfin, nous dresserons une typologie des apports de l'Intelligence Artificielle à la simulation, structurée autour de trois axes d'intervention : la constitution géométrique et visuelle, la représentation des phénomènes physiques, et les capacités d'interaction dynamique.

3.1.1 . Cadre Conceptuel : Environnement Virtuel et Jumeau Numérique

Le concept de jumeau numérique, popularisé et formalisé dès le début des années 2000 par les travaux de Michael Grieves dans le domaine manufacturier [8], puis théorisé comme un pilier des systèmes cyber-physiques (CPS) par des auteurs comme Negri et al. [9], a connu une adoption rapide et variée à travers l'industrie.

Si le terme de "jumeau numérique" s'est imposé dans le paysage technologique, sa définition précise fait l'objet d'un débat animé entre une vision idéale et une approche pragmatique. D'un côté, une conception formelle, s'appuyant sur les travaux fondateurs de la NASA [8], défend l'idée qu'un véritable jumeau numérique se caractérise par un couplage bidirectionnel et dynamique avec son homologue physique. Dans cette perspective exigeante, le jumeau n'est pas une simple représentation ; il constitue une instance virtuelle dont l'état est synchronisé en continu par les flux de données issus du système physique et qui, en retour, pilote, optimise et prédit le comportement de ce dernier [9].. Cette boucle fermée est considérée comme la condition permettant de distinguer le jumeau numérique d'un simple modèle ou d'une simulation. De l'autre, une approche plus pragmatique, largement répandue dans l'industrie, adopte une définition évolutive et par niveaux de maturité. Dans cette vision, une maquette 3D enrichie de données, parfois qualifiée de "digital shadow", peut déjà être appelé "jumeau numérique". Cette

flexibilité sémantique, bien que source de confusion, reflète la réalité des projets industriels où la complexité et le coût d'une intégration parfaite imposent une progression par étapes. Malgré tout, une ligne de démarcation essentielle fait consensus : l'existence d'un transfert de données automatique du système physique vers son représentant virtuel. Sans ce flux, la représentation demeure une simulation ou un modèle générique, que nous qualifierons ici d'« environnement numérique ». Par exemple les simulateurs de conduite autonome comme CARLA [10] sont des environnements numériques essentiels pour l'entraînement des algorithmes d'IA, mais ils simulent un monde routier générique non couplé à un véhicule physique unique, et ne sont en ce sens pas des jumeaux numériques. En revanche, certains simulateurs de moteur d'avion, comme ceux déployés par General Electric [11], qui est alimenté en temps réel par les données de vol de l'équipement spécifique, incarnent la définition minimale du jumeau numérique, souvent appelée « Digital Shadow ». Ils permettent un suivi individualisé de l'état de santé et de l'usure de chaque moteur de la flotte.

Pour désigner les représentations numériques qui ne sont pas couplées à une instance physique unique, nous recourons donc au terme plus large et unificateur d'Environnement Numérique (Virtual Environment - VE).

La notion d'Environnement Virtuel est interdisciplinaire, et sa définition varie selon que l'on se place dans la communauté de la Réalité Virtuelle, de l'Ingénierie Système ou de l'Intelligence Artificielle. La recherche en Réalité Virtuelle, historiquement focalisée sur l'immersion sensorielle et l'interaction humain-machine, définit souvent les VE comme des « mondes synthétiques générés par ordinateur dans lesquels l'utilisateur a un sentiment d'être présent et d'y interagir » [12]. Cette perspective met l'accent sur les aspects perceptuels et cognitifs. En revanche, dans les domaines de l'ingénierie et de l'IA, l'accent est davantage porté sur la fonction de simulation et de cadre d'expérimentation. Ici, un VE est vu comme un « modèle informatique exécutable d'un système » [13] ou un « cadre de simulation qui permet le test et la validation d'algorithmes dans des conditions contrôlées et reproductibles » [14]. Cette vision est moins concernée par l'immersion de l'utilisateur que par la fidélité de la modélisation des processus et des interactions.

Pour englober ces différentes finalités – de la formation immersive au banc d'essai algorithmique – nous proposons la définition unificatrice suivante : Un Environnement Virtuel (VE) désigne une simulation numérique interactive modélisant un ensemble d'entités et de phénomènes, dans le but d'observer, d'analyser ou d'expérimenter des comportements au sein d'un cadre contrôlé.

Cette définition permet de tracer une ligne de démarcation nette avec le concept voisin de Jumeau Numérique. La distinction fondamentale réside dans le principe d'individualisation par les données. Le jumeau numérique se définit intrinsèquement comme l'avatar d'une instance physique unique, tel un moteur spécifique, dont l'essence est indissociable d'un lien de données continu avec son homologue réel. En revanche, l'environnement numérique se conçoit comme une représentation générique d'une classe de systèmes, dont la valeur réside dans la modélisation fidèle de lois physiques au sein d'un cadre reproductible, indépendamment d'une instance matérielle particulière.

Ainsi clarifiée, la notion de VE couvre un spectre étendu, allant du monde immersif interactif au simulateur technique. Dans le contexte spécifique du développement algorithmique, qui est le nôtre, le VE devient un outil de prototypage et de validation indispensable : il permet de reproduire des situations expérimentales complexes, de générer des données synthétiques massives et de tester des modèles de manière intensive, sûre et économique, sans les contraintes logistiques des dispositifs physiques réels. Pour structurer l'état de l'art des méthodes d'Intelligence Artificielle au service de ces environnements, nous organiserons la suite de notre analyse selon trois axes d'intervention distincts : la constitution géométrique et visuelle de l'environnement, la représentation des phénomènes physiques par apprentissage, et enfin les capacités d'interaction et d'adaptation dynamique.

3.1.2 . Principes mathématiques de l'Apprentissage Profond

Avant d'explorer les architectures génératives et séquentielles avancées, il convient de formaliser les fondements mathématiques de l'apprentissage profond (*Deep Learning*). Si le vocabulaire associé à l'intelligence artificielle emprunte souvent à la biologie, son mécanisme relève en réalité de l'optimisation mathématique de fonctions paramétriques.

3.1.2.1 . Risque théorique, risque empirique et surapprentissage

Un réseau de neurones artificiels n'est autre qu'une fonction mathématique paramétrique f_θ , qui associe une donnée d'entrée x à une prédiction \hat{y} . La spécificité de cette fonction réside dans sa structure : elle est construite comme la composition successive de L fonctions plus simples, appelées couches. La fonction globale s'écrit ainsi :

$$f_\theta(x) = f_L(f_{L-1}(\dots f_1(x; \theta_1) \dots; \theta_{L-1}); \theta_L)$$

C'est de cet enchevêtrement de couches que le modèle tire son qualificatif de "profond" (*deep*). L'ensemble des paramètres $\theta = \{\theta_1, \dots, \theta_L\}$ (les poids et les biais de chaque couche $(f_i)_{1 \leq i \leq L}$) peut se compter en millions, voire milliards.

L'objectif de l'apprentissage est de trouver le jeu de paramètres optimal θ^* qui minimise le risque théorique, caractérisant la qualité de reconstruction des prédictions futures en les comparant à la vérité terrain. La qualité de reconstruction est calculé via la fonction de perte notée $\mathcal{L}(\hat{y}, y)$, qui mesure l'écart entre la prédiction \hat{y} et la référence y . Le problème d'optimisation se formule comme la minimisation du risque théorique, c'est-à-dire l'espérance de la perte \mathcal{L} sur la distribution de probabilité conjointe $P(x, y)$ régissant le phénomène physique étudié :

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{(x,y) \sim P} [\mathcal{L}(f_\theta(x), y)]$$

Cependant, la distribution P étant inconnue, cette espérance ne peut être calculée analytiquement. La stratégie consiste alors à approximer cette espérance par une moyenne calculée sur un jeu de données d'entraînement fini, constitué de N exemples (x_i, y_i) . Le problème se

transforme ainsi en la minimisation du risque empirique :

$$\hat{\theta}^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_{\theta}(x_i), y_i)$$

Cette substitution de l'espérance par une moyenne empirique met en lumière l'une des limites fondamentales de l'apprentissage automatique : le surapprentissage (*overfitting*) : si l'espace des solutions modélisables est trop permissif, le réseau parvient à exploiter ses nombreux degrés de liberté pour mémoriser parfaitement l'échantillon d'entraînement. Le risque empirique tombe strictement à zéro, mais au prix d'oscillations extrêmes et aberrantes entre ces points, faisant diverger le risque théorique.

Cette divergence trouve son illustration la plus canonique dans la régression polynomiale : lorsqu'un polynôme possède un degré supérieur ou égal au nombre de points d'entraînement, l'espace des solutions autorise une interpolation parfaite. Le risque empirique tombe strictement à zéro, mais au prix d'oscillations extrêmes et aberrantes entre ces points, faisant exploser le véritable risque théorique.

En pratique, on constitue un échantillon de données indépendant, tenu à l'écart de la phase d'optimisation, appelé ensemble de validation. Cet ensemble ne pourra donc pas être influencé positivement par un sur-apprentissage et permet d'évaluer la capacité de généralisation d'un modèle.

3.1.2.2 . L'optimisation par descente de gradient et la règle de la chaîne

Cette minimisation du risque empirique est strictement identique à celle d'un problème d'apprentissage élémentaire tel que la régression linéaire. Cependant, dans une régression linéaire, les approximations sont de simples projections affines et la fonction de perte forme un paysage d'optimisation strictement convexe. Cette simplicité permet de trouver le minimum global de manière directe grâce à une solution analytique exacte (l'équation normale $\theta = (X^T X)^{-1} X^T y$).

Dans le cadre de l'apprentissage profond, la succession des couches (f_1, \dots, f_L) est systématiquement entrelacée de fonctions d'activation non-linéaires. Cette non-linéarité est essentielle pour modéliser des phénomènes complexes, mais elle rend le paysage d'optimisation non-convexe. Il devient indispensable de recourir à une méthode d'optimisation itérative : la descente de gradient. À chaque itération, les paramètres sont mis à jour dans la direction opposée au gradient de l'erreur, selon un pas d'apprentissage α :

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}$$

Pour évaluer ce gradient sur un réseau profond, on formalise le flux d'information par une suite d'états intermédiaires. En posant $a_0 = x$ comme vecteur d'entrée, la sortie de chaque

couche l se définit par la récurrence $a_l = f_l(a_{l-1}; \theta_l)$. La sortie de la dernière couche correspond ainsi à la prédiction finale du modèle : $a_L = \hat{y}$.

Le calcul du gradient $\nabla_{\theta} \mathcal{L}$ pour une matrice de paramètres spécifique θ_l , située dans les couches profondes, repose sur la formule de dérivation des fonctions composées (*chain rule*). La dérivée s'exprime alors comme le produit des matrices Jacobiennes successives le long du graphe de calcul :

$$\frac{\partial \mathcal{L}}{\partial \theta_l} = \frac{\partial \mathcal{L}}{\partial a_L} \frac{\partial a_L}{\partial a_{L-1}} \cdots \frac{\partial a_{l+1}}{\partial a_l} \frac{\partial a_l}{\partial \theta_l}$$

Son implémentation algorithmique, connue sous le nom de rétropropagation du gradient (*backpropagation*), permet de calculer ces termes informatiquement et efficacement en remontant la chaîne de dérivées partielles de la sortie vers l'entrée du réseau .

3.1.2.3 . Géométrie de l'optimisation

La nature itérative et non-convexe de l'optimisation a longtemps nourri la crainte de voir l'algorithme d'apprentissage se bloquer irrémédiablement dans un "minimum local" de mauvaise qualité. Cette intuition, issue de notre perception des espaces géométriques à deux ou trois dimensions, s'avère toutefois trompeuse pour les réseaux profonds.

Des travaux [15], [16] ont démontré qu'en très haute dimension, la probabilité de rencontrer un véritable minimum local de mauvaise qualité est très faible. En réalité, l'écrasante majorité des obstacles rencontrés en apprentissage sont des "points-selles" : des configurations formant des plateaux instables, où la gradient tend vers zéro, provoquant un ralentissement de l'apprentissage. L'emploie d'optimiseurs avancés (comme la méthode Adam) permet de conserver une inertie temporelle (le *momentum*) et de s'échapper de ces zones.

Cette dynamique d'optimisation en régime sur-paramétré éclaire une pratique centrale de l'IA moderne : l'apprentissage se fait désormais sur des gros modèles, dont les connaissances sont ensuite transférées à des modèles plus petits par distillation [17]. Les recherches récentes confirment que cette stratégie n'est pas qu'un artifice d'ingénierie : elle repose sur le constat que l'apprentissage est plus efficace sur les grands modèles, qui capturent des représentations riches que les petits modèles, entraînés seuls, ne peuvent atteindre. Ainsi, un modèle de 6 à 7 milliards de paramètres distillé peut égaler, voire surpasser sur certaines tâches, un modèle de 175 milliards non distillé, tout en offrant des performances opérationnelles très supérieures en vitesse et en coût [18].

Bien que nous n'aspirons pas à construire des modèles immenses, ces faits nous permettent d'envisager une solution dans le domaine de l'apprentissage profond en gardant l'esprit serein.

3.1.3 . L'IA pour la constitution géométrique et visuelle de l'environnement

L'intégration de l'apprentissage profond dans la chaîne de production des environnements virtuels marque une transition technologique de la modélisation explicite vers les représentations apprises. Cette évolution s'articule autour de deux axes. Le premier concerne la capacité de l'IA à encoder la scène non plus sous forme de maillages géométriques, mais via des représentations implicites optimisées pour le réalisme visuel. Ces méthodes encodent la scène dans les poids d'un réseau, permettant de modéliser des phénomènes complexes comme la transparence ou les réflexions dépendantes du point de vue. Elles réduisent ainsi l'écart de performance lors du transfert du simulateur vers le réel en fournissant aux agents des observations capteurs quasi-identiques à la réalité. Le second axe porte sur l'automatisation de la synthèse d'objets, où les modèles génératifs assistent la création géométrique pour accélérer le peuplement des univers virtuels. Ces deux mécanismes, la reconstruction par représentation implicite et la génération d'éléments par diffusion, seront détaillés respectivement dans les deux paragraphes suivants.

3.1.3.1 . Modélisation : reconstruction neurale et représentations implicites

L'enjeu de la numérisation d'environnements pour la simulation est de reproduire le réel avec une fidélité suffisante pour garantir la validité des expérimentations virtuelles. Le défi technique est la Synthèse de Nouveaux Points de Vue qui permet d'observer une scène depuis des angles non capturés initialement. Les méthodes classiques de photogrammétrie montrent ici une limitation majeure : en figeant l'apparence sur des maillages statiques, elles échouent à reproduire les variations d'éclairage dépendantes de la position de l'observateur (reflets, transparence). Pour lever ce verrou, les *Neural Radiance Fields* (NeRF) [19] ont introduit les représentations implicites. En apprenant la fonction de transport de la lumière via un réseau de neurones, cette approche permet de générer des vues inédites où les interactions lumineuses s'adaptent dynamiquement au mouvement de la caméra, garantissant un photoréalisme supérieur.

Toutefois, le coût calculatoire inhérent au NeRF limite son usage interactif. Des optimisations algorithmiques majeures ont été proposées pour pallier cette latence, notamment le *Hash Encoding* multi-résolution [20] qui réduit drastiquement les temps d'apprentissage. Plus récemment, une rupture vers des représentations hybrides a été opérée avec le *3D Gaussian Splatting* [21]. Cette méthode substitue au coûteux échantillonnage volumétrique une projection (rasterization) rapide de gaussiennes 3D anisotropes. Elle concilie ainsi la fidélité visuelle des représentations neurales avec les performances temps réel (> 100 FPS) indispensables à l'interactivité au sein d'un environnement numérique.

3.1.3.2 . Génération : synthèse d'objets par diffusion

Au-delà de la reproduction du réel, la simulation requiert la capacité de générer des environnements variés incluant des objets ou des conditions non observés. L'IA générative intervient ici pour la création d'objets 3D, palliant la rareté des banques de modèles 3D par l'exploitation des vastes ensembles de données image-texte 2D.

L'état de l'art actuel s'appuie sur le transfert de connaissances depuis des modèles de diffusion 2D pré-entraînés vers la 3D. La méthode *DreamFusion* (Poole et al., 2022) [22] a formalisé ce

principe via le *Score Distillation Sampling* (SDS). Cette technique utilise un modèle de diffusion 2D comme fonction de critique pour optimiser une représentation 3D (telle qu'un NeRF), de sorte que ses rendus 2D correspondent à une description textuelle donnée. Des itérations ultérieures, comme *ProlificDreamer* (Wang et al., 2023) [23], ont affiné ce processus via le *Variational Score Distillation* (VSD) pour améliorer la fidélité géométrique et la résolution des textures. Ces approches permettent d'envisager des pipelines de "texte-vers-environnement", où la description sémantique d'une scène suffit à instancier un cadre de simulation complet et physiquement cohérent.

3.1.4 . L'IA pour l'accélération et la modélisation des phénomènes physiques

L'objectif de cette seconde injection est de substituer ou d'accélérer les solveurs numériques traditionnels (Éléments Finis, Volumes Finis) dont la complexité calculatoire limite les applications temps réel. L'état de l'art s'articule autour de la manière dont la connaissance physique est intégrée dans le modèle d'apprentissage. Nous distinguons trois niveaux d'intégration, allant de l'apprentissage pur par les données à l'intégration structurelle des lois physiques.

3.1.4.1 . Apprentissage par Observation (Data-Driven)

Le premier niveau considère le simulateur comme une "boîte noire" dont il faut approximer la fonction de transfert à partir d'observations. L'IA apprend ici les corrélations spatio-temporelles sans connaissance explicite des équations sous-jacentes. Les Graph Neural Networks (GNN) se sont imposés comme l'architecture de référence pour cette tâche, notamment pour les systèmes lagrangiens (particules). Les travaux sur les *Graph Network-based Simulators* (GNS) [24] démontrent une capacité remarquable à prédire la dynamique de fluides et de solides déformables en modélisant les interactions locales par passage de messages. Bien que très rapides à l'inférence, ces modèles souffrent d'un manque de garanties physiques : sans contrainte explicite, ils peuvent violer les lois de conservation (masse, énergie) et dériver sur de longues horizons temporels.

3.1.4.2 . Apprentissage constraint par les Équations (Physics-Informed)

Pour pallier le manque de robustesse physique et la dépendance aux données, une seconde approche intègre les Équations aux Dérivées Partielles (EDP) directement dans l'optimisation. C'est le paradigme des Physics-Informed Neural Networks (PINNs) [25]. Ici, le réseau de neurones agit comme un approximateur universel de la solution, et sa fonction de coût inclut les résidus de l'équation physique (ex : Navier-Stokes). Cette méthode permet de s'affranchir partiellement ou totalement de données d'étiquetage (apprentissage non supervisé par la physique). Cependant, les PINNs font face à des défis d'optimisation majeurs lorsqu'ils sont confrontés à des dynamiques multi-échelles ou chaotiques.

3.1.4.3 . Apprentissage structuré par la Physique (Inductive Bias)

Le troisième niveau d'intégration cherche à inscrire les lois physiques non plus dans la fonction de perte (contrainte douce), mais dans l'architecture même du réseau (contrainte dure ou biais inductif). D'une part, les Hamiltonian Neural Networks (HNN) [26] imposent une structure symplectique au réseau. Au lieu d'apprendre directement les accélérations, le réseau apprend l'Hamiltonien (l'énergie totale) du système, garantissant par construction la conservation de l'énergie et la réversibilité temporelle, ce qui est crucial pour la stabilité des simulations orbitales ou mécaniques sur le très long terme. D'autre part, les Fourier Neural Operators (FNO) [27] exploitent la structure spectrale des solutions d'EDP. En apprenant l'opérateur intégral dans l'espace de Fourier, ils acquièrent une propriété d'invariance à la discréétisation (zero-shot super-resolution), permettant de prédire la physique à des résolutions arbitraires, une propriété structurelle absente des CNN ou MLP classiques.

3.1.5 . L'IA au service de l'interactivité et de l'adaptation décisionnelle

Cette dernière dimension transforme l'environnement numérique d'un cadre passif en un écosystème réactif et adaptatif. L'objectif est d'enrichir la dynamique interactionnelle pour confronter le système sous test à des situations d'une complexité réaliste, impossible à coder manuellement via des scénarios déterministes.

3.1.5.1 . L'environnement peuplé d'agents apprenants (IA comme Acteur)

La première contribution de l'IA est le remplacement des entités scriptées (PNJ, trafic, adversaires) par des agents autonomes pilotés par des politiques neuronales. Contrairement aux machines à états finis classiques, prévisibles et limitées, ces agents sont entraînés via l'Apprentissage par Renforcement Multi-Agents (MARL) ou des mécanismes de Self-Play [28]. Cela permet de peupler le VE d'adversaires ou de collaborateurs capables de stratégies émergentes et optimales. L'exemple du défi DARPA AlphaDogfight (2020), où des agents IA ont développé des manœuvres de combat aérien surclassant les experts humains, illustre comment l'injection d'agents apprenants permet de soumettre le système testé à des niveaux de difficulté et de réalisme inatteignables par des méthodes heuristiques [29]. Ici, l'environnement devient "intelligent" car ses composantes actives s'adaptent au comportement de l'utilisateur ou du système validé.

3.1.5.2 . L'environnement comme générateur de curriculum (IA comme Superviseur)

La seconde contribution concerne le pilotage des paramètres de la simulation par des algorithmes d'optimisation ou évolutionnaires. Au-delà du simple Domain Randomization aléatoire [30], qui manque de direction, l'IA est utilisée pour structurer activement l'apprentissage : c'est l'Apprentissage de Curriculum Automatique (Automatic Curriculum Learning). Des méthodes comme POET (Paired Open-Ended Trailblazer) utilisent des algorithmes évolutionnaires pour co-générer l'environnement en même temps que l'agent [31]. L'algorithme cherche spécifiquement à générer les configurations topologiques ou physiques (terrains accidentés, conditions météo limites) qui maximisent le progrès de l'agent, créant une "course à l'armement" entre la

difficulté du monde et la compétence de l'agent. Dans ce cadre, les algorithmes évolutionnaires agissent comme une forme d'IA générative fonctionnelle, créant des scénarios pertinents et ciblés ("Edge cases") que le hasard seul ne produirait que rarement.

Ainsi, l'IA transforme le simulateur : d'un simple banc d'essai physique, il devient un partenaire d'entraînement actif, capable de générer des opposants redoutables et d'adapter sa propre complexité pour guider l'apprentissage.

3.1.6 . Ancrage dans la problématique

Au regard des distinctions établies précédemment, la classification du système étudié s'opère sans équivoque. Le simulateur de capteur de Mesures de Soutien Électronique que nous analysons a pour fonction de générer des données synthétiques à partir de scénarios tactiques génératrices. Il ne modélise pas le comportement d'un équipement matériel spécifique connecté en temps réel, mais reproduit le fonctionnement théorique d'une classe de capteurs face à des environnements simulés. Par conséquent, l'absence de couplage à une instance physique unique et la nature générique des scénarios valident l'usage exclusif du terme d'Environnement Numérique (VE) pour désigner notre cadre d'étude.

L'analyse des trois axes d'intégration de l'IA révèle par ailleurs la singularité de notre approche au sein de ce VE. Bien que l'objectif fonctionnel rejoigne la "Seconde injection" visant l'accélération de la physique, les méthodes classiques sont inadaptées car elles traitent principalement des champs spatiaux continus. Or, notre goulot d'étranglement réside dans la transformation de séquences d'événements discrets (les PDW). Notre problématique se situe donc à l'intersection de la simulation physique et du traitement de l'information : il s'agit d'apprendre la fonction de transfert du capteur, un processus qui s'apparente conceptuellement à une tâche de "traduction" d'un état physique vers un état perçu. Ce constat motive l'adoption d'une approche fondée sur l'IA générative constructive et les architectures de traitement de séquence, dont nous explorerons les fondements théoriques dans les sections suivantes.

3.2 . IA générative

Dans le paysage contemporain de l'apprentissage profond, la définition de l'IA générative a évolué au-delà de la stricte opposition statistique entre modèles de densité et modèles discriminants. Là où un modèle classique condense l'information (classification, réduction de dimension), un modèle génératif apprend à construire des données de haute dimension, structurées spatialement ou temporellement, en capturant les dépendances complexes inhérentes au jeu d'entraînement. L'IA générative désigne aujourd'hui une classe d'architectures neuronales caractérisée par sa capacité de synthèse.

Un modèle est qualifié de génératif dès lors qu'il construit une donnée structurée en capturant la distribution de probabilité sous-jacente. L'objectif n'est pas simplement d'estimer une valeur locale, mais de bâtir une cohérence globale, respectant les corrélations intrinsèques du

domaine d'apprentissage. Cette définition par la capacité constructive est particulièrement pertinente pour la modélisation de systèmes physiques, où la distinction entre discret et continu s'estompe

Dans le contexte spécifique de l'accélération de simulation, nous nous intéressons particulièrement aux modèles génératifs conditionnels, capables de produire une sortie structurée complexe, tel un champ physique ou un état futur, correspondant à une condition initiale. Cette section explore l'évolution chronologique de ces architectures, depuis les approches opérant dans des espaces continus jusqu'aux paradigmes séquentiels discrets.

3.2.1 . L'approche probabiliste explicite : Les VAE (2013)

La première avancée significative dans l'apprentissage profond de distributions complexes fut l'introduction des Auto-encodeurs Variationnels (VAE). Contrairement aux auto-encodeurs classiques qui compressent l'information en un point déterministe de l'espace latent, les VAE imposent une structure probabiliste à cet espace, généralement sous la forme d'une distribution gaussienne multivariée. L'innovation majeure réside dans l'introduction de l'astuce de reparamétrisation (reparameterization trick), qui rend le processus d'échantillonnage différentiable et permet l'optimisation du modèle par descente de gradient en maximisant la borne inférieure de la vraisemblance (ELBO) [32]. Cette capacité à structurer l'espace latent est particulièrement pertinente pour les problèmes de simulation où une même condition initiale peut mener à plusieurs résultats possibles. Dans leur article sur les VAE Conditionnels (C-VAE) [33], il est prouvé qu'il est possible de modéliser des sorties structurées multimodales en conditionnant la génération à la fois par une variable latente aléatoire et par une observation d'entrée. Bien que théoriquement élégants, les VAE ont souffert historiquement d'une limitation qualitative, leur fonction de perte tendant à produire des résultats lissés. Cependant, des développements récents ont redonné une pertinence majeure à cette famille, notamment via la quantification vectorielle de l'espace latent (VQ-VAE). Ces modèles discrets sont désormais au cœur d'architectures de pointe comme les World Models, où un agent apprend à "rêver" des futurs possibles dans un espace latent compact pour accélérer l'apprentissage par renforcement en robotique [34], [35].

3.2.2 . La révolution antagoniste : Les GAN (2014)

Pour répondre au manque de piqué et de réalisme des méthodes variationnelles, une rupture paradigmique a été introduite avec les Réseaux Antagonistes Génératifs (GAN). Cette approche délaisse l'estimation explicite de la densité de probabilité au profit d'une méthode implicite fondée sur la théorie des jeux. Le processus d'apprentissage est modélisé comme un jeu minimax à somme nulle entre un générateur qui tente de créer des données indiscernables du réel, et un discriminateur qui tente de distinguer les échantillons générés des données d'entraînement [36]. Comme le soulignent les travaux théoriques sur les modèles implicites, cette formulation permet au générateur d'apprendre des statistiques d'ordre supérieur souvent ignorées par les méthodes classiques, s'affranchissant des contraintes de vraisemblance [37]. Dans le cadre de la "traduction" d'environnement, les variantes conditionnelles telles que l'architecture Pix2Pix se sont imposées pour transformer une représentation sémantique en une image

photoréaliste, produisant des structures fines et des textures détaillées [38]. Si les GAN restent difficiles à stabiliser durant l'entraînement, ils ont démontré des capacités de généralisation spectaculaires au-delà de l'image statique. Des travaux comme tempoGAN ont par exemple appliqué ce principe à la mécanique des fluides, parvenant à super-résoudre des simulations volumétriques de fumée ou de liquide tout en garantissant une cohérence temporelle que les méthodes purement statistiques peinent à maintenir [39].

3.2.3 . La génération par raffinement : Les Modèles de Diffusion (2020)

La dernière vague d'innovation, qui définit une grande partie de l'état de l'art actuel, puise son inspiration dans la physique statistique hors équilibre. Les modèles de diffusion probabilistes proposent de construire la génération comme l'inversion d'un processus de destruction d'information. L'idée consiste à détruire progressivement la structure des données par l'ajout successif de bruit gaussien, puis d'entraîner un réseau de neurones à inverser ce processus temporel pour reconstruire la donnée originale étape par étape [40]. Ce concept a été porté à maturité avec les Denoising Diffusion Probabilistic Models (DDPM), qui offrent un compromis inédit : ils atteignent une qualité d'échantillonnage supérieure aux GAN tout en couvrant mieux la diversité de la distribution des données, évitant le problème d'effondrement de mode [41]. Bien que le processus itératif soit intrinsèquement lent, des méthodes d'échantillonnage accélérées (DDIM) ont rendu ces modèles exploitables en production [42]. Au-delà de la génération d'images 2D, ce paradigme est aujourd'hui le moteur de la génération de contenus pour les environnements virtuels. Des approches comme DreamFusion utilisent un modèle de diffusion 2D pré-entraîné pour optimiser une représentation volumétrique (NeRF), permettant de générer des objets 3D complets et cohérents à partir d'une simple description textuelle, ouvrant la voie à la création procédurale d'environnements physiques complexes [22].

3.2.4 . Le paradigme séquentiel et l'Autorégression

Enfin, une approche radicalement différente considère la génération comme une prédiction séquentielle discrète. Ce paradigme trouve ses racines dans les Réseaux de Neurones Récurrents (RNN), historiquement utilisés pour générer du texte ou des séries temporelles, mais limités par leur mémoire à court terme et leur séquentialité stricte [43]. La rupture fondamentale survient avec l'introduction de l'architecture Transformer et du mécanisme d'attention, qui permet de modéliser des dépendances à très long terme et de paralléliser le calcul [44]. L'évolution majeure de ce paradigme réside dans le concept de pré-entraînement génératif (GPT). Il a été démontré qu'un modèle entraîné massivement sur l'objectif simple de prédire le prochain élément d'une séquence acquiert une capacité de généralisation et de compréhension structurelle émergente [45]. Aujourd'hui, cette approche déborde largement du cadre du texte. Des architectures multimodales comme Gato [46] ou les Vision Transformers (ViT) [47] traitent les images ou les actions de contrôle robotique comme des séquences de tokens, unifiant ainsi la génération de contenu visuel et la prise de décision séquentielle au sein d'un même formalisme autorégressif. Cela positionne le traitement de séquence comme une méthode universelle pour la simulation, justifiant l'analyse détaillée des architectures séquentielles qui suivra.

3.2.5 . Ancrage dans la problématique

L'analyse du paysage de l'IA générative nous permet d'identifier les architectures les plus adaptées à la modélisation de notre simulateur de capteur. Si les modèles probabilistes explicites comme les VAE offrent une gestion intéressante de l'incertitude structurelle, leur tendance historique à produire des sorties lissées peut poser question quant à la fidélité des signaux radar, où la précision des paramètres fins tels que les fréquences est critique. Concernant les approches antagonistes (GAN), bien qu'elles soient performantes pour la synthèse d'images, leur adaptation directe à des séquences d'événements discrets paramétriques (les PDW) s'avère complexe, notamment pour gérer la causalité temporelle et la nature irrégulière du flux d'impulsions, sans compter leur instabilité d'entraînement connue. De même, si les modèles de diffusion définissent l'état de l'art actuel, leur processus de débruitage itératif est intrinsèquement coûteux en temps de calcul. Cette caractéristique entre potentiellement en conflit avec notre objectif premier d'accélération de la simulation, en plus de nécessiter une adaptation lourde pour traiter des vecteurs de paramètres physiques plutôt que des pixels.

Cette analyse invite à considérer le paradigme séquentiel auto-régressif. Contrairement au Traitement du Langage Naturel qui opère sur des vocabulaires finis, notre simulation numérique évolue dans un espace métrique continu : chaque PDW est défini par des coordonnées réelles (temps d'arrivée, fréquence, largeur). Dans ce contexte, l'acte génératif ne consiste pas à sélectionner un symbole parmi un dictionnaire, mais à prédire directement les valeurs d'un état dans un espace vectoriel continu \mathbb{R}^n . Bien que ce processus s'apparente mathématiquement à une régression multivariée, il conserve la nature intrinsèque de la génération : le modèle doit bâtir, étape par étape, une cohérence globale du signal temporel. Ainsi, la reconstruction de la séquence de PDW perçue à partir de la séquence émise se formule comme une tâche de traduction de signal continu. Cela motive l'orientation de notre étude vers les architectures spécialisées dans le traitement de séquence, capables de capturer les dépendances à long terme comme le pistage temporel, justifiant l'analyse approfondie des RNN et des Transformers qui suivra.

3.3 . Méthodes de traitement : Séquences et structures spatiales

3.3.1 . Typologie des données : De la causalité temporelle à la topologie spatiale

Avant d'aborder les architectures neuronales spécifiques, il est essentiel de définir formellement l'objet mathématique qu'elles manipulent. Dans l'apprentissage statistique classique, les données sont souvent supposées être indépendantes et identiquement distribuées (hypothèse i.i.d.). Le traitement de séquences et de structures spatiales rompt fondamentalement avec cette hypothèse en introduisant une structure de dépendance intrinsèque. Une donnée n'est pas un simple ensemble non ordonné, mais une collection indexée $X = \{x_1, x_2, \dots, x_N\}$ où l'indice porte une information topologique ou causale déterminante. La valeur d'un élément x_i n'a de sens que relativement à son contexte, c'est-à-dire son voisinage ou son historique. Cette propriété de dépendance locale, formalisée par les processus de Markov pour le temps ou les Champs de Markov pour l'espace, caractérise la nature de l'ensemble et impose des

contraintes spécifiques de modélisation que les architectures neuronales doivent satisfaire.

3.3.1.1 . La séquence et la causalité

Dans le cadre des séquences, la notion de proximité est dictée par la causalité : l'état présent est une fonction de l'histoire passée. C'est le fondement de la théorie de l'information de Shannon [48], où le langage est modélisé comme un processus stochastique discret. Dans cette vision, la probabilité d'apparition d'un symbole (lettre ou mot) dépend conditionnellement de la séquence des symboles précédents, définissant la notion d'entropie d'une source d'information. Cette logique s'applique identiquement aux séries temporelles continues. Des travaux sur les modèles ARIMA [49] ont montré qu'une observation à l'instant t est mathématiquement corrélée à ses prédecesseurs immédiats et aux termes d'erreur passés. Dans ce formalisme statistique, la séquence est définie par une dépendance directionnelle irréversible vers le futur.

3.3.1.2 . La donnée spatiale et la contiguïté

Le champ physique ou l'image, de leur côté, se définissent comme des grilles spatiales (multidimensionnelle ou simplement bidimensionnelle) où la notion d'ordre séquentiel disparaît au profit de la contiguïté topologique. Ici, un pixel ou une cellule n'a pas de "passé" ou de "futur", mais un voisinage omnidirectionnel. Cette structure, régie par la contiguïté spatiale locale, s'affranchit de la causalité temporelle au profit d'un voisinage omnidirectionnel propre aux Champs de Markov. Elle permet de fonder théoriquement l'usage des opérations de convolution, où la notion d'ordre chronologique est remplacée par celle de proximité euclidienne. Ce concept de donnée spatiale reste néanmoins proche de celui de séquence : les travaux sur PixelRNN [50] montrent qu'en traitant l'image comme une séquence autorégressive, où chaque pixel dépend de ceux situés "avant" lui (en haut et à gauche), on peut modéliser la distribution conjointe des pixels et générer des structures visuelles cohérentes.

3.3.1.3 . Universalité de la modélisation séquentielle

Finalement, le défi central des architectures de traitement que nous allons présenter (CNN, RNN, Transformer) est de modéliser cette fonction de dépendance conditionnelle $P(x_t|\text{Contexte})$. La nature de ce contexte varie selon le domaine : il sera un historique causal pour les séries temporelles et la génération de texte, alors qu'il est bidirectionnel et topologique pour l'image ou la compréhension sémantique globale. Cependant, l'objectif mathématique reste identique : capturer les corrélations à courte et longue portée qui structurent la donnée, transformant une collection de valeurs isolées en une entité cohérente.

3.3.2 . Réseaux de convolution

Bien que les données séquentielles soient intuitivement associées à une dimension temporelle linéaire, le traitement de l'information repose fondamentalement sur l'extraction de

motifs locaux et de relations de voisinage. C'est dans cette optique que les Réseaux de Neurones Convolutionnels (CNN) se positionnent comme une méthode incontournable. Initialement conçus pour la grille spatiale de l'image, ils formalisent une approche du traitement de séquence fondée sur la localité, l'invariance par translation et la hiérarchie des caractéristiques.

3.3.2.1 . Genèse et prédominance dans l'imagerie

L'histoire des réseaux de convolution est indissociable de la vision par ordinateur et de la volonté de s'affranchir des descripteurs manuels (SIFT [51], SURF [52] et HOG [53]). Inspirée par les travaux biologiques sur le cortex visuel, le premier modèle [54] a introduit les concepts fondateurs de champ récepteur local et de partage des poids pour la reconnaissance de caractères manuscrits. Cependant, c'est l'avènement d'AlexNet [55] qui a marqué le véritable point d'inflexion en démontrant la supériorité de l'apprentissage profond sur GPU pour l'extraction de caractéristiques. Cette percée a ouvert la voie à des architectures plus profondes et plus efficientes. Par exemple, l'architecture GoogLeNet [56] factorise les convolutions pour réduire le coût de calcul tout en augmentant la largeur du réseau, permettant de traiter des motifs à différentes échelles simultanément.

3.3.2.2 . Mécanisme d'interaction : Filtrage local et expansion hiérarchique

L'interaction fondamentale d'un réseau de convolution avec une séquence repose sur l'application répétée d'un opérateur de filtrage caractérisé par un noyau w de support fini. Contrairement à une couche dense qui apprendrait un poids spécifique pour chaque élément de la séquence globale, la convolution impose une contrainte de partage des poids qui nécessite que les données soient structurées dans un espace métrique régulier. En effet, l'opération suppose l'existence d'une fonction $p(\cdot)$ permettant d'associer à chaque élément x sa position sur une grille sous-jacente, qu'elle soit unidimensionnelle pour des données temporelles ou multidimensionnelle pour des données spatiales.

Mathématiquement, le noyau w est défini sur un support \mathcal{V} centré à l'origine. Ainsi, le noyau définit pour tout élément cible x_t un voisinage d'interaction \mathcal{V}_t , correspondant à la translation du support \mathcal{V} en la position $p(x_t)$. L'opération de convolution consiste alors à calculer une somme pondérée des éléments appartenant à ce voisinage, où les poids sont déterminés exclusivement par la position relative entre les éléments et le centre. La sortie h_t (avant activation) s'exprime par l'équation :

$$h_t = \sum_{x_j \in \mathcal{V}_t} w_{\Delta(x_j, x_t)} \cdot x_j + b$$

Dans cette expression, $\Delta(x_j, x_t) = p(x_j) - p(x_t)$ représente le vecteur de position relative du voisin x_j par rapport au centre x_t , b est un biais. Une conséquence directe de la structure en grille des données est que ce vecteur de différence correspond systématiquement à un n-uplet d'entiers (n étant la dimension de la grille). Cette propriété discrète est fondamentale pour l'implémentation neuronale : elle implique que la fonction w n'a pas besoin d'être modélisée comme une fonction continue, mais se réduit à un ensemble fini de paramètres scalaires (les poids du

filtre) qu'il suffit d'apprendre pour chaque décalage entier possible dans le support. Cette formulation garantit l'invariance par translation, assurant que le même motif de poids est appliqué uniformément sur toute la structure. Par ailleurs, l'application de ce voisinage aux bornes d'une grille finie nécessite une gestion des effets de bord, typiquement résolue par l'ajout de valeurs nulles (zero-padding) en périphérie afin de conserver la dimension de la séquence traitée.

Ce mécanisme permet l'extraction robuste de motifs locaux, mais la compréhension de la structure globale de la séquence émerge de la composition hiérarchique de ces opérations. L'illustration 3.1 permet de visualiser comment l'empilement de couches induit une expansion mathématique du champ récepteur. Considérons une première couche définie par un filtre w de taille 3. Pour un instant t , ce filtre induit un voisinage immédiat. L'équation locale est, en notant ϕ la fonction d'activation :

$$\begin{aligned} h_t^{(1)} &= w_1 x_{t-1} + w_2 x_t + w_3 x_{t+1} \\ x_t^{(1)} &= \phi(h_t^{(1)}) \end{aligned}$$

La sortie $x_t^{(1)}$ est une fonction des entrées x_{t-1} à x_{t+1} . Lorsqu'une seconde couche définie par un filtre v de même support est appliquée sur cette représentation intermédiaire, elle opère selon le même principe d'invariance en translation :

$$\begin{aligned} h_t^{(2)} &= v_1 x_{t-1}^{(1)} + v_2 x_t^{(1)} + v_3 x_{t+1}^{(1)} \\ x_t^{(2)} &= \phi(h_t^{(2)}) \end{aligned}$$

La sortie $x_t^{(2)}$ est une fonction des entrées x_{t-2} à x_{t+2} . Ainsi, par simple composition algébrique, l'horizon d'interaction s'est étendu de 3 éléments (couche 1) à 5 éléments (couche 2). La profondeur du réseau agit donc comme un multiplicateur mécanique de l'horizon d'interaction, permettant de reconstruire des dépendances causales à longue portée à partir de règles de construction strictement locales et invariantes.

Au-delà de l'expansion de l'horizon d'interaction, la géométrie du support de convolution détermine la nature causale ou non du traitement, une caractéristique nécessaire pour la modélisation de systèmes dynamiques. La partie gauche de la figure 3.2 la configuration standard, dite convolution centrée. Pour calculer un élément de sortie y_4 à l'instant $t = 4$, le champ récepteur effectif (cône violet) agrège les informations d'un voisinage symétrique de l'entrée x , de x_2 à x_6 . Cette approche est naturelle pour l'analyse de données statiques (comme une image) ou le traitement de séquences complètes a posteriori, mais n'est pas adapté à la modélisation d'un système dynamique. Par exemple dans le cas de filtrage en ligne, le système ne peut physiquement pas accéder aux mesures futures pour débruiter les données du présent. Pour adapter l'architecture à ces contraintes temporelles strictes, on recourt à la convolution causale. Cette variante consiste à décaler le support du filtre de manière à ce que le voisinage d'interaction au temps t ne contienne aucun indice supérieur à t . L'exemple illustré sur la partie droite de la figure 3.2 assure que le cône d'influence de la sortie y_5 est alors strictement orienté vers le passé (x_1 à x_5). Dans cette configuration, le réseau conserve sa capacité d'extraction de motifs et de

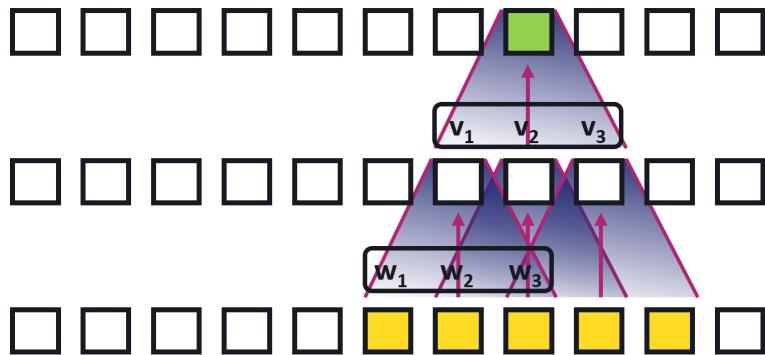


Figure 3.1 – Illustration d'une convolution 1D standard et de l'expansion hiérarchique du champ récepteur

parallélisation, mais adopte une topologie d'interaction compatible avec la physique, simulant le comportement d'un système causal sans recourir à la mémoire récurrente.

3.3.2.3 . La convolution dans l'image : Une séquence spatiale 2D

Dans le contexte de l'image, la séquence est bidimensionnelle et le CNN y opère une extraction hiérarchique. Les premières couches détectent des primitives simples comme des bords ou des textures, qui sont ensuite combinées pour former des motifs sémantiques complexes. Cette capacité d'abstraction a été poussée à son paroxysme par l'architecture VGG [57], qui a standardisé l'usage de filtres de très petite taille (3×3) empilés en grande profondeur. Les auteurs ont démontré qu'une séquence de petites convolutions est plus efficace pour capturer des non-linéarités complexes qu'une seule grande convolution. Cependant, l'augmentation de la profondeur a engendré des problèmes de disparition du gradient, résolus avec ResNet [58]. L'introduction de connexions résiduelles a permis d'entraîner des réseaux dépassant la centaine de couches, essentiels pour capturer les dépendances à très longue portée dans des images haute résolution. Pour les tâches de "traduction" d'image vers image, cruciales en simulation (par exemple, passer d'une carte de densité à un champ de pression), il est impératif de ne pas perdre l'information spatiale lors de la compression. L'architecture U-Net [59], initialement pour la segmentation biomédicale combine un chemin de contraction et un chemin d'expansion reliés par des connexions latérales (skip connections). Cette structure permet de générer une sortie de même résolution que l'entrée en fusionnant le contexte sémantique global et les détails locaux. Cette architecture est aujourd'hui une référence pour les modèles de substitution en physique. D'autres variantes comme DenseNet [60] ont poussé cette logique plus loin en connectant chaque couche à toutes les suivantes pour maximiser le flux d'information, bien que cela se fasse au prix d'une consommation mémoire accrue.

3.3.2.4 . La convolution dans les séquences 1D (Texte, Audio, Séries Temporelles)

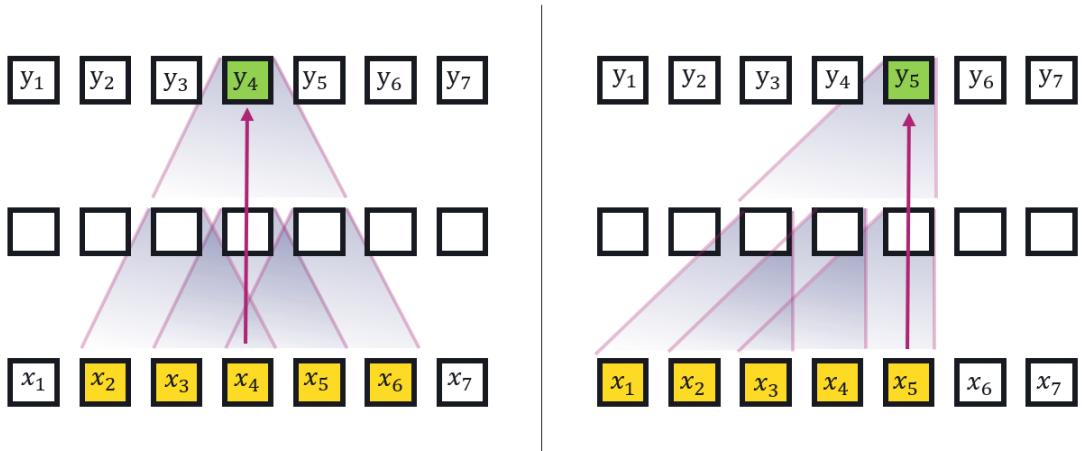


Figure 3.2 – Impact de la topologie du support de convolution sur la causalité temporelle : approche centrée (gauche) et approche causale (droite)

Bien que souvent associés à l'image, les CNN se sont révélés extrêmement performants pour traiter des séquences unidimensionnelles, surpassant parfois les réseaux récurrents grâce à leur capacité de parallélisation. Dans le traitement du signal audio, l'architecture WaveNet [61] a marqué une rupture en utilisant des convolutions causales dilatées. Ce mécanisme permet au champ récepteur du réseau de croître exponentiellement avec la profondeur sans augmenter le nombre de paramètres, capturant ainsi des dépendances temporelles sur des milliers de pas de temps, ce qui est impossible pour un RNN standard REF. Dans le domaine du traitement du langage naturel (NLP), cette logique a été appliquée avec succès à la traduction automatique. L'architecture ConvS2S [62] entièrement convolutionnelle pour la séquence à séquence, utilise des mécanismes d'attention multi-pas pour pondérer l'importance des mots sources. De même, ByteNet [63], réalise la traduction en temps linéaire en empilant des convolutions dilatées. Ces travaux ont démontré que l'induction de biais locaux propres aux convolutions est pertinente pour la syntaxe et la sémantique locale. Cette approche a été généralisée aux séries temporelles génériques sous le nom de Temporal Convolutional Networks (TCN) [64]. L'étude comparative démontre que sur une vaste gamme de tâches séquentielles, comme la prédiction de charge énergétique ou la modélisation de séquences symboliques, les TCN surpassent souvent les réseaux récurrents (LSTM/GRU) REF tout en offrant une stabilité d'entraînement supérieure. Une étude récente [65] prolonge ce constat en suggérant que des architectures convolutionnelles modernes pré-entraînées peuvent rivaliser avec les Transformers sur certaines tâches textuelles, soulignant la pertinence continue de ce paradigme.

3.3.2.5 . Généralisation : De la grille volumétrique aux topologies irrégulières

Le principe de convolution, initialement restreint aux images planes, a fait l'objet d'extensions successives pour traiter des structures de données de plus en plus complexes. Une pre-

mière étape de généralisation concerne les données volumétriques (3D) et spatio-temporelles, qui conservent toutefois une structure de grille régulière (euclidienne). Pour l'analyse de vidéos, C3D [66] déploie des filtres tridimensionnels (x, y, t) capturant conjointement l'espace et le mouvement. De même, dans le domaine médical, l'architecture V-Net [67] étend le principe du U-Net à la 3D en opérant sur des voxels. Bien que performantes, ces méthodes restent contraintes par la régularité de la grille : elles imposent une voxelisation des données qui induit une complexité cubique et une perte de résolution, les rendant inadaptées aux géométries éparses.

La généralisation la plus significative pour la simulation scientifique concerne donc les données non-euclidiennes, intrinsèquement irrégulières. Dans une simulation lagrangienne ou un système moléculaire, le "voisinage" n'est plus défini par une position dans une matrice, mais par la topologie d'un graphe. Les Graph Convolutional Networks (GCN) [68] redéfinissent alors la convolution comme une agrégation spectrale ou spatiale des caractéristiques des noeuds connectés, indépendamment de leur disposition géométrique absolue. Cette approche a été enrichie par GraphSAGE [69], proposant une convolution inductive capable de généraliser à des noeuds invisibles durant l'entraînement. Enfin, pour traiter des nuages de points 3D bruts sans passer par la case voxelisation, des architectures comme PointNet++ [70] appliquent des opérations hiérarchiques directement sur des ensembles continus, comblant le fossé entre convolution discrète et géométrie continue.

3.3.3 . Réseaux de neurones récurrents et Espaces d'Etats (RNN et SSM)

Si les réseaux de convolution abordent la séquence par une fenêtre glissante locale, une autre famille d'architectures adopte une approche intrinsèquement temporelle : la modélisation récursive. Qu'il s'agisse des Réseaux de Neurones Récurrents (RNN) historiques ou des récents Modèles d'Espaces d'États (SSM), le principe fondateur reste la persistance de l'information. Le modèle maintient un état caché interne h_t qui agit comme une mémoire compressée de tout l'historique passé, mise à jour à chaque nouvelle observation. Cette formulation est particulièrement naturelle pour la simulation physique, car elle mime la dynamique des systèmes causaux où l'état futur dépend de l'état présent et des forces appliquées.

3.3.3.1 . Genèse et mécanismes d'interaction : De la boucle simple aux portes logiques

L'histoire de cette approche débute avec les RNN classiques [71] qui introduisent une boucle de rétroaction permettant au réseau de maintenir une trace du contexte temporel. Cependant, bien que ces réseaux parviennent à générer des séquences continues complexes comme de l'écriture manuscrite, ils souffrent d'une instabilité critique lors de l'entraînement : le problème de la disparition ou de l'explosion du gradient [43]. Sur de longues séquences, le signal d'erreur se dilue, empêchant l'apprentissage des causes lointaines d'un événement. Pour y remédier, le LSTM (Long Short-Term Memory) [72] propose des "cellules" mémoires protégées par des portes logiques, et peut choisir de retenir ou d'effacer une information sur des milliers de pas de temps. Cette capacité a été affinée par l'introduction du GRU [73], [74], une variante plus économique.

3.3.3.2 . Mécanisme d'interaction : Récurrence et Mémoire d'État

L'interaction fondamentale des architectures récurrentes (RNN) et des modèles d'espaces d'états (SSM) avec la séquence repose sur un principe de persistance de l'information, radicalement différent de la localité spatiale des convolutions. Au lieu d'agréger un voisinage statique, ces modèles introduisent une variable latente dynamique, l'état caché h_t , qui agit comme une mémoire compressée de l'historique causal. Mathématiquement, la modélisation d'une telle trajectoire dynamique se formalise comme un problème de valeur initiale. Elle nécessite impérativement de définir une condition d'ancrage h_0 avant de décrire la loi d'évolution f . Pour une séquence d'entrée x , le système est ainsi entièrement décrit par le couple d'équations suivant :

$$\begin{cases} h_0 = \lambda \\ h_t = f(h_{t-1}, x_t; \theta) \quad \forall t \geq 1 \end{cases}$$

La première équation initialise l'état de la mémoire avant toute observation. Si la convention standard pose un vecteur nul ($\lambda = \vec{0}$), supposant une mémoire vierge, l'apprentissage machine laisse la possibilité au réseau d'optimiser l'encodage de cette mémoire vierge sous la forme d'un paramètre appris λ . La seconde équation décrit la mise à jour récursive où f est une fonction de transition paramétrée par θ . Cette formulation implique que h_t ne dépend pas seulement de l'entrée locale x_t , mais indirectement de toute la trajectoire passée $\{x_0, \dots, x_t\}$ accumulée dans h_{t-1} .

En pratique, cette dynamique de mise à jour est régie par un ensemble de paramètres apprenables qui sont partagés sur des parties de la séquence, garantissant l'invariance du traitement par tâche conceptuelle (genre encodage d'une phrase puis autre matrice pour décodage et traduction). Historiquement, les premières architectures (genre RNN avant LSTM et GRU etc) encodaient simplement ces paramètres sous la forme d'une matrice W_{ih} (Input-to-Hidden) qui projette l'entrée courante dans l'espace latent, d'une matrice W_{hh} (Hidden-to-Hidden) qui contrôle l'évolution de la mémoire interne, et d'un vecteur de biais b . En notant ϕ la fonction d'activation non-linéaire, l'équation de propagation de récurrence s'écrit pour notre exemple :

$$h_{t+1} = \phi(W_{ih}x_t + W_{hh}h_t + b)$$

L'illustration 3.3 permet de visualiser la propagation de la dépendance temporelle à travers une architecture récurrente élémentaire, souvent qualifiée de RNN d'Elman [71]. Dans ce modèle simplifié (ici représenté sans biais pour la clarté), les informations sont transportées d'état caché en état caché à travers la matrice de poids W_1 et d'entrée à l'état caché à travers W_2 . On illustre en bleu le cône de perception de l'état caché h_3 (représenté par le carré vert) intégrant par récurrence les informations des observations passées x_1 , x_2 et x_3 , créant un lien causal ininterrompu.

$$\begin{aligned} h_3 &= \phi(W_2x_3 + W_1h_2) \\ &= \phi(W_2x_3 + W_1\phi(W_2x_2 + W_1h_1)) \\ &= \phi(W_2x_2 + W_1\phi(W_2x_2 + W_1\phi(W_2x_1 + W_1\lambda))) \end{aligned}$$

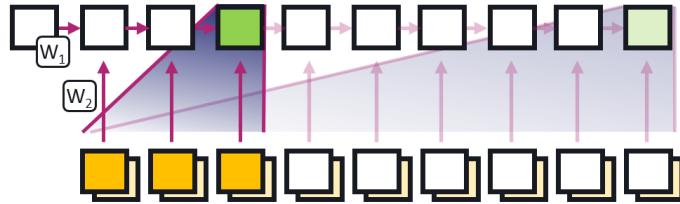


Figure 3.3 – Mécanisme fondamental de la récurrence et propagation de l'état mémoire

Cette formulation met en évidence la différence fondamentale avec la convolution : alors que le champ récepteur d'un CNN s'élargit progressivement par empilement de couches, ici le cône d'influence (en bleu) s'étend horizontalement vers le passé jusqu'à la première entrée, capturant théoriquement la totalité de l'historique causal disponible.

En fonction de la fonction accomplie par l'architecture récurrente, les états cachés sont exploités différemment. Le mode "Flux à Flux" (ou Many-to-Many synchronisé), illustré à gauche dans la figure 3.4, aligne la production de la sortie sur la réception de l'entrée. À chaque pas de temps t , l'état caché h_t est utilisé immédiatement pour prédire une sortie y_t . Cette configuration, où la causalité est stricte et le délai minimal, est caractéristique des systèmes de filtrage en ligne ou de contrôle, où la réaction doit être instantanée. Le mode "Séquence vers Séquence" (Seq2Seq), à droite dans la figure 3.4, nécessite d'opérer en deux phases distinctes. La phase d'encodage représenté en violet, ingurgite toutes les informations et les condense dans l'état caché h_5 . Les états cachés h_1 à h_5 ne sont utilisés que pour encoder l'information du passé, et ne sont exploités que pour transmettre cette information à l'avenir. La phase de décodage récupère ce contexte passé pour générer la séquence de sortie. Les états cachés de h_6 à h_8 sont à la fois utilisés pour transmettre et renseigner l'information de mémoire mais aussi pour prédire les nouveaux éléments, qui sont réinjectés en entrée (boucle autorégressive). Ce mécanisme permet de transformer une séquence en une autre de longueur différente et de modéliser des dépendances non-monotones, mais impose que toute l'information pertinente soit compressée dans un goulot d'étranglement. C'est cette distinction topologique, plus que la nature des données, qui différencie fondamentalement l'usage des RNN pour le suivi de signal (mode flux) de leur usage pour la traduction (mode Seq2Seq).

3.3.3 . Renouveau architectural : Les Modèles d'Espaces d'Etats (SSM)

Malgré leur robustesse, les LSTM conservent une limitation structurelle majeure : leur traitement séquentiel interdit la parallélisation sur GPU. C'est pour lever ce verrou qu'une nouvelle classe de modèles a émergé : les State Space Models (SSM). Ces modèles puisent leur origine théorique dans le papier HiPPO [75], qui formalise mathématiquement comment compresser optimalement une histoire continue dans un vecteur de taille fixe via des projections polynomiales orthogonales. Cette base a permis de développer S4 (Structured State Space sequence model) [76], capable de modéliser des dépendances sur plus de 10 000 pas de temps en

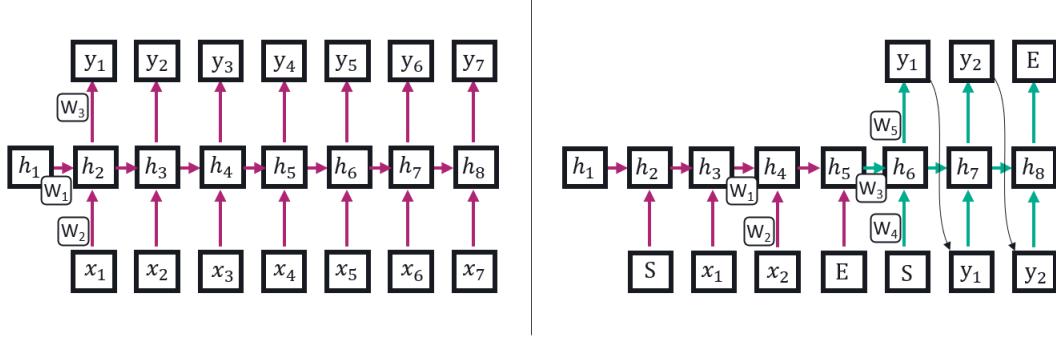


Figure 3.4 – Topologies d’application des architectures récurrentes : traitement de flux (gauche) et traduction globale (droite)

résolvant une équation différentielle continue discrétisée. Cependant, les premiers SSM souffraient d’une rigidité dynamique, peinant à sélectionner l’information pertinente en fonction du contexte (“Content-based selection”). Cette limite a été adressée par l’architecture Mamba [77]. En rendant les matrices d’état dépendantes de l’entrée, Mamba atteint des performances comparables aux premiers Transformers REF tout en conservant une complexité linéaire. Toutefois, ces modèles restent délicats à stabiliser sur des dynamiques hautement chaotiques où la discréttisation numérique peut introduire des dérives.

3.3.3.4 . Application au texte : L’ère du Sequence-to-Sequence et de la Traduction

Dans le traitement du langage, l’approche récurrente a connu son apogée avec le paradigme Seq2Seq [78]. En utilisant deux LSTM (un encodeur et un décodeur), cette architecture a permis de traiter des séquences de longueurs variables. Cette avancée a transformé l’industrie de la traduction avec le déploiement du Google’s Neural Machine Translation System (GNMT) [79] en 2016, réduisant les erreurs de traduction de près de 60 % par rapport aux systèmes statistiques. Au-delà de la traduction, ce paradigme a permis des avancées dans la modélisation prédictive de parcours complexes, comme l’illustre par le modèle Doctor AI [80]. Ce modèle utilise des RNN pour prédire les futurs diagnostics médicaux et la durée avant la prochaine visite à partir de l’historique clinique des patients, démontrant la capacité des RNN à capturer des dynamiques temporelles irrégulières et multivariées dans des données réelles bruitées.

3.3.3.5 . Application aux systèmes temporels, physiques et créatifs

Au-delà du texte, les RNN se sont imposés comme l’outil naturel pour la modélisation de systèmes dynamiques continus, un domaine crucial pour la simulation. Une étude sur la prévision de systèmes chaotiques [81] a mis en avant que les LSTM pouvaient apprendre la dynamique de l’attracteur de Lorenz ou de l’équation de Kuramoto-Sivashinsky mieux que les modèles physiques simplifiés, en capturant les propriétés non-linéaires de l’évolution temporelle à court terme. Cette capacité à modéliser le chaos déterministe fait des RNN des candidats sérieux

pour accélérer les simulations de mécanique des fluides turbulents. Dans l'industrie, cette robustesse est exploitée pour la prévision probabiliste avec DeepAR [82], utilisé par Amazon pour sa chaîne logistique. Ce modèle apprend une distribution de probabilité future à chaque pas de temps, permettant de quantifier l'incertitude via des simulations de Monte Carlo. Enfin, la capacité "générationne constructive" des RNN a été pionnière dans la création artistique. Le modèle Performance RNN [83], développé par Google Magenta, a montré qu'un LSTM pouvait générer des performances de piano expressives (avec nuances de vitesse et de timing) en traitant la musique non pas comme une partition rigide, mais comme une séquence temporelle continue d'événements, prouvant que les RNN peuvent capturer des structures hiérarchiques globales (phrasé musical) tout en gérant des détails micro-temporels.

3.3.4 . L'architecture Transformer

Si les réseaux récurrents ont introduit la mémoire et les réseaux convolutionnels la localité, l'architecture Transformer a proposé un changement de paradigme radical en postulant que l'interaction entre les éléments d'une séquence doit être modélisée par une relation directe de contenu à contenu, et non par une contrainte de proximité spatiale ou temporelle. Cette architecture, devenue l'épine dorsale de l'IA générative moderne, repose sur le mécanisme d'attention [44].

3.3.4.1 . Genèse : Du goulot d'étranglement récurrent à l'Attention pure

L'émergence du Transformer est le fruit d'une lente maturation visant à résoudre le goulot d'étranglement des architectures Encodeur-Décodeur récurrentes (RNN). Dans le paradigme Seq2Seq classique [78], toute l'information de la phrase source devait être compressée dans un unique vecteur de contexte de taille fixe, entraînant une perte d'information critique sur les longues séquences. Une première solution [84] introduit un mécanisme d'attention additive permettant au décodeur de "chercher" (search) et d'aligner (align) les parties pertinentes de la phrase source à chaque étape de la génération. Ici, l'attention n'était encore qu'un module auxiliaire greffé sur des RNN. Une seconde étape conceptuelle fut franchie avec les Pointer Networks [85] où le réseau de neurones apprend à résoudre des problèmes combinatoires en utilisant l'attention comme un pointeur pour sélectionner des éléments de l'entrée comme sortie. Cela a ancré l'idée que le mécanisme de sélection basé sur le contenu ("Content-based addressing") était suffisamment puissant pour structurer la sortie. La rupture définitive survient avec l'article Attention Is All You Need [44]. Les auteurs ont démontré que la récurrence, jugée jusqu'alors indispensable pour encoder l'ordre séquentiel, était en réalité superflue et limitante pour la parallélisation. Ne conserver que l'attention a permis un traitement massivement parallèle et une réduction le chemin maximal de propagation de l'information entre deux mots quelconques à une constante $O(1)$, contre $O(N)$ pour un RNN. Cette réduction drastique de la distance topologique facilite considérablement le flux de gradient et l'apprentissage des dépendances à long terme.

Contrairement aux CNN ou RNN qui se définissent par une opération élémentaire, le Transformer se définit d'abord comme une architecture systémique modulaire. Son fonctionnement repose sur le concept de "Scaled Dot-Product Attention", que l'on peut apprécier par une analogie avec la recherche d'information. Chaque élément de la séquence émet une Requête (*Query*, Q), une Clé (*Key*, K) et une Valeur (*Value*, V). Le mécanisme calcule la pertinence entre la requête d'un élément et les clés de tous les autres pour déterminer un poids d'attention. Ce poids pondère ensuite l'agrégation des valeurs, permettant de synthétiser un nouveau vecteur de contexte. Nous décrivons l'architecture en commençant par le flux macroscopique de l'information, avant d'analyser la composition interne et spécifiquement le mécanisme d'attention.

3.3.4.2 . Architecture Macroscopique : Le paradigme Encodeur-Décodeur

Dans sa configuration canonique pour la traduction automatique [44], l'architecture adopte une structure bipartite illustrée par la figure 3.5. Son objectif est de transformer une séquence source $(x_i)_{1 \leq i \leq N}$ en une séquence cible $(y_i)_{1 \leq i \leq M}$. Pour la clarté de l'analyse, nous formalisons ici les représentations intermédiaires : nous noterons $(h_i)_{1 \leq i \leq N}$ la séquence latente produite par l'encodeur, et $(z_i)_{1 \leq i \leq M}$ la séquence de vecteurs de sortie du décodeur. Par convention, chaque vecteur z_k est la représentation destinée à être projetée pour prédire le symbole cible y_k . Cette architecture se justifie par deux apports majeurs : l'enrichissement sémantique via l'attention et l'accélération de l'apprentissage. En effet, contrairement aux réseaux récurrents (RNN) contraints à un traitement itératif, le Transformer permet de calculer l'intégralité de la séquence de prédiction $(z_i)_i$ en une unique passe parallèle.

Le flux de traitement débute par l'encodeur qui traite la source $(x_i)_i$. Grâce au mécanisme d'auto-attention, chaque élément agrège l'information du contexte global pour produire la séquence latente $(h_i)_i$ à haute teneur sémantique. Le décodeur exploite ensuite cette représentation $(h_i)_i$ pour générer la séquence $(z_i)_i$. C'est ici qu'intervient la distinction critique entre entraînement et inférence. Pour garantir que le modèle puisse générer des phrases mot après mot en inférence (où y_k n'est pas encore connu), l'entraînement doit simuler cette causalité tout en restant parallèle. On applique donc un décalage des entrées : pour produire le vecteur de prédiction z_k (visant le symbole y_k), le décodeur reçoit en entrée les symboles précédents (S, y_1, \dots, y_{k-1}) . Ce décalage aligne mécaniquement l'information passée face à l'objectif de prédiction présent z_k .

Ce principe est illustré par la figure 3.5 à travers le cône de perception du vecteur z_3 . Puisque z_3 a pour fonction de prédire y_3 , il a accès via l'attention croisée à la totalité de la source $(x_i)_i$, mais son accès à la séquence cible est strictement contraint aux antécédents S, y_1 et y_2 via le masquage. Ainsi, chaque vecteur z_k est construit en intégrant tout le contexte source $(h_i)_{1 \leq i \leq N}$ et le passé cible $(y_i)_{1 \leq i \leq k-1}$, sans jamais violer la causalité temporelle en accédant à y_k ou à ses successeurs. Une fois la séquence $(z_i)_i$ calculée, une simple projection linéaire suivie d'un Softmax permet d'obtenir les probabilités de chaque symbole y_k .

Cette structure macroscopique ne décrit pas un simple passage de couches, mais un em-

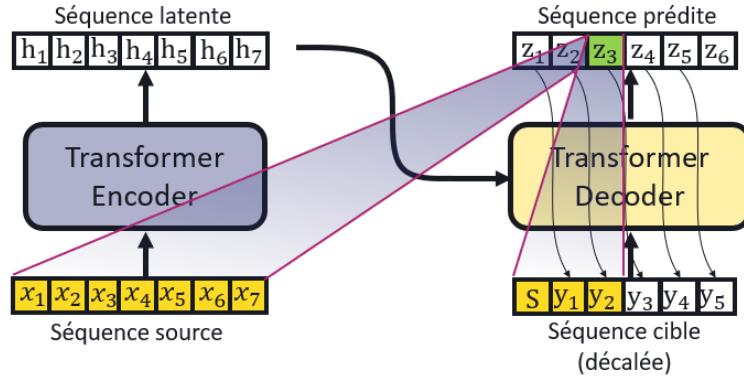


Figure 3.5 – Flux d'information macroscopique dans l'architecture Transformer Encodeur-Décodeur

pilement profond. Comme le montre la figure 3.6, l'Encodeur et le Décodeur sont constitués respectivement d'une pile de N blocs identiques. C'est la répétition de ces blocs qui confère au modèle sa capacité d'abstraction. Entrons dans le détail de ces deux blocs.

3.3.4.3 . L'Encodeur

L'illustration 3.7 détaille l'architecture interne d'un bloc d'encodage. Chaque bloc est conçu pour transformer les représentations vectorielles entrantes en une version plus contextualisée. Il s'articule autour de deux sous-modules séquentiels. Le premier est l'Attention Multi-Têtes, où les rôles de Requête, Clé et Valeur (Q, K, V) sont tous issus de la sortie du bloc précédent. Cela permet à chaque position de capturer des relations avec toutes les autres positions de la séquence. Le second est un réseau de neurones dense (Feed-Forward Network - FFN) appliqué indépendamment à chaque position. Pour permettre l'entraînement de réseaux profonds, chaque sous-module est encapsulé par une connexion résiduelle [58] suivie d'une normalisation de couche (Layer Norm) [86], assurant stabilité et une bonne propagation du gradient dans toutes les couches. Enfin, l'architecture étant invariante par permutation, l'ajout dès l'entrée d'un Encodage Positionnel est indispensable pour injecter la topologie temporelle ou spatiale dans les représentations vectorielles.

3.3.4.4 . Le Décodeur

L'architecture du Décodeur, présentée en figure 3.8, adapte la structure de l'encodeur aux contraintes de la génération. Chaque bloc intègre trois sous-modules au lieu de deux. Le premier est une Attention Multi-Têtes Masquée. Ce masquage est l'implémentation technique de la causalité : il force les poids d'attention à zéro pour toutes les positions futures, interdisant au modèle de d'accéder aux mots (y_i) $_{k \leq i \leq M}$ lors du calcul de la représentation de z_k . Le second

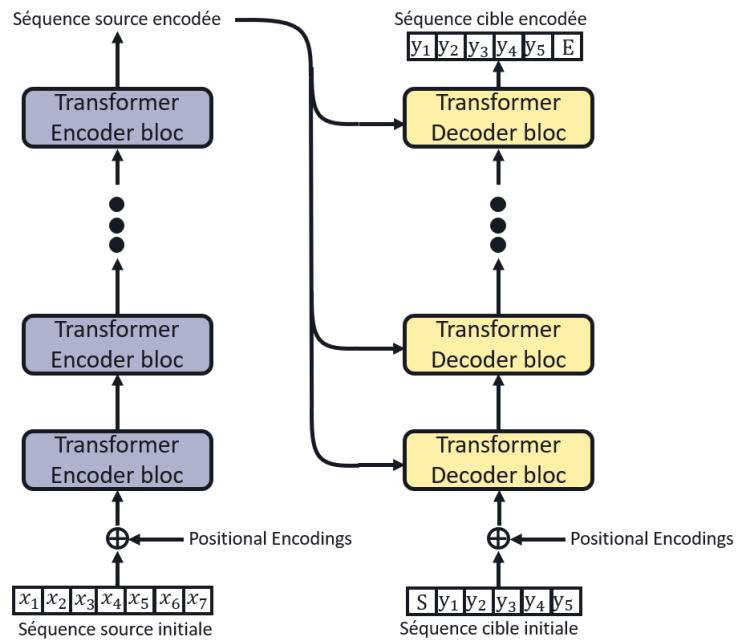


Figure 3.6 – Architecture Transformer complète : empilement des blocs

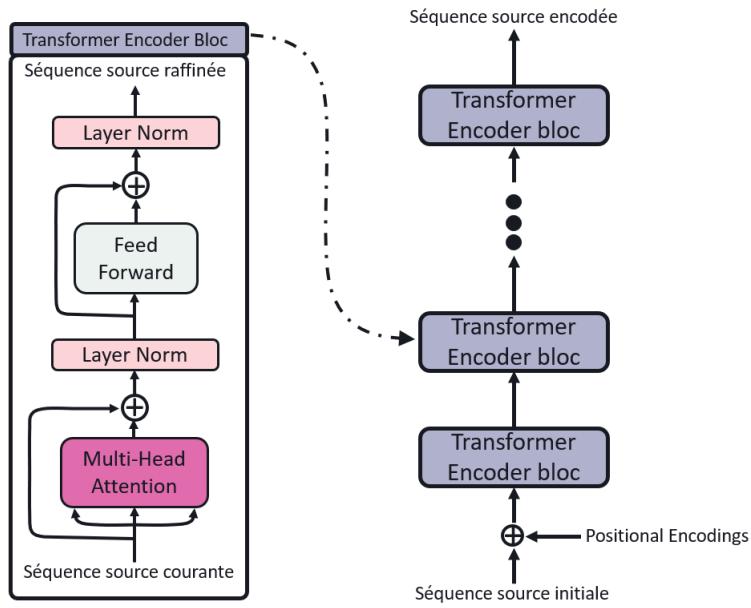


Figure 3.7 – Architecture interne du bloc Encodeur

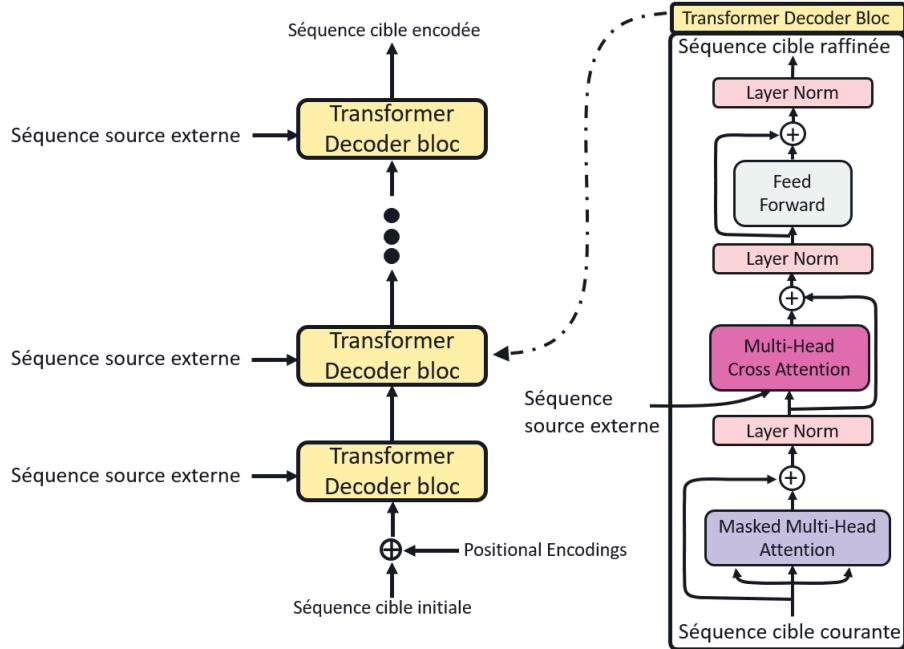


Figure 3.8 – Architecture interne du bloc Décodeur

module est l'Attention Croisé : les Requêtes (Q) proviennent de la chaîne de décodeur (ce que l'on cherche à enrichir), tandis que les Clés (K) et Valeurs (V) proviennent du contexte source (h_i). C'est ce pont qui permet à la génération d'être conditionnée par la source. Enfin, on retrouve le réseau FFN classique. Comme pour l'encodeur, l'emploi systématique de connexions résiduelles et de normalisation de couche assure la stabilité du gradient à travers la profondeur du réseau.

3.3.4.5 . Micro-architecture : Le mécanisme d'Attention et ses variantes

Au cœur de ces blocs macroscopiques réside le véritable moteur de l'interaction : le mécanisme d'Attention. Contrairement aux architectures précédentes qui traitent la séquence par voisinage spatial ou récursivité temporelle, le Transformer repose sur un mécanisme d'interaction directe et globale. Il est basé sur un principe de recherche d'information entre une séquence de requête $\mathcal{Q} = (q_i)_{i \in \mathbb{I}}$ et une séquence valeur $\mathcal{V} = (v_j)_{j \in \mathbb{J}}$ caractérisée par une séquence de clé $\mathcal{K} = (k_j)_{j \in \mathbb{J}}$. Les ensembles d'indications de ces séquences \mathbb{I} et \mathbb{J} ne sont pas nécessairement identiques. On notera aussi par commodité $\mathcal{KV} = ((k_j, v_j))_{j \in \mathbb{J}}$.

En pratique, n'importe quelle séquence $(x_i)_{i \in \mathbb{X}}$, indiqué par \mathbb{X} , peut être projetée dans un espace de représentation de requêtes $\mathcal{Q} = (Q(x_i))_{i \in \mathbb{X}}$, ou/et les espaces de représentation de clés et valeurs $\mathcal{K} = (K(x_i))_{i \in \mathbb{X}}$ et $\mathcal{V} = (V(x_i))_{i \in \mathbb{X}}$. Pour toute requête donnée $q \in \mathcal{Q}$ et toute clé candidate $k \in \mathcal{K}$, le score d'attention défini par le produit scalaire normalisé mesure

la pertinence de l'association :

$$a(q, k) = \frac{\langle q, k \rangle}{\sqrt{d_{att}}} \quad (3.1)$$

À requête q fixée, la séquence de score d'attention $(a(q, k))_{k \in \mathcal{K}}$ est transformée par une fonction *softmax* créant une pondération normalisée :

$$p(q, k, \mathcal{K}) = \frac{\exp(a(q, k))}{\sum_{k' \in \mathcal{K}} \exp(a(q, k'))} \quad (3.2)$$

L'opérateur d'attention, noté A , agrège alors l'information en calculant la somme des valeurs \mathcal{V} pondérées par ces poids :

$$A(q, \mathcal{KV}) = \sum_{(k, v) \in \mathcal{KV}} p(q, k, \mathcal{K})v \quad (3.3)$$

Cette formulation mathématique traduit le concept intuitif : pour un couple $(k, v) \in \mathcal{KV}$, plus le score $a(q, k)$ est élevé, plus la contribution de la valeur v est grande dans la représentation contextuelle de la requête $A(q, \mathcal{KV})$.

Le facteur déterminant pour l'expressivité du mécanisme est l'aspect multi-tête : au lieu d'avoir un triplet de fonctions de projection Q, K, V nous disposons de h triplets $(Q_l, K_l, V_l)_{1 \leq l \leq h}$. Cette diversité permet la capture d'une plus grande variété d'interaction en projetant les séquences dans h espaces de comparaisons différents. N'importe quelle séquence traitée $(x_i)_{i \in \mathbb{X}}$, sera alors projetée simultanément dans h espaces de représentation de requêtes

$$\mathcal{Q}_l = (Q_l(x_i))_{i \in \mathbb{X}}, \quad 1 \leq l \leq h \quad (3.4)$$

et/ou dans h espaces de représentation des clés et valeurs

$$\mathcal{K}_l = (K_l(x_i))_{i \in \mathbb{X}}, \quad \mathcal{V}_l = (V_l(x_i))_{i \in \mathbb{X}}, \quad 1 \leq l \leq h \quad (3.5)$$

En pratique, on dispose donc de h triplets de séquences $(\mathcal{Q}_l, \mathcal{K}_l, \mathcal{V}_l)_{1 \leq l \leq h}$. Le calcul de chaque tête d'attention est réalisé en appliquant, pour l variant de 1 à h , l'opérateur d'attention 3.3 aux séquences $\mathcal{Q}_l, \mathcal{K}_l$ et \mathcal{V}_l .

$$A(q, \mathcal{KV}_l) = \sum_{(k, v) \in \mathcal{KV}_l} \frac{\exp(a(q, k))}{\sum_{k' \in \mathcal{K}_l} \exp(a(q, k'))} v$$

Enfin, les sorties des h têtes sont concaténées (opérateur \oplus) pour former un vecteur de dimension $h \times d_{att}$. Ce vecteur est reprojeté par une fonction O , définissant ainsi l'opérateur d'attention multi-tête (MHA – Multi-Head Attention) applicable à tout élément x de la séquence formant les requêtes 3.4.

$$MHA(x) = O(A(Q_1(x), \mathcal{KV}_1) \oplus A(Q_2(x), \mathcal{KV}_2) \oplus \cdots \oplus A(Q_h(x), \mathcal{KV}_h)) \quad (3.6)$$

Cette formulation générique connaît plusieurs variantes fondamentales qui se distinguent par la nature des interactions qu'elles sont sensées capturer.

Premièrement, lorsque l'objectif est d'enrichir une séquence cible $(x_i)_{i \in \mathbb{I}}$ par le contexte d'une séquence source distincte $(y_j)_{j \in \mathbb{J}}$, on assigne les rôles de manière asymétrique. Pour toute tête $l \in \{1, \dots, h\}$, l'ensemble des requêtes est généré par la cible, soit $\mathcal{Q}_l = \{Q_l(x_i)\}_{i \in \mathbb{I}}$, tandis que l'ensemble des paires clé-valeur est généré par la source, soit $\mathcal{KV}_l = \{(K_l(y_j), V_l(y_j))\}_{j \in \mathbb{J}}$. La sortie est calculé en appliquant directement l'opération MHA 3.6 à ces séquences de requêtes, clés et valeur, définissant ainsi l'opérateur d'attention croisée multi-tête (MHCA - Multi-Head Cross-Attention). Une schématisation de ce mécanisme sur une unique tête est représenté dans la figure 3.9.

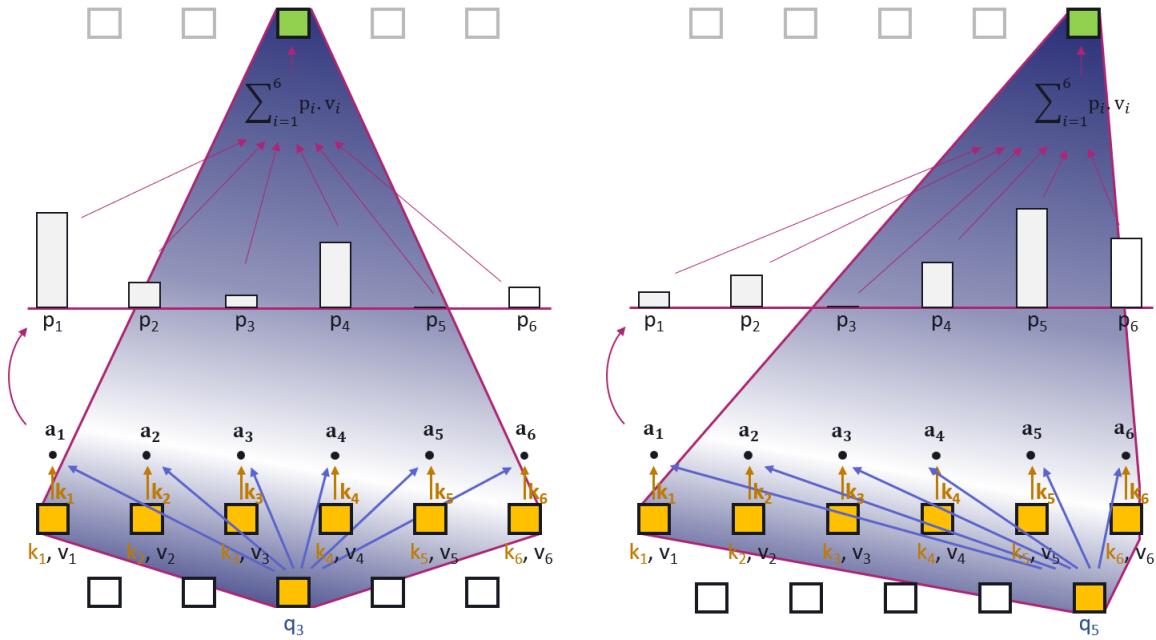


Figure 3.9 – Illustration de l'Attention Croisée (MHCA). Les requêtes (en bleu) proviennent de la séquence cible et interrogent l'intégralité du contexte source (clés en jaune). À gauche, la requête q_3 agrège l'information de la séquence source (à l'origine des clés et valeurs \mathcal{KV}) pour construire $A(q_3, \mathcal{KV})$, représenté dans la cellule verte. Le cône d'interaction est représenté en dégradé de bleu. À droite, le même calcul est effectué indépendamment pour la requête q_5 . Cette indépendance des calculs permet leur parallélisation.

Deuxièmement, lorsque l'objectif est d'enrichir une séquence $(x_i)_{i \in \mathbb{I}}$ par son propre contexte, la même séquence est utilisée pour tous les rôles. Pour toute tête l , on définit $\mathcal{Q}_l = \{Q_l(x_i)\}_{i \in \mathbb{I}}$ et $\mathcal{KV}_l = \{(K_l(x_i), V_l(x_i))\}_{i \in \mathbb{I}}$. L'application de l'opération MHA 3.6 à ces séquences de requêtes, clés et valeur définit alors l'opérateur d'auto-attention multi-tête (MHSA - Multi-Head Self-Attention). Une schématisation de ce mécanisme sur une unique tête est représenté à

gauche de la figure 3.10.

Enfin, la génération autorégressive impose un respect strict de la causalité. Dans l'objectif est d'enrichir une séquence $(x_i)_{i \in \mathbb{I}}$ par son propre contexte de manière causale, les rôles sont encore assignés de manière symétrique, mais le calcul du score d'attention est modifié. Soient x_i et x_j deux éléments de la séquence, générant respectivement une requête q_i et une clé k_j dans la tête l . Le score d'attention est redéfini ainsi :

$$a(q_i, k_j) = \frac{\langle q_i, k_j \rangle}{\sqrt{d_{att}}} \quad \text{si } j \leq i \quad (3.7)$$

$$-\infty \quad \text{si } j > i \quad (3.8)$$

Cette condition assure que pour tout $j > i$, le poids p associé est nul, empêchant toute fuite d'information du futur vers le passé. L'attention masquée 3.8 est réinjecté dans le calcul de pondération 3.2 et l'agrégation suit ensuite la mécanique standard jusqu'à l'équation 3.6, définissant ainsi l'opérateur d'auto-attention masquée multi-tête (MMHSA - Masked Multi-Head Self-Attention). Une schématisation de ce mécanisme sur une unique tête est représenté à droite de la figure 3.10.

3.3.4.6 . Le Transformer dans le texte : La divergence des architectures

Dans le traitement du langage naturel (NLP), le Transformer a provoqué une véritable explosion cambrienne des modèles, se scindant en trois familles distinctes. La première, celle des Encodeurs, est incarnée par BERT [87]. Utilisant une attention bidirectionnelle, ces modèles excelltent dans la compréhension et la classification, car chaque mot a accès au contexte passé et futur simultanément. La seconde, celle des Décodeurs, est dominée par la lignée GPT [45], [88]. Ici, l'attention est causale (masquée vers le futur), optimisée pour la génération autorégressive. C'est cette branche qui a mis en évidence les "lois d'échelle" (Scaling Laws), montrant que la performance de prédiction du prochain token suit une loi de puissance en fonction du nombre de paramètres et de données, ouvrant la voie aux gros modèles de langage (Large Language Model - LLM) actuels. La troisième famille, Encodeur-Décodeur, reste fidèle à l'architecture originale [44] pour les tâches de traduction ou de résumé. Le modèle T5 [89] a poussé ce paradigme à son extrême en reformulant toute tâche NLP (y compris la classification) comme un problème de génération de texte-vers-texte.

3.3.4.7 . Le Transformer dans les systèmes temporels : Promesses et controverses

L'application des Transformers aux séries temporelles continues (consommation énergétique, trafic, météo) a fait l'objet de recherches intenses [90]. L'attrait principal réside dans la capacité théorique de l'attention à capturer des corrélations à très long terme et des saisonsnalités complexes que les RNN peinent à retenir. Des architectures spécifiques ont été proposées pour briser la complexité quadratique. Informer [91] introduit une attention "ProbSparse" pour

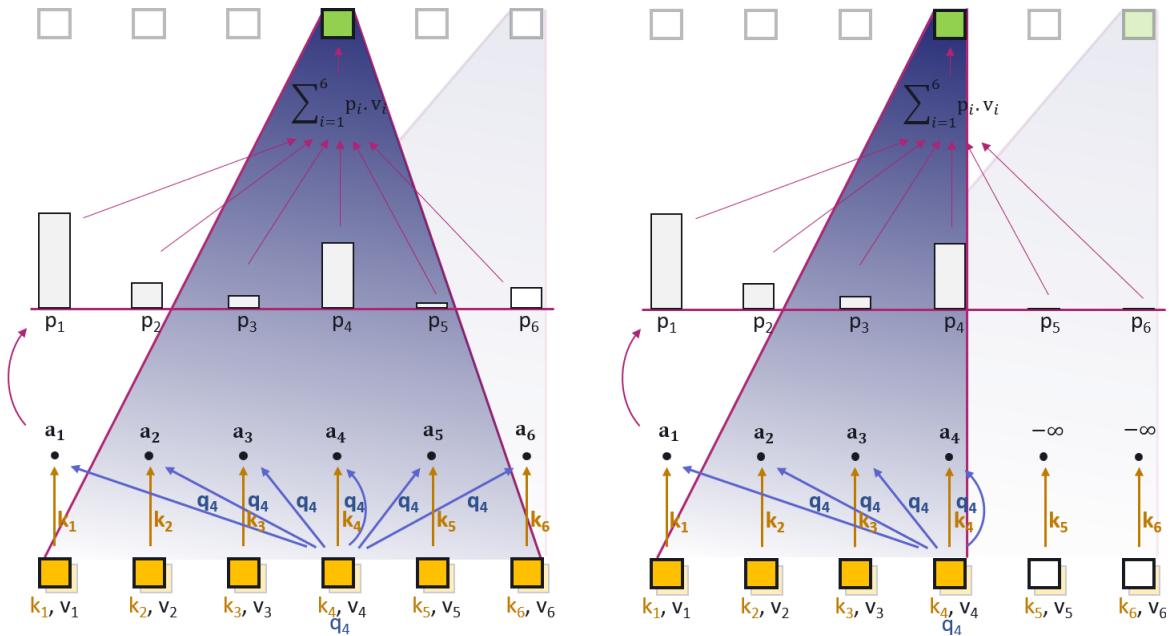


Figure 3.10 – Comparaison des modes d'Auto-Attention. La séquence d'entrée nourrit à la fois \mathcal{Q} et \mathcal{KV} . À gauche, l'interaction est globale. La requête q_4 agrège l'information de toute la séquence pour construire $A(q_4, \mathcal{KV})$, représenté dans la cellule verte. Le cône d'interaction est représenté en dégradé de bleu. À droite, l'interaction est causale. L'accès aux clés futures est donc bloqué (score $-\infty$), cette restriction est visible sur le cône d'interaction en nuance de bleu. La requête q_4 agrège aux positions $j \leq 4$ pour construire $A(q_4, \mathcal{KV})$, qui constitue une prédiction des états futurs. Pour les deux images, on représente en arrière plan les mêmes calculs pour la requête q_6 , réalisés de manière indépendante.

sélectionner uniquement les interactions dominantes, réduisant la complexité à $O(N \log N)$. Autoformer [92] va plus loin en remplaçant le produit scalaire par une auto-corrélation pour mieux capturer les périodicités. Cependant, l'efficacité réelle des Transformers sur des signaux continus est contestée. Une étude [93] assure qu'un simple modèle linéaire bien calibré (DLInear) surpassait souvent des Transformers complexes sur les benchmarks standards. La raison invoquée est que l'attention, conçue pour la sémantique discrète, tend à sur-interpréter le bruit dans les signaux continus et perd l'information d'ordre temporel cruciale, malgré les encodages positionnels. Néanmoins, des approches récentes comme PatchTST [94], qui segmentent le signal en patchs (comme en vision) avant d'appliquer l'attention, semblent redonner l'avantage aux Transformers en traitant des dynamiques locales plutôt que des points isolés.

3.3.4.8 . Le Transformer dans l'image : Patchs et hiérarchie

L'hégémonie des CNN en vision a été remise en cause par le Vision Transformer (ViT) [47]. En découplant l'image en une séquence de patchs carrés traités comme des mots, ViT a prouvé qu'un Transformer pur, sans biais inductif de convolution, pouvait atteindre l'état de l'art, à condition d'être pré-entraîné sur des volumes de données massifs (JFT-300M, [95]). Pour pallier le coût quadratique sur les images haute résolution et le manque de localité, l'architecture Swin Transformer [96] a réintroduit une structure hiérarchique similaire aux CNN. En calculant l'attention uniquement à l'intérieur de fenêtres locales glissantes (Shifted Windows), Swin combine la modélisation globale des Transformers avec l'efficacité locale des convolutions, devenant le standard actuel pour la segmentation et la détection d'objets.

3.3.4.9 . Généralisation : Physique et Prise de décision

La capacité du Transformer à modéliser des graphes d'interaction arbitraires en fait un outil puissant pour la physique et la biologie. L'exemple le plus spectaculaire est AlphaFold 2 [97], qui a résolu le problème du repliement des protéines. Son module central, l'Evoformer, est une variante du Transformer qui traite la protéine comme un graphe dynamique, mettant à jour itérativement la représentation de la séquence d'acides aminés et la matrice de distances 3D par des mécanismes d'attention triangulaire. Enfin, dans le domaine du contrôle et de la simulation, le Decision Transformer [98] a reformulé l'apprentissage par renforcement comme un problème de modélisation de séquence. Au lieu d'estimer des fonctions de valeur ou des gradients de politique, ce modèle prédit simplement l'action suivante conditionnée par les états passés et la récompense désirée (le "Return-to-go"). Cette approche "générationnelle" du contrôle permet d'appliquer les techniques de pré-entraînement des LLM à la robotique ou à la navigation d'agents autonomes, unifiant ainsi perception, prédiction physique et prise de décision sous une même architecture.

3.3.5 . Ancrage dans la problématique

L'exploration des architectures de traitement de séquence met en lumière un éventail de mécanismes complémentaires pour la modélisation de notre simulateur de capteur, dont la

pertinence doit être pondérée par les contraintes spécifiques des signaux radar. Les réseaux convolutionnels, par leur biais inductif de localité, offrent une approche adaptée pour modéliser les interactions à courte portée, telles que les interférences immédiates entre impulsions proches au sein d'un même train. Cependant, leur architecture à fenêtre glissante impose une limitation structurelle majeure : la difficulté à maintenir un état mémoire persistant sur des horizons temporels arbitrairement longs, ce qui peut s'avérer insuffisant pour reproduire fidèlement les processus de pistage temporel qui nécessitent de lier des événements très distants.

De leur côté, les architectures récurrentes (RNN) et les modèles d'espaces d'états (SSM) présentent une affinité naturelle avec la causalité physique du capteur, mimant le comportement des algorithmes de traitement du signal qui mettent à jour des pistes au gré des réceptions. Néanmoins, leur usage impliquerait un changement de paradigme par rapport à notre approche orientée "traduction". Ces modèles excellent dans le traitement séquentiel flux à flux, mais leur application à une tâche de transformation globale de séquence (Seq2Seq) sur de très longs scénarios est complexe. Le goulot d'étranglement du vecteur de contexte, censé compresser toute l'information de la séquence d'entrée avant la génération, devient rapidement prohibitif face à la densité des données radar, limitant leur capacité à reconstruire fidèlement l'ensemble du scénario en une seule passe.

Enfin, l'architecture Transformer et le mécanisme d'attention apportent une capacité de modélisation contextuelle globale, permettant à chaque impulsion d'interagir directement avec l'ensemble de la séquence. Cette propriété est puissante pour capturer des corrélations complexes non-locales et apprendre la fonction de transfert globale du simulateur sans les contraintes de compression des RNN. Toutefois, l'application de ce modèle exige une vigilance particulière quant à sa complexité quadratique, qui peut devenir prohibitive face à la haute densité des flux d'impulsions radar, ainsi qu'à la nécessité d'adapter l'encodage positionnel pour traiter le temps continu irrégulier des PDW plutôt que des indices discrets.

4 - Capacité de discernement des modèles

La fidélité de la modélisation par intelligence artificielle du bloc de Détection et Caractérisation des Impulsions (DCI) dépend intrinsèquement de la capacité du modèle à discerner et comparer finement les éléments d'une séquence. Contrairement aux applications de Traitement du Langage Naturel (NLP) où les données (mots ou tokens) appartiennent à un ensemble fini, les signaux radar évoluent dans des espaces continus. Par conséquent, l'espace latent du modèle est densément peuplé de représentations vectorielles qu'il est complexe de séparer sans ambiguïté (à l'inverse du NLP).

Pour évaluer et valider les capacités de discernement de nos architectures dans ce contexte continu, nous avons défini une tâche de référence : un problème d'ordonnancement supervisé. Le modèle reçoit en entrée une séquence de vecteurs et doit prédire une nouvelle séquence qui doit correspondre à l'entrée réordonnée selon une règle implicite afin de minimiser l'erreur d'apprentissage. L'objectif est de mesurer l'aptitude générale d'un réseau de neurones à identifier les caractéristiques pertinentes et à établir des comparaisons fines entre les éléments pour construire une sortie cohérente.

Cette démarche s'inscrit directement dans le domaine de l'interprétabilité mécaniste (mechanistic interpretability, [99]), qui analyse les capacités algorithmiques des réseaux de neurones. Des études récentes ([100], [101]) ont spécifiquement exploité des tâches de tri sur des entiers pour analyser le fonctionnement des Transformers.

Il est important de préciser que notre objectif n'est pas de concevoir un algorithme de tri performant dans l'absolu, mais d'évaluer des architectures de traitement de séquence généralistes. Nous avons donc exclu de cette étude les modèles spécifiquement conçus pour le tri algorithmique, souvent structurellement incapables de généraliser à d'autres tâches. Notre démarche vise à valider une architecture polyvalente capable, par la suite, de traiter la complexité des signaux radar.

Ce chapitre reprend les résultats de nos travaux publiés à la conférence EUSIPCO [102]. Nous y détaillons notamment deux contributions méthodologiques :

- une fonction de coût pondérée équilibrant l'impact de chaque séquence dans l'apprentissage
- une stratégie de fenêtrage des données permettant un apprentissage par curriculum.

4.1 . Génération des données et protocole de construction

Cette section détaille la méthodologie de construction des ensembles de données synthétiques utilisés pour l'apprentissage supervisé. L'objectif est de générer des couples (\mathbf{X} , \mathbf{Y}) où \mathbf{X} est une séquence de $N = 10$ vecteurs $\{V_1, \dots, V_N\}$ de \mathbb{R}^5 et \mathbf{Y} est la séquence cible ordonnée correspondante.

4.1.1 . Règle d'ordonnancement implicite

Afin d'évaluer la capacité du modèle à extraire une relation d'ordre dans un espace continu, nous formalisons le problème de tri de la manière générale suivante. Soit une séquence d'entrée désordonnée $\mathbf{X} = (V_1, V_2, \dots, V_N)$ composée de N vecteurs appartenant à un espace continu \mathbb{R}^d . L'ordre cible de cette séquence est déterminé par une règle latente, strictement inconnue du réseau de neurones lors de l'apprentissage.

Cette règle repose sur la projection géométrique de chaque élément sur un vecteur directeur caché, noté $U \in \mathbb{R}^d$, de norme unitaire ($\|U\|_2 = 1$). Pour chaque vecteur V_i , un score scalaire S_i est ainsi calculé via le produit scalaire :

$$S_i = \langle V_i, U \rangle \quad (4.1)$$

Ce score unidimensionnel définit l'ordre absolu des éléments. Le processus d'ordonnancement consiste alors à trouver la permutation σ des indices $\{1, \dots, N\}$ garantissant que les scores soient triés par ordre croissant :

$$S_{\sigma(1)} \leq S_{\sigma(2)} \leq \dots \leq S_{\sigma(N)} \quad (4.2)$$

La séquence triée correspondante, que le modèle doit apprendre à générer, est donc la séquence des vecteurs d'entrée réorganisés selon cette permutation exacte :

$$\mathbf{Y} = (V_{\sigma(1)}, V_{\sigma(2)}, \dots, V_{\sigma(N)})$$

Ce principe global de cette transformation, du calcul du score jusqu'à la restitution de la séquence ordonnée, est illustré dans la figure 4.1.

La tâche du modèle consiste à prédire la séquence ordonnée de vecteurs \mathbf{Y} à partir de \mathbf{X} , en se basant uniquement sur l'observation des données, sans connaissance a priori de la règle d'ordonnancement donnant naissance à σ , ce qui oblige l'architecture à développer une capacité de discernement fin entre des vecteurs très proches dans l'espace multidimensionnel. Dans le cadre spécifique de nos expérimentations, nous avons instancié ce problème général pour des séquences de $N = 10$ éléments évoluant dans un espace de dimension $d = 5$.

4.1.2 . Définition de la difficulté et invariances

Afin d'évaluer la robustesse du modèle, nous introduisons une mesure quantifiant la difficulté intrinsèque de tri d'une séquence. Intuitivement, une séquence est difficile à trier si les scores de ses éléments sont proches les uns des autres relativement à l'étendue globale des valeurs possibles.

Pour caractériser cette mesure de difficulté, nous définissons deux métriques intermédiaires à l'échelle de la séquence de vecteurs :

- La moyenne μ_S , correspond à la moyenne arithmétique des scores de la séquence.
- L'écart-type σ_S , quantifie la dispersion des scores autour de cette moyenne.

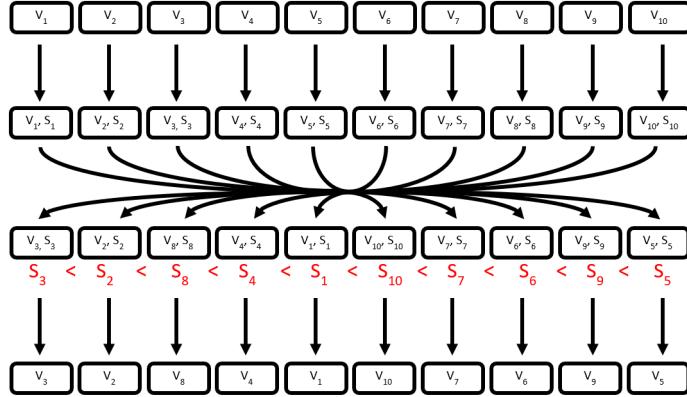


Figure 4.1 – Diagramme de la procédure de tri. Les différentes étapes sont visibles de haut en bas : réception de la séquence désordonnée, projection et association d'un score par élément, détermination de la permutation σ par tri croissant des scores, et enfin restitution de la séquence vectorielle ordonnée.

Ces métriques prennent leurs sens lorsqu'elles sont confrontées aux contraintes globales du problème. Ces contraintes globales en partie caractérisés par un $[\mu_{min}, \mu_{max}]$, une constante fixée pour la génération des données, correspondant à l'intervalle dans lequel les μ_S sont contraintes de se trouver.

Nous pouvons à présent introduire le **ratio de séparabilité** r d'une séquence, définit comme le rapport entre sa mesure σ_S et cette étendue globale sur laquelle sa mesure μ_S peut se trouver :

$$r = \frac{\sigma_S}{\mu_{max} - \mu_{min}} \quad (4.3)$$

Cette formulation respecte les intuitions naturelles concernant une tâche de discernement :

- Trier des scores espacés de 1 sur une plage de 10 répartie autour de 0 ($r = \frac{1}{5-(-5)} = 0.1$) ou autour de 100 ($r = \frac{1}{105-95} = 0.1$) **est équivalente** pour un réseau de neurones, qui peut aisément apprendre un biais additif.
- Trier des scores espacés de 1 sur une plage de 10 ($r = \frac{1}{10} = 0.1$) ou trier des scores espacés de 10 sur une plage de 100 ($r = \frac{10}{100} = 0.1$) **est équivalent** pour un réseau de neurones, qui peut apprendre un changement d'échelle.
- Trier des scores espacés de 1 sur une plage de 100 ($r = \frac{1}{100} = 0.01$) ou trier des scores espacés de 10 sur une plage de 100 ($r = \frac{10}{100} = 0.1$) **n'est pas équivalent** pour un réseau de neurones, qui nécessite un niveau de discernement accru pour le premier cas.

Ainsi, le paramètre r contrôle le niveau de discernement exigé du réseau pour ordonner la

séquence correspondante. La difficulté de traitement d'une séquence par le modèle peut donc être définie comme l'inverse du ratio de séparabilité, qui augmente à mesure que r diminue.

4.1.3 . Algorithme de génération sous contrainte

Pour imposer le ratio de séparabilité r souhaité, nous procédons par une méthode inverse : nous fixons d'abord les statistiques des scores cibles $(S_i)_i$, puis nous construisons les vecteurs $(V_i)_i$ correspondants.

La procédure de génération pour une séquence de ratio r , contrainte au domaine global $[\mu_{min}, \mu_{max}]$, est la suivante :

- **Échantillonnage des paramètres locaux** : Nous tirons aléatoirement la moyenne spécifique de la séquence, notée μ , selon une loi uniforme sur le domaine global : $\mu \sim \mathcal{U}([\mu_{min}, \mu_{max}])$. L'écart-type cible de la séquence est ensuite déterminé par la contrainte de difficulté : $\sigma = r \times (\mu_{max} - \mu_{min})$.
- **Génération des scores** : Les scores $\{S_1, \dots, S_N\}$ sont tirés de manière i.i.d. selon une loi normale $\mathcal{N}(\mu, \sigma^2)$.
- **Initialisation des vecteurs** : Des vecteurs bruts V_i^{raw} sont générés. Afin de maintenir une cohérence statistique entre la norme du vecteur et son score projeté, V_i^{raw} est tiré selon $\mathcal{N}(0, S_i \mathbf{I}_5)$.
- **Correction du score** : On ajuste la composante de V_i^{raw} colinéaire à U pour assurer l'égalité $\langle V_i, U \rangle = S_i$, tout en préservant la composante orthogonale :

$$V_i = V_i^{raw} - \langle V_i^{raw}, U \rangle U + S_i U \quad (4.4)$$

Ce protocole assure une double propriété statistique à l'ensemble de données généré. À l'échelle macroscopique des données, les séquences couvrent uniformément tout le domaine $[\mu_{min}, \mu_{max}]$. À l'échelle des séquences, chaque exemple respecte la contrainte de séparabilité r avec laquelle elle a été contrainte, permettant de composer les ensembles d'apprentissage avec plus ou moins de difficulté.

4.2 . Stratégie d'apprentissage et architectures neuronales

La génération de données synthétiques nous permet de produire des séquences de difficulté arbitraire, mais l'entraînement d'un réseau de neurones sur ces données requiert une méthodologie spécifique. Comme nous l'avons exposé dans nos travaux précédents, l'approche standard de minimisation de l'erreur quadratique moyenne s'avère insuffisante face à la dynamique des scores dans un espace continu. Cette section détaille la fonction de coût pondérée et la stratégie de curriculum mises en place pour permettre la convergence des modèles.

4.2.1 . Fonction de coût normalisée

Nous détaillons ici la fonction de coût utilisée lors de l'entraînement de nos modèles, telle que présentée dans l'article [CITE]. Cette métrique se base initialement sur une erreur de re-

construction brute $\mathcal{L}_{raw}(\hat{\mathbf{Y}}_k, \mathbf{Y}_k)$ entre une séquence prédictée $\hat{\mathbf{Y}}_k$ et une séquence cible ordonnée \mathbf{Y}_k :

$$\mathcal{L}_{raw}(\hat{\mathbf{Y}}_k, \mathbf{Y}_k) = \frac{\|\hat{\mathbf{Y}}_k - \mathbf{Y}_k\|_2}{\sqrt{N \cdot d}} \quad (4.5)$$

Si l'on considère un prédicteur naïf générant en sortie N répétitions du vecteur moyen de la séquence cible μ_k , l'erreur brute résultante $\mathcal{L}_{raw}(\mathbf{1}\mu_k^\top, \mathbf{Y}_k)$ devient proportionnelle à la dispersion (ou l'amplitude) des éléments de la séquence \mathbf{Y}_k . L'utilisation directe de l'erreur brute induirait donc un déséquilibre dans le processus d'optimisation, favorisant les séquences de forte amplitude au détriment de la précision fine.

Pour corriger ce biais, nous définissons un facteur de normalisation par séquence, correspondant à cette erreur "naïve", qui permet d'uniformiser l'influence des séquences dans le gradient total :

$$\mathcal{L}_{ref}(\mathbf{Y}_k) = \frac{\|\mathbf{Y}_k - \mathbf{1}\mu_k^\top\|_2}{\sqrt{N \cdot d}} \quad (4.6)$$

La métrique pondérée, assurant une contribution équitable de chaque séquence au processus d'apprentissage, est alors définie par le ratio :

$$\mathcal{L}_{weighted}(\hat{\mathbf{Y}}_k, \mathbf{Y}_k) = \frac{\mathcal{L}_{raw}(\hat{\mathbf{Y}}_k, \mathbf{Y}_k)}{\mathcal{L}_{ref}(\mathbf{Y}_k)} \quad (4.7)$$

Enfin, cette métrique, sommée sur un ensemble d'apprentissage (batch) de b éléments, définit la fonction de coût globale minimisée durant l'entraînement :

$$\mathcal{L}\left((\hat{\mathbf{Y}}_k)_{1 \leq k \leq b}, (\mathbf{Y}_k)_{1 \leq k \leq b}\right) = \sqrt{\frac{1}{b} \sum_{k=1}^b \mathcal{L}_{weighted}(\hat{\mathbf{Y}}_k, \mathbf{Y}_k)^2} \quad (4.8)$$

4.2.2 . Métrique d'évaluation : La Justesse Positionnelle

Bien que la fonction de coût \mathcal{L} guide l'optimisation des poids, elle reste une mesure de distance continue qui peut être difficile à interpréter en termes de réussite ou d'échec du tri. Pour évaluer la performance opérationnelle du modèle, nous introduisons une seconde métrique : la justesse (ou accuracy), notée \mathcal{J} .

La justesse évalue la capacité du modèle à restaurer l'ordre des éléments. Comme on ne produit pas explicitement les éléments de la séquence triée $\mathbf{Y} = (V_1, \dots, V_{10})$ mais seulement une approximation $\hat{\mathbf{Y}} = (\hat{V}_1, \dots, \hat{V}_{10})$, on procède en calculant l'indice correspondant au vecteur de la séquence triée le plus proche de chaque prédiction \hat{V}_i :

$$id(\hat{V}_i, \mathbf{Y}) = \operatorname{argmin}_{1 \leq j \leq 10} (\|V_j - \hat{V}_i\|_2)$$

Si $\text{id}(\hat{V}_i, \mathbf{Y}) = i$, on considère que le réseau a placé correctement l'élément i . En moyennant sur l'ensemble de la séquence, on obtient la justesse d'une prédiction :

$$\mathcal{J}_{local}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{10} \sum_{i=1}^{10} \mathbb{1}_{\{i\}}(\text{id}(\hat{V}_i, \mathbf{Y})) \quad (4.9)$$

Cette métrique est ensuite moyennée sur l'ensemble des b séquences du lot (batch) ou du jeu de test pour obtenir une évaluation globale de la justesse du modèle :

$$\mathcal{J} = \frac{1}{b} \sum_{k=1}^b \mathcal{J}_{local}(\mathbf{Y}_k, \hat{\mathbf{Y}}_k) \quad (4.10)$$

Le résultat fournit une valeur entre 0 et 1 (100%), directement interprétable comme le pourcentage d'impulsions correctement réordonnées par le modèle.

4.2.3 . Apprentissage progressif

L'exposition immédiate du modèle à l'ensemble de la distribution des difficultés mène fréquemment à des optima locaux insatisfaisants. Pour contourner ce problème, nous adoptons une stratégie d'apprentissage progressive.

L'objectif final est de permettre au modèle de généraliser sur l'espace complet du problème. Dans nos expériences, cet espace cible est défini par l'intervalle de moyennes $[\mu_{min}, \mu_{max}] = [-10, 10]$ et l'intervalle d'écart-types $[\sigma_{min}, \sigma_{max}] = [10^{-4}, 5]$, ce qui correspond à une exigence de séparabilité de $r = 5.10^{-6}$.

Pour y parvenir, nous définissons des fenêtres d'apprentissage transitoires caractérisées par le domaine $[\mu_{min}, \mu_{max}] \times [\lambda, \sigma_{max}]$. Le paramètre évolutif λ agit comme une borne inférieure mobile pour l'écart-type, augmentant progressivement la complexité des données.

La stratégie consiste à faire décroître λ par paliers successifs. L'entraînement débute avec $\lambda = 1$. À ce stade, les séquences sont générées avec un écart-type $\sigma \in [1, 5]$, correspondant exclusivement à des cas facilement séparables. Le paramètre est ensuite réduit progressivement à travers 9 fenêtres successives, suivant la suite de valeurs $\lambda \in [1; 3.10^{-1}; 1.10^{-1}; 3.10^{-2}; 1.10^{-2}; 3.10^{-3}; 1.10^{-3}; 3.10^{-4}; 1.10^{-4}; 3.10^{-5}; 1.10^{-5}; 3.10^{-6}; 1.10^{-6}]$ jusqu'à exposer le modèle à la totalité de la densité des données. Chaque étape du curriculum comprend 170 itérations sur 500 000 séquences, assurant un transfert d'apprentissage efficace et stable du cas simple vers le cas complexe.

4.2.4 . Architectures évaluées

Afin de dissocier la difficulté intrinsèque du problème des capacités du modèle, nous évaluons cette méthodologie sur deux architectures distinctes. La première est un Réseau de Neurones Convolutif (CNN) spécialisé dans le traitement de séquence [63], composé de 10 couches

de 10 neurones, totalisant 1.9 millions de paramètres. La seconde, qui constitue notre cible principale, est l'encodeur de l'architecture Transformer [44]. Ce modèle, configuré avec 10 couches d'attention à 4 têtes dans un espace de dimension 128 (0.6M paramètres), exploite le mécanisme de self-attention pour capturer les dépendances globales nécessaires à la comparaison inter-éléments.

4.3 . Résultats expérimentaux et analyse

Cette section présente l'évaluation des performances des modèles sur la tâche de tri continu.

4.3.1 . Protocole de visualisation

Afin d'analyser le comportement des modèles sur l'ensemble du domaine opératoire, nous adoptons une double approche. La première repose sur une représentation cartographique permettant de visualiser pour quelles valeurs de moyenne μ et d'écart-type σ les séquences mettent le modèle en difficulté. La seconde consiste en une analyse quantitative globale, permettant de visualiser, en fonction de la difficulté des données, à la fois l'erreur et la justesse du modèle, mais aussi la distribution de ces quantités (à travers des diagrammes à boîte) sur les données.

4.3.1.1 . Cartographies de performance

Pour visualiser la performance du modèle au-delà d'un simple scalaire, nous introduisons une représentation matricielle de l'espace des séquences sous forme d'images de 40×40 pixels. Chaque pixel de cette cartographie correspond à la performance du modèle à un point de fonctionnement précis, évaluée sur un ensemble de séquences générées avec des statistiques (μ, σ) fixes. Dans cette représentation, l'axe horizontal représente la moyenne μ des scores de la séquence, tandis que l'axe vertical représente l'écart-type σ des scores sur une échelle logarithmique. La couleur du pixel code l'intensité de la performance (Erreur ou Justesse). Cette visualisation permet d'identifier instantanément les corrélations entre les propriétés statistiques des données et la capacité du modèle à les traiter. Par exemple, la figure 4.2 illustre un modèle échouant sur des séquences avec des faibles valeurs σ , donc difficilement séparable.

4.3.1.2 . Analyse statistique par diagrammes en boîte

En complément des cartographies, nous utilisons des diagrammes en boîte couplés à des courbes de performance moyenne. Ces graphiques représentent l'erreur et la justesse des modèles en fonction de la difficulté des données (paramétrée par λ). L'intérêt de cette représentation est de visualiser non seulement la moyenne, mais aussi la dispersion des performances à travers l'usage de diagramme en boîte, exprimant simplement la présence de régions qui échappent à la compréhension du modèle.

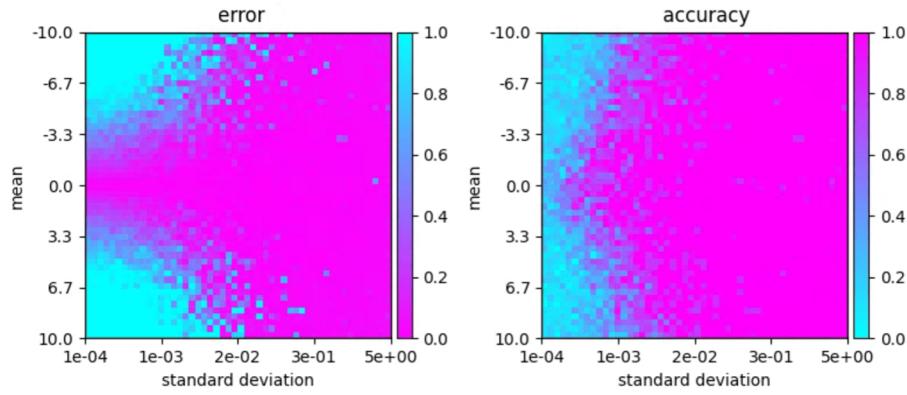


Figure 4.2 – Cartographie des performances du Transformer de référence (Baseline). Le modèle est entraîné directement sur l’intégralité du domaine ($[\mu_{min}, \mu_{max}] = [-10, 10]$ et $[\sigma_{min}, \sigma_{max}] = [10^{-4}, 5]$) durant 1600 itérations (500 000 séquences). L’erreur de reconstruction est représenté à gauche, la justesse est représenté à droite. On observe un échec sur les zones de faible séparabilité (gauche de chaque image). La décorrélation entre les images visible au centre gauche (zone violette de faible erreur apparente mais de justesse nulle) illustre l’incapacité de la fonction de coût standard non pondérée à guider l’apprentissage sur des séquences de faible amplitude.

4.3.2 . Évaluation du modèle de référence

Dans un premier temps, nous évaluons un modèle de référence entraîné selon une approche conventionnelle, c'est-à-dire sans les contributions méthodologiques proposées dans ce chapitre. L'apprentissage est réalisé directement sur l'ensemble du domaine (sans curriculum) et minimise une Erreur Quadratique Moyenne (RMSE) classique. Cette erreur est uniquement normalisée par l'écart-type global des sorties du jeu de données pour assurer la stabilité numérique, mais reste dépourvue de la pondération séquence par séquence que nous avons introduite.

L'analyse des cartographies de ce modèle, figure 4.2, révèle un échec significatif de l'apprentissage sur une large partie du domaine. La carte de justesse montre que le modèle échoue quasi-systématiquement à trier les séquences présentant une faible dispersion σ (partie inférieure de l'image). Plus notable encore, la carte d'erreur présente une anomalie majeure : une zone de faible erreur apparente (en violet) est visible sur la gauche de l'image, là où la justesse est pourtant nulle. Cet artefact démontre l'incapacité de la fonction de coût standard à guider l'apprentissage : le modèle minimise mécaniquement l'erreur absolue sur les séquences de faible amplitude (faible μ et σ) sans apprendre l'ordre correct, créant une illusion de convergence.

Ces observations qualitatives sont corroborées par l'analyse quantitative, figure 4.3. Les courbes de performance montrent une chute rapide de la justesse dès que la difficulté aug-

mente. De plus, les diagrammes en boîte mettent en évidence une dispersion très importante des performances. Cette hétérogénéité confirme que le modèle échoue sur les séquences à faible séparabilité, cas critiques qui deviennent prépondérants dans les données testées à mesure que la complexité augmente.

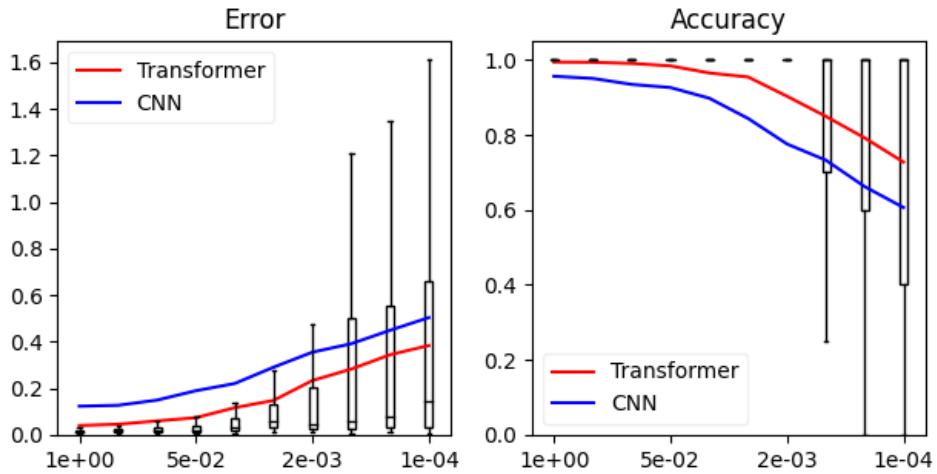


Figure 4.3 – Évolution de la performance globale en fonction de la difficulté (λ). Les courbes pleines comparent les performances moyennes (Erreur à gauche et Justesse à droite) du CNN et du Transformer entraînés en 1600 itérations sur 500000 séquences distribuées sur $[\mu_{min}, \mu_{max}] = [-10, 10]$ et $[\sigma_{min}, \sigma_{max}] = [\lambda, 5]$. Les diagrammes en boîte (représentés uniquement pour le Transformer par souci de lisibilité) mettent en évidence une disparité croissante des capacités de traitement : à mesure que la difficulté augmente (λ diminue vers 10^{-4}), la distribution des résultats s'étale considérablement, confirmant que le modèle échoue sur les séquences les plus complexes devenues prépondérantes.

4.3.3 . Évaluation de la méthodologie proposée

Nous évaluons ensuite l'efficacité de notre méthodologie complète, qui couple la stratégie d'apprentissage progressif par fenêtre à la fonction de coût pondérée.

L'évolution des cartographies de performance, figure 4.4, illustre la dynamique de cette approche : on observe la progression d'un "front". Le modèle étend progressivement sa capacité de généralisation vers les zones de faible séparabilité à mesure que les fenêtres d'apprentissage s'agrandissent. Contrairement au cas de référence, la progression de la justesse est ici parfaitement corrélée à celle de l'erreur pondérée.

Cette robustesse est confirmée par l'analyse statistique finale, représenté figure 4.5. Les diagrammes en boîte montrent une dispersion des performances bien plus faible, même pour les niveaux de difficulté élevés. Cela indique que les performances du modèle sont homogènes

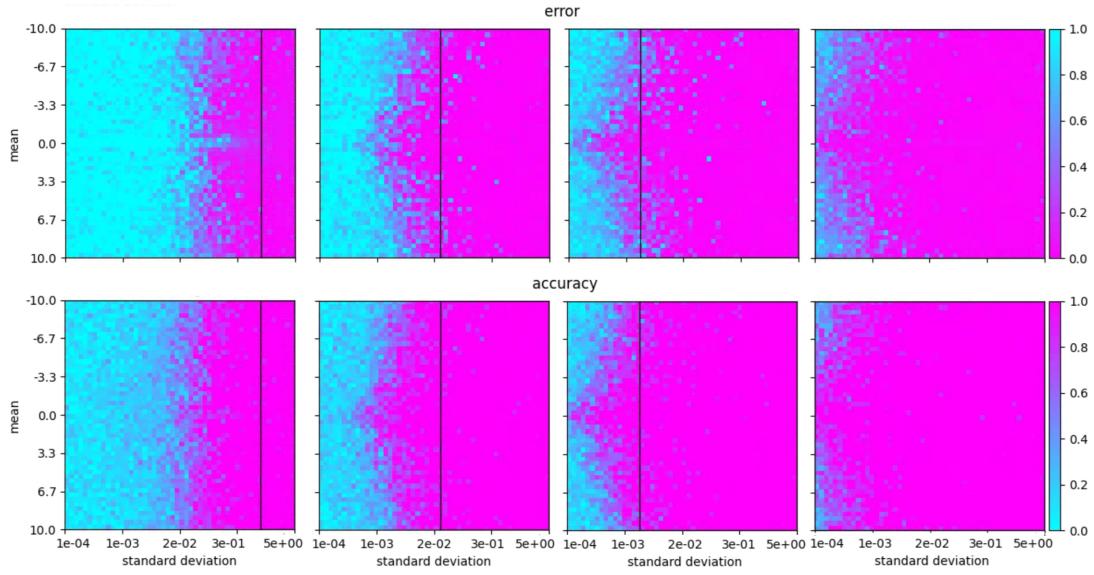


Figure 4.4 – Performance du Transformer à différentes étapes de l'apprentissage (de gauche à droit) : après la 2^e, la 4^e, la 8^e et la 10^e fenêtres. L'erreur est présenté à gauche, la justesse à droite. La ligne noire verticale représente la limite de la dernière fenêtre d'entraînement, ce qui permet d'identifier la propagation des performances du modèle vers les zones de faible séparabilité.

sur l'ensemble des séquences, quelle que soit leur configuration statistique (μ, σ), validant ainsi l'efficacité de la stratégie de pondération et d'apprentissage progressif.

4.3.4 . Synthèse et comparaison des architectures

En conclusion, nous comparons les performances finales atteintes par le Transformer et le CNN grâce à notre méthodologie. Les résultats démontrent la supériorité de l'architecture Transformer pour cette tâche de discernement fin. Alors que les deux modèles affichent des performances comparables sur les tâches triviales ($\lambda \approx 1$), le CNN voit ses performances se dégrader plus rapidement lorsque la séparabilité diminue. Le Transformer, grâce à son mécanisme d'attention globale, maintient une justesse élevée (proche de 100%) sur une plage de difficulté beaucoup plus étendue.

Quantitativement, l'apport de la stratégie d'apprentissage progressif et de la pondération est déterminant : il permet de repousser la limite de discernement du modèle de plusieurs ordres de grandeur. Là où l'approche de référence échouait pour des ratios de séparabilité $r \approx 10^{-4}$, notre stratégie permet au Transformer de maintenir un tri cohérent jusqu'à $r \approx 5.10^{-6}$. Cette marge de progression valide la pertinence de l'approche itérative pour l'apprentissage de relations d'ordre dans les espaces continus denses.

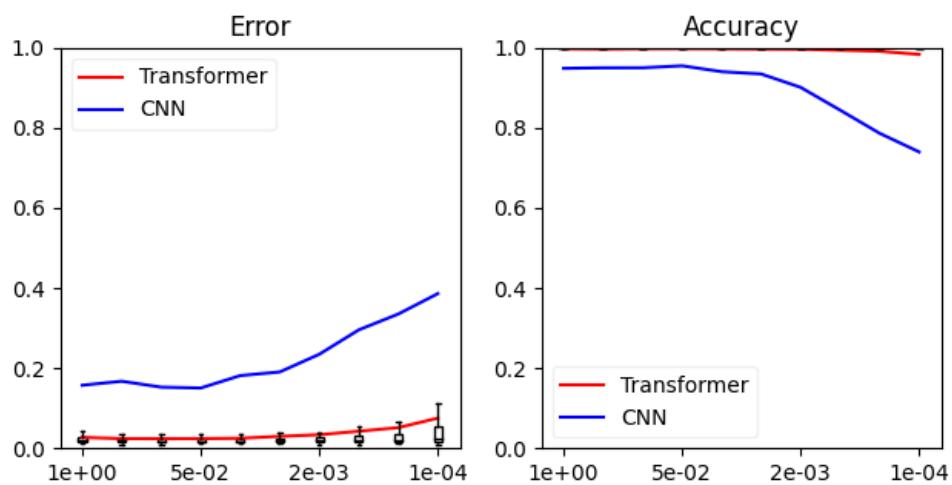


Figure 4.5 – Performance finale avec stratégie d'apprentissage progressif et pondération. Les courbes présentent l'évolution de l'erreur (gauche) et de la justesse (droite) à l'issue du protocole d'apprentissage progressif sur les mêmes fenêtres de difficulté (λ variant de 1 à 10^{-4}). Contrairement au cas de référence 4.3, les diagrammes en boîte du Transformer révèlent une dispersion minime même lorsque la difficulté est maximale. Cette homogénéité appuie l'efficacité de l'approche.

5 - Le cœur de ma thèse ici

Ici ça va être un peu long.

6 - Dépliage du problème : une approche événementielle

Plan

- Motivation
- Formalisation du problème
- Visualisation (vraiment utile)
- Présentation CAID
- Extension des essais
- pré-traitement
- architecture
- fonction de perte
- résultats
- Conclusion

Ce chapitre introduit une méthodologie alternative pour le traitement des signaux radar. En rupture avec les approches globales de type *sequence-to-sequence* (Seq2Seq) abordées précédemment, nous proposons ici une reformulation du problème axée sur la causalité et le traitement flux-à-flux (*stream processing*). Cette approche vise à s'affranchir des limitations de longueur de séquence tout en mimant plus fidèlement le comportement physique du système de Détection et Caractérisation des Impulsions (DCI).

6.1 . Motivation : Du traitement de séquence à la modélisation dynamique

L'approche classique consistant à traiter l'intégralité d'une séquence d'impulsions en une seule passe se heurte à deux obstacles majeurs : un goulot d'étranglement computationnel et une dissonance conceptuelle avec le système réel.

6.1.1 . Le goulot d'étranglement des architectures Seq2Seq

Le premier obstacle est la difficulté intrinsèque des architectures neuronales actuelles à maintenir des dépendances temporelles sur de longs horizons :

- **Architectures récurrentes (RNN/LSTM)** : Ces modèles peinent à conserver une mémoire contextuelle pertinente au-delà d'une dizaine d'éléments. La propagation du gradient à travers le temps (Backpropagation Through Time) sur de longues séquences entraîne des problèmes de disparition ou d'explosion du gradient, rendant l'apprentissage instable.
- **Architectures attentionnelles (Transformers)** : Bien que plus robustes, elles présentent une complexité algorithmique quadratique $\mathcal{O}(N^2)$ en fonction de la longueur de la sé-

quence. Au-delà de séquences de l'ordre de 1000 éléments, l'empreinte mémoire devient prohibitive.

Ces limitations imposent souvent des mécanismes de segmentation arbitraires (découpage de la scène en sous-blocs) qui brisent la continuité temporelle du signal et nuisent à la cohérence globale de la détection.

6.1.2 . Le DCI comme système dynamique réactif

Le second obstacle est d'ordre conceptuel. Modéliser le DCI comme une fonction globale $f(\mathbf{X}_{total}) \rightarrow \mathbf{Y}_{total}$ est une approximation qui masque la réalité physique du module. Le DCI ne "voit" pas le futur; il fonctionne intrinsèquement en mode événementiel. C'est un système dynamique réactif régis par deux types d'événements :

1. **L'arrivée d'une impulsion incidente** déclenche une mise à jour des corrélations et de l'état des mémoires internes.
2. **L'émission d'une impulsion synthétisée** par le mécanisme de suivi provoque la libération de ressources mémoires.

Il apparaît donc plus naturel de chercher à modéliser ce comportement "pas-à-pas". L'objectif est de transformer le problème de prédiction globale en un problème de prédiction locale causale : étant donné une impulsion incidente, le modèle doit prédire la cascade d'événements (émissions) qui surviennent avant l'arrivée de l'impulsion incidente suivante.

6.2 . Formalisation du problème "Flux-à-flux"

Pour implémenter cette approche avec un réseau de neurones, nous devons transformer la structure des données. Cette transformation résulte de la confrontation entre la nature continue du flux radar et la nature discrète des réseaux de neurones. Nous détaillons ici les choix de conception nécessaires pour résoudre ce conflit.

6.2.1 . Contraintes architecturales et choix de conception

Un réseau de neurones standard est une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ qui associe un tenseur de sortie unique à un tenseur d'entrée unique. Cette définition rigide nous impose plusieurs contraintes pour le traitement d'un flux d'impulsions incidentes $\{I_k\}$ et d'impulsions émises (sorties) $\{O_k\}$:

Contrainte 1 : Discrétilsation de la réponse Le modèle ne peut pas "naturellement" générer un nombre variable d'impulsions de sortie pour une seule impulsion d'entrée sans modifier profondément son architecture.

- *Choix de conception* : Nous adoptons une approche itérative. Nous interrogeons le réseau de manière répétée avec la même impulsion incidente tant qu'il reste des impulsions à générer pour ce créneau temporel.

Contrainte 2 : Condition d'arrêt Puisque le nombre de sorties est variable (potentiellement nul), le modèle doit posséder un moyen explicite de signaler la fin de la séquence d'émission associée à l'impulsion incidente courante.

- *Choix de conception* : Nous introduisons un jeton de contrôle spécial, noté `NEXT`. La prédiction de ce jeton par le réseau signifie "Il n'y a plus d'émission à générer, passez à l'impulsion incidente suivante".

Contrainte 3 : Maintien du contexte Le modèle doit savoir où il se situe dans la séquence des émissions (doit-il prédire la première ou la troisième impulsion associée à I_k ?).

- *Choix de conception* : Nous optons pour une architecture à double entrée. Le modèle reçoit à chaque pas :
 1. L'impulsion incidente courante I_k .
 2. La dernière prédiction effectuée P_{t-1} (ou un jeton d'initialisation).

Ce dernier choix s'apparente à une approche auto-régressive guidée (Teacher Forcing), souhaitant la mémoire interne du modèle en lui rappelant explicitement son état précédent.

6.3 . Protocole de transformation des données

La combinaison de ces choix aboutit à un algorithme de "dépliage" des données. Une séquence temporelle d'événements est transformée en une série de couples ((Entrées), Cible) indépendants.

6.3.1 . Illustration du mécanisme

Considérons une chronologie mixte d'événements réels, triés par temps, impliquant 3 impulsions incidentes (I) et 5 impulsions émises (O) :

$$\text{Séquence Globale : } [I_1, O_1, O_2, O_3, I_2, I_3, O_4, O_5] \quad (6.1)$$

L'objectif est de prédire les événements en gras (O) en fonction des événements incidents (I). Le jeton `NEXT` est utilisé pour initialiser la boucle (entrée) et pour la clore (cible). Le processus de génération des données d'entraînement se déroule comme suit :

1. **Traitements de I_1 (3 émissions associées) :**
 - Entrée : $(I_1, \text{NEXT}) \rightarrow \text{Cible} : O_1$ (Première émission).
 - Entrée : $(I_1, O_1) \rightarrow \text{Cible} : O_2$ (Le modèle sait qu'il a émis O_1).
 - Entrée : $(I_1, O_2) \rightarrow \text{Cible} : O_3$ (Le modèle sait qu'il a émis O_2).
 - Entrée : $(I_1, O_3) \rightarrow \text{Cible} : \text{NEXT}$ (Fin des émissions pour I_1).
2. **Traitements de I_2 (0 émission associée) :**
 - L'impulsion suivante I_3 arrive avant toute émission.
 - Entrée : $(I_2, \text{NEXT}) \rightarrow \text{Cible} : \text{NEXT}$ (Passage immédiat à la suite).
3. **Traitements de I_3 (2 émissions associées) :**

- Entrée : $(I_3, \text{NEXT}) \rightarrow \text{Cible} : O_4$.
- Entrée : $(I_3, O_4) \rightarrow \text{Cible} : O_5$.
- Entrée : $(I_3, O_5) \rightarrow \text{Cible} : \text{NEXT}$.

Au final, la séquence temporelle initiale est convertie en un ensemble de 8 échantillons d'apprentissage. Cette transformation permet de ramener un problème de traitement de séquence complexe et long à une succession de tâches de régression locales de courte portée, parfaitement adaptées à l'entraînement d'un réseau de neurones récurrent supervisé.

6.4 . Validation de l'approche sur architectures récurrentes

Dans cette section, nous présentons les résultats publiés lors de la conférence CAID. Cet article démontre que pour la modélisation du traitement radar, l'approche "flux-à-flux" (ici désignée par le terme Forecasting) est particulièrement pertinente pour capturer la causalité temporelle du système. L'étude établit également quelle variante d'architecture récurrente offre le meilleur compromis entre mémoire et stabilité dans ce contexte. Cette section se concentre exclusivement sur les résultats obtenus en mode "prévision" (forecasting), qui correspond à la configuration opérationnelle cible. Nous omettrons ici la description détaillée de la transformation des données (le dépliage en séquences d'événements) ainsi que la métrique spécifique utilisée pour l'optimisation. Ces éléments méthodologiques sont détaillés dans l'article et une transformation analogue sera exposée en détail dans la section suivante consacrée aux extensions de ces travaux. La fonction de coût utilisée ici est comparable à l'équation (4) présentée précédemment, où le terme de normalisation est calculé sur la séquence événementielle dépliée.

6.4.1 . Les architectures

Afin d'identifier la structure la plus adaptée, nous avons évalué un panel d'architectures récurrentes classiques. Ce choix est motivé par la nature séquentielle et causale du problème, où chaque prédiction dépend de l'historique des états internes. Les modèles comparés sont les suivants :

- **RNN (Vanilla Recurrent Neural Network)** : Il constitue l'architecture de référence la plus simple. Bien que théoriquement capable de traiter des séquences, il souffre notoirement du problème de disparition du gradient, limitant sa capacité à retenir l'information sur de longues périodes.
- **LSTM (Long Short-Term Memory)** : Cette architecture introduit des portes logiques (entrée, oubli, sortie) pour réguler le flux d'information. Elle est conçue spécifiquement pour capturer les dépendances à long terme qui échappent aux RNN simples.
- **GRU (Gated Recurrent Unit)** : Variante simplifiée du LSTM, le GRU fusionne certaines portes pour réduire le nombre de paramètres tout en conservant une capacité de mémoire comparable. Il offre souvent une convergence plus rapide et une meilleure stabilité numérique.

- **LSTMAT (LSTM + Attention Additive)** : Il s'agit d'un décodeur LSTM augmenté d'un mécanisme d'attention de type Bahdanau. À chaque pas de temps, le modèle calcule un score additif pour pondérer l'importance des états cachés passés, permettant de "relire" l'historique pertinent.
- **LSTMAT-L (LSTM + Attention de Luong)** : Cette variante utilise un mécanisme d'attention bilinéaire (produit scalaire généralisé). Plus efficace calculatoirement, cette formulation tend à mieux passer l'échelle sur les longues séquences en offrant des gradients plus nets lors de l'alignement temporel.

6.4.2 . Résultats expérimentaux

L'évaluation concerne ici uniquement la configuration Forecasting, où le modèle ne dispose d'aucun accès au futur et génère les impulsions de manière purement causale. Afin de garantir une comparaison équitable, chaque architecture a fait l'objet une optimisation rigoureuse de ses hyper-paramètres via un algorithme bayésien (TPE via Optuna). Cette étape a permis de s'assurer que les écarts de performance observés sont structurels et non liés à un mauvais réglage. La table ci-dessous résume la configuration géométrique optimale (nombre de couches L et dimension latente H) retenue pour chaque modèle à l'issue de cette optimisation : Les entraînements ont été réalisés sur des ensembles de données synthétiques de 500

Architecture	Dimension cachée (H)	Nombre de couches (L)
RNN	640	4
GRU	640	5
LSTM	768	4
LSTMAT (Basic)	768	4
LSTMAT-L (Luong)	768	6

Table 6.1 – Configurations optimales des architectures évaluées (issues de l'annexe de l'article [ref]).

ooo séquences, permettant de couvrir une grande diversité de scénarios d'interférence. La performance est mesurée par l'erreur quadratique moyenne normalisée (NRMSE) sur des horizons de prédiction croissants (10, 20 et 50 impulsions).

Modèle	10 impulsions	20 impulsions	50 impulsions
RNN	0.1261	0.2109	0.2156
GRU	0.1260	0.1981	0.1945
LSTM	0.1146	0.2192	0.2001
LSTMAT (Basic)	0.1053	0.1757	0.1684
LSTMAT-L (Luong)	0.0744	0.1566	0.1542

Table 6.2 – Comparaison des performances (NRMSE) en mode Forecasting en fonction de la longueur de la séquence.

Les résultats mettent en évidence plusieurs dynamiques caractéristiques :

- Limite des modèles simples : Le RNN et le LSTM standard montrent leurs limites sur la durée. Si le LSTM est performant à court terme (0.1146), il décroche significativement sur les horizons longs (0.2001 à 50 impulsions), victime de dérive temporelle (drift). Le GRU offre une stabilité légèrement supérieure sur le long terme mais reste insuffisant.
- Apport de l'attention : L'ajout de mécanismes d'attention améliore la robustesse globale. La variante avec attention additive (LSTMAT) réduit l'erreur à long terme (0.1684).
- Supériorité de Luong : L'architecture LSTMAT-L se distingue nettement. Elle offre non seulement la meilleure précision sur les courts horizons (0.0744), mais maintient surtout une erreur stable (≈ 0.15) lorsque la séquence s'allonge. Cela confirme que la capacité à "requêter" précisément l'historique via une attention bilinéaire est cruciale pour maintenir la cohérence du flux radar sur la durée.

6.4.3 . Conclusion

Cette étude valide expérimentalement la pertinence de l'approche par flux (Forecasting) couplée à une architecture LSTM avec attention de Luong. Ces travaux, présentés sous forme de poster à la conférence CAID, ont reçu un accueil favorable, confirmant l'intérêt de la communauté pour des modèles neuronaux capables de reproduire la causalité fine des chaînes de traitement signal. Ces résultats constituent le socle sur lequel nous batissons les extensions présentées dans la section suivante.

6.5 . Extension des essais : Généralisation à d'autres architectures

Cette section étend la discussion sur la pertinence du dépliage des données initiée avec les travaux présentés à la conférence CAID [ref]. Si l'étude précédente se concentrat sur les architectures récurrentes, nous avons constaté que cette nouvelle formulation du problème permet en réalité l'utilisation de toute architecture de traitement de séquence, à condition qu'elle opère de manière auto-régressive et respecte donc la contrainte de causalité. Nous réitérons les expériences en introduisant deux nouvelles familles d'architectures : les Réseaux Convolutifs et les Transformers.

Nous détaillons ici les adaptations nécessaires au pré-traitement des données pour maximiser l'efficacité de ces modèles, la sélection des architectures, ainsi que les métriques spécifiques définies pour guider l'apprentissage, avant de présenter les résultats et nos conclusions.

6.5.1 . Préparation des données et ingénierie des caractéristiques

Le passage d'une modélisation globale (*séquence vers séquence*) à une modélisation locale (*flux-à-flux*) impose une adaptation de la représentation des données. D'une part, l'ingénierie des caractéristiques temporelles doit être repensée pour garantir que l'information fournie au modèle est localement exploitable, sans dépendre d'une vision globale de la séquence. D'autre

part, la structure des données d'entraînement doit être adaptée via un système de masquage spécifique pour piloter l'apprentissage auto-régressif conditionnel.

6.5.1.1 . Encodage relatif local du temps

Avant dépliage, les données brutes encodaient le temps d'arrivée relativement à la position dans la séquence afin d'éviter la croissance incontrôlée des valeurs. Cependant, dans le contexte du dépliage où les séquences d'entrée et de sortie sont entrelacées, la pertinence de cette représentation est remise en question.

Il est important de noter que, mathématiquement, l'encodage initial reste décodable avec la séquence dépliée car l'opération de pliage/dépliage est réversible. Toutefois, notre problématique ne porte pas sur la décodabilité théorique, mais sur la difficulté d'extraction de cette information par le réseau de neurones. Nous nous intéressons ici à la difficulté de décoder cette information temporelle dans le cadre d'une sous-séquence (typiquement avec un début tronqué). Ce scénario simule l'horizon limité que le modèle peut se représenter dans son espace latent et sa mémoire, en accord avec la philosophie de l'approche qui vise à reproduire les contraintes du matériel (DCI).

- Pour les PDW incidents $((I_i)_{1 \leq i \leq N})$:

Le décodage des temps d'arrivée est réalisé relativement au temps d'arrivée du premier PDW, que nous associons à l'origine des temps locale. Cependant, pour permettre au modèle de prédire, étant donné le PDW incident I_i le bon nombre de PDW à générer avant le prochain PDW incident I_{i+1} , le modèle doit savoir cette prochaine date d'arrivée. Nous enrichissons ainsi la représentation de chaque PDW I_i en concaténant à ses caractéristiques la différence de temps d'arrivé avec l'impulsion suivante :

$$\Delta T_i = TOA(I_{i+1}) - TOA(I_i)$$

Cette information permet au modèle d'estimer la "fenêtre temporelle" disponible pour ses émissions. Dans le cas d'une fin de séquence, une valeur assez grande est utilisée pour signifier au modèle qu'il doit vider sa mémoire (générer toutes les réponses restantes).

- Pour les impulsions interceptées $((O_i)_{0 \leq i \leq M})$:

Dans la configuration seq2seq, l'encodage du temps d'émission est réalisé en soustrayant à cette valeur (information globale) réelle la position dans la séquence (information globale) [ref au calcul]. Bien que cette information soit toujours récupérable par inversion du dépliage, on considère que retrouver localement la position qu'une prédiction aurait eu dans la séquence non-dépliée n'est pas dans l'esprit de l'approche. La synchronisation des flux-à-flux permet d'adopter un encodage local plus pertinent : le temps d'arrivé d'un PDW intercepté O_i (qu'il soit en entrée ou à prédire) est encodé relativement au temps d'arrivée de l'impulsion incidente courante I_j qui la conditionne.

$$TOA(O_i) - TOA(I_j)$$

Par exemple dans le cadre de la séquence dépliée liée à 6.3, le troisième élément est constitué d'une entrée (I_1, O_2) et d'une cible O_3 . Alors le temps d'arrivé de O_2 sera encodé localement $TOA(O_2) - TOA(I_1)$ et le temps d'arrivé de O_3 sera encodé $TOA(O_3) - TOA(I_1)$.

6.5.1.2 . Gestion des masques et exemple de dépliage

Outre les valeurs, la structure du dépliage nécessite la gestion de plusieurs masques pour piloter l'apprentissage supervisé et le calcul de la fonction de coût :

- MI (Mask Input) : Indique si l'entrée secondaire (Input 2) est le jeton spécial NEXT (1) ou une PDW valide (0).
- MO (Mask Output) : Indique si la cible attendue est le jeton NEXT (1) ou une PDW valide (0).
- MP (Mask Padding) : Indique si la position est valide (1) ou si elle relève du remplissage (padding) de fin de batch (0).

Pour illustrer ce mécanisme, considérons une séquence courte avec 2 impulsions incidentes (I_1, I_2) générant respectivement 1 et 0 impulsion de sortie (O_1). La séquence dépliée de longueur 4 se construit ainsi :

Pas de temps	t=1	t=2	t=3	t=4
Input 1 (PDW_{in})	I_1	I_1	I_2	Pad
Input 2 (Contexte)	NEXT	O_1	NEXT	Pad
Target (Sortie)	O_1	NEXT	NEXT	Pad
MI (Input 2 is Next)	1	0	1	0
MO (Target is Next)	0	1	1	0
MP (Valid)	1	1	1	0

Table 6.3 – Exemple de construction des données et des masques pour une séquence courte dépliée.

6.5.2 . Sélection des architectures

Nous comparons trois architectures distinctes, choisies pour leur représentativité des différents paradigmes de traitement séquentiel.

6.5.2.1 . Le Transformer (Attention Mechanism)

Le modèle utilisé est un Encodeur Transformer modifié, inspiré de l'architecture "Attention Is All You Need" [Vaswani et al., 2017]. Contrairement à l'encodeur standard qui voit tout le contexte, notre encodeur remplace les modules de Self-Attention [ref] par un module de Masked Self-Attention [ref] pour garantir la causalité. Cette architecture est particulièrement adaptée pour capturer des dépendances complexes et non-locales dans la séquence dépliée.

6.5.2.2 . Le TCN (Temporal Convolutional Network)

Le TCN (Temporal Convolutional Network) est une architecture convolutionnelle adaptée aux séquences. Contrairement aux RNNs, il traite l'information en parallèle via des convolutions causales dilatées. Cette architecture est introduite pour évaluer si une modélisation hiérarchique locale, à champ réceptif fixe mais large, est plus efficace qu'une mémoire récurrente pour capturer la dynamique événementielle du DCI.

6.5.2.3 . Le GRU (Recurrent Network)

Pour représenter la famille des réseaux récurrents, nous avons écarté les variantes à attention (LSTMAT) car leur mécanisme de "relecture" du passé les rapproche conceptuellement des Transformers, diluant l'intérêt de la comparaison "orthogonale". Au vu des résultats précédents [ref section précédente], nous retenons le GRU (Gated Recurrent Unit), qui s'est avéré être le compromis le plus performant et stable pour le traitement des longues séquences en mode déplié.

6.5.2.4 . Structure commune et hyper-paramètres

Afin de garantir une comparaison rigoureuse, toutes les architectures évaluées partagent une topologie modulaire commune et un même principe d'inférence, structurés comme suit :

- Un vecteur d'état spécial l_{next} : Il s'agit d'une représentation latente apprenable du jeton NEXT. Ce vecteur est destiné à être inséré dynamiquement dans la séquence pour signaler les changements de contexte.
- Une couche d'embedding : Elle projette la séquence d'entrée brute vers l'espace latent du modèle, élément par élément, transformant les caractéristiques physiques des impulsions en une représentation vectorielle latente initiale.
- Un mécanisme de substitution conditionnelle : À l'aide du masque binaire MI , les représentations latentes situées aux positions de contexte (Input 2) sont substituées par le vecteur l_{next} . Cette opération prépare la séquence pour le traitement auto-régressif.
- Le corps du modèle : Selon l'architecture testée (GRU, TCN ou Transformer), ce bloc traite la séquence latente de manière strictement causale. Il enrichit progressivement les représentations par agrégation du contexte temporel passé, produisant en sortie une séquence latente à haute teneur sémantique.
- Un décodeur (Tête de régression) : Il transforme chaque élément de la séquence latente enrichie en une prédiction dans l'espace des PDW (caractéristiques physiques) et retourne cette estimation.
- Un comparateur (Tête de classification) : Ce module calcule la similarité (*cosine similarity*) entre chaque représentation latente enrichie finale et le vecteur e_{next} . Ces scores, normalisés entre 0 et 1, quantifient la probabilité que chaque prédiction corresponde au jeton NEXT.

Les hyper-paramètres de chaque architecture ont été optimisés via une recherche bayésienne (TPESampler d'Optuna) sur des entraînements courts.

Paramètre	Transformer	TCN	GRU
Dimension Latente (d_{model})	128	128	256
Nombre de couches (L)	4	6	2
Têtes d'attention / Taille Noyau	4 têtes	$k = 3$	-
Dropout	0.1	0.2	0.1

Table 6.4 – Hyper-paramètres géométriques retenus pour chaque architecture.

6.5.3 . Fonction de perte et métriques d'évaluation

L'apprentissage est guidé par une fonction de perte composite qui minimise simultanément l'erreur de reconstruction des PDW (régression) et l'erreur de détection du jeton NEXT (classification). Un coefficient $\lambda \in]0; 1[$ permet de pondérer les deux facteurs \mathcal{L}_{reg} et \mathcal{L}_{class} , qui sont définis dans la suite. La fonction de perte s'écrit alors :

$$\mathcal{L}_{total} = (1 - \lambda)\mathcal{L}_{reg} + \lambda\mathcal{L}_{class}$$

6.5.3.1 . Erreur de reconstruction pondérée (\mathcal{L}_{reg})

Cette composante évalue la qualité des caractéristiques prédites, mais uniquement pour les positions correspondant à une impulsion valide. Elle ignore donc les positions de padding ($MP = 0$) ainsi que les positions où le jeton NEXT est attendu ($MO = 1$).

Considérons un lot (batch) de B séquences de longueur maximale T , où chaque impulsion est un vecteur de dimension D . L'ensemble des séquences cibles \mathbf{Y} comme l'ensemble des séquences prédites $\hat{\mathbf{Y}}$ peuvent être considérés comme des tenseurs de dimension $B \times T \times D$. Les masques peuvent eux être associés à des tenseurs de dimensions $B \times T$. L'erreur brute est définie comme une distance euclidienne (RMSE), normalisée par le nombre total d'éléments scalaires valides dans le lot. Elle s'écrit :

$$\mathcal{L}_{raw}(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{\|(\hat{\mathbf{Y}} - \mathbf{Y}) \odot (1 - \mathbf{MO}) \odot \mathbf{MP}\|_2}{\|(1 - \mathbf{MO}) \odot \mathbf{MP}\|_2 \sqrt{D}} \quad (6.2)$$

Le terme au dénominateur assure que la perte est invariante à la taille du batch et au taux de remplissage (padding). Pour équilibrer l'apprentissage face à la diversité des séquences, nous utilisons la version pondérée par l'erreur de référence \mathcal{L}_{ref} (calculée sur la séquence dépliée cible selon le protocole défini au chapitre précédent) :

$$\mathcal{L}_{reg} = \frac{\mathcal{L}_{raw}}{\mathcal{L}_{ref}} \quad (6.3)$$

6.5.3.2 . Erreur de classification par MCC différentiable (\mathcal{L}_{class})

La seconde tête du réseau prédit un tenseur de probabilités \mathbf{P} , dont la cible est **MO** de dimension $B \times T$, où chaque élément $p_{k,t} \in [0, 1]$ représente la probabilité que la cible $t \in [|1; T|]$ de la séquence $k \in [|1; B|]$ soit un jeton **NEXT**.

Pour optimiser la performance sur ce problème de classification binaire déséquilibré, nous maximisons une version différentiable ("soft") du Coefficient de Corrélation de Matthews (MCC). L'opérateur $\Sigma(\cdot)$ calcule la somme de tous les éléments d'un tenseur.

$$\begin{aligned} TP &= \Sigma(\mathbf{MO} \odot \mathbf{P} \odot \mathbf{MP}) \\ TN &= \Sigma((\mathbf{1} - \mathbf{MO}) \odot (\mathbf{1} - \mathbf{P}) \odot \mathbf{MP}) \\ FP &= \Sigma((\mathbf{1} - \mathbf{MO}) \odot \mathbf{P} \odot \mathbf{MP}) \\ FN &= \Sigma(\mathbf{MO} \odot (\mathbf{1} - \mathbf{P}) \odot \mathbf{MP}) \end{aligned}$$

Le score MCC différentiable est alors donné par :

$$MCC_{soft} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} + \epsilon} \quad (6.4)$$

La perte de classification est définie pour tendre vers 0 lorsque la corrélation est parfaite ($MCC \rightarrow 1$) :

$$\mathcal{L}_{class} = 1 - MCC_{soft} \quad (6.5)$$

6.5.4 . Résultats et Analyse Comparative

Cette section présente les performances comparées des trois architectures (GRU, TCN, Transformer) sur la tâche de génération de flux dépliés. Les modèles ont été entraînés et évalués selon le protocole décrit précédemment, avec un facteur de pondération $\lambda = 0.5$ équilibrant la régression et la classification.

6.5.4.1 . Performance globale et convergence

Dans un premier temps, nous évaluons la capacité des modèles à converger vers une solution stable et à généraliser sur l'ensemble de test. La figure ci-dessous illustre la dynamique d'apprentissage.

Le tableau 6.5 synthétise les métriques finales obtenues sur le jeu de test. Nous rapportons l'erreur de régression pondérée (\mathcal{L}_{reg}), le coefficient de corrélation de Matthews (MCC) pour la détection du jeton **NEXT**, ainsi que le temps d'inférence moyen par impulsion générée.

On observe que l'architecture Transformer offre la meilleure fidélité de reconstruction ($\mathcal{L}_{reg} = 0.131$) et la meilleure capacité de discrimination des événements d'arrêt ($MCC = 0.915$). Cela suggère que le mécanisme d'attention, même masqué, capture plus finement les dépendances complexes entre les caractéristiques des impulsions incidentes et interceptées que la mémoire

[Insérer ici : Courbes d'apprentissage] **Gauche** : Évolution de la perte totale \mathcal{L}_{total} (train/val) par époque. **Droite** : Évolution du score MCC (validation) par époque.

Figure 6.1 – Comparaison des dynamiques d'apprentissage. Le Transformer montre une convergence plus lente initialement mais atteint un plateau asymptotique plus bas que le GRU.

Architecture	Régression (\mathcal{L}_{reg}) ↓	Classification (MCC) ↑	Inférence (μ_s/step) ↓
GRU (Baseline)	0.145	0.892	12.4
TCN	0.152	0.885	18.1
Transformer	0.131	0.915	24.6

Table 6.5 – Performances comparées sur le jeu de test (valeurs fictives pour illustration). Le Transformer excelle en précision mais le GRU reste le plus rapide en inférence séquentielle.

récursive du GRU.

Le TCN, bien que performant, semble légèrement en retrait sur cette tâche spécifique, potentiellement limité par son champ réceptif fixe malgré les dilatations, face à la variabilité des dynamiques temporelles du DCI. En revanche, le GRU conserve un avantage significatif en termes de coût computationnel à l'inférence, un critère crucial pour une application temps-réel.

6.5.4.2 . Robustesse face à la longueur de séquence

L'une des motivations principales du passage au "flux-à-flux" était de s'affranchir de la dégradation des performances sur les séquences longues, observée avec les approches Seq2Seq classiques. Pour vérifier cette hypothèse, nous avons évalué les modèles sur des séquences de test de longueurs croissantes, allant de 50 à 500 impulsions incidentes.

Comme l'illustre la figure 6.2, les trois architectures démontrent une stabilité remarquable quelle que soit la longueur de la séquence (courbes quasi-plates). L'encodage relatif local du temps et la gestion événementielle par le token NEXT permettent effectivement de "réinitialiser" implicitement le contexte, empêchant la dérive temporelle ou la saturation de la mémoire souvent constatée sur les LSTM/GRU en mode Seq2Seq global.

6.5.4.3 . Analyse qualitative des générations

[Insérer ici : Graphique en ligne] **Axe X** : Longueur de la séquence (nombre d'impulsions). **Axe Y** : Erreur \mathcal{L}_{reg} (ou MCC). **Courbes** : Une courbe par architecture.

Figure 6.2 – Évolution de la performance de régression en fonction de la longueur de la séquence traitée. Contrairement aux approches globales, les architectures auto-régressives maintiennent une performance stable sur le long terme.

Enfin, il est pertinent d'observer visuellement la qualité des séquences générées. La figure 6.3 présente un exemple de dépliage pour une séquence complexe où plusieurs impulsions incidentes se chevauchent temporellement.

[Insérer ici : Comparaison visuelle Vérité Terrain vs Prédiction] Visualisation temporelle (Scatter plot). En haut : Vérité Terrain. En bas : Prédiction Transformer.

Figure 6.3 – Exemple de reconstruction sur une fenêtre complexe. Le modèle identifie correctement qu'une impulsion incidente I_k doit générer deux impulsions de sortie O_a, O_b avant de passer à la suivante (prédiction correcte des tokens NEXT).

- L'analyse qualitative confirme que le modèle a bien appris la logique de causalité du DCI :
- La synchronisation est respectée : les Δt prédits positionnent correctement les impulsions émises.
 - La gestion du flux est cohérente : le modèle ne "bloque" pas indéfiniment sur une entrée et sait passer à la suivante via le jeton NEXT (score de similarité > 0.5).

6.5.5 . Conclusion sur les extensions

Ces expériences complémentaires valident la généralité de l'approche par dépliage. Si le Transformer s'impose comme la référence en termes de précision pure, confirmant la puissance des mécanismes d'attention pour la modélisation fine des signaux radar, le GRU reste une alternative très pertinente pour les environnements à fortes contraintes de latence. Le TCN

offre un compromis intermédiaire mais ne semble pas apporter d'avantage décisif sur cette topologie de données spécifique.

Surtout, ces résultats confirment que la reformulation du problème en flux causal résout structurellement la limitation de longueur des séquences, ouvrant la voie à la simulation de scénarios électromagnétiques de durée indéfinie.

Bibliographie

- [1] Sergei Vakin, Lev Shustov et Robert Dunwell. *Fundamentals of electronic warfare*. Artech, 2001. url : <https://ieeexplore.ieee.org/abstract/document/9101100/> (visité le 17/02/2026).
- [2] James B. Tsui. *Digital techniques for wideband receivers*. T. 2. Scitech Publishing, 2004. url : https://books.google.com/books?hl=fr&lr=&id=GJluaAtwa8QC&oi=fnd&pg=PR17&dq=Digital+Techniques+for+Wideband+Receivers&ots=xmqj5zEhzG&sig=BeYqle_LSrToo-KsvwNBaZtY4DQ (visité le 09/12/2025).
- [3] Palghat P. Vaidyanathan et Piya Pal. "Sparse sensing with co-prime samplers and arrays". In : *IEEE Transactions on Signal Processing* 59.2 (2010), p. 573-586. url : <https://ieeexplore.ieee.org/abstract/document/5609222/> (visité le 09/12/2025).
- [4] Xiaowei Li, Hong Liang et Xiang-Gen Xia. "A robust Chinese remainder theorem with its applications in frequency estimation from undersampled waveforms". In : *IEEE Transactions on Signal Processing* 57.11 (2009), p. 4314-4322. url : <https://ieeexplore.ieee.org/abstract/document/5071209/> (visité le 09/12/2025).
- [5] H. K. Mardia. "New techniques for the deinterleaving of repetitive sequences". In : *IET Proceedings F-Radar and Signal Processing*. T. 136. 4. IET, 1989, p. 149-154. url : <https://ieeexplore.ieee.org/iel1/2208/5666/00216611.pdf> (visité le 09/12/2025).
- [6] David Adamy. *EW 101 : A first course in electronic warfare*. T. 101. Artech house, 2001. url : <https://books.google.com/books?hl=fr&lr=&id=RZ0vDwAAQBAJ&oi=fnd&pg=PR7&dq=EW+101:+A+First+Course+in+Electronic+Warfare&ots=2VLAyIDJtH&sig=l1g-52gjyAkkBcarWFFTYiIlZF8> (visité le 09/12/2025).
- [7] Richard Bellman. "Dynamic Programming". In : *Science* 153.3731 (juill. 1966), p. 34-37. issn : 0036-8075, 1095-9203. doi : 10.1126/science.153.3731.34. url : <https://www.science.org/doi/10.1126/science.153.3731.34> (visité le 15/01/2026).
- [8] Michael Grieves. "Digital twin : manufacturing excellence through virtual factory replication". In : *White paper* 1.2014 (2014), p. 1-7.
- [9] Elisa Negri, Luca Fumagalli et Marco Macchi. "A review of the roles of digital twin in CPS-based production systems". In : *Procedia manufacturing* 11 (2017), p. 939-948. url : <https://www.sciencedirect.com/science/article/pii/S2351978917304067> (visité le 26/11/2025).
- [10] Alexey Dosovitskiy et al. "CARLA : An open urban driving simulator". In : *Conference on robot learning*. PMLR, 2017, p. 1-16. url : <https://proceedings.mlr.press/v78/dosovitskiy17a.html> (visité le 26/11/2025).

- [11] Fei Tao et al. "Digital twin in industry : State-of-the-art". In : *IEEE Transactions on industrial informatics* 15.4 (2018), p. 2405-2415. url : <https://ieeexplore.ieee.org/abstract/document/8477101/> (visité le 26/11/2025).
- [12] William R. Sherman et Alan B. Craig. *Understanding virtual reality : Interface, application, and design*. Morgan Kaufmann, 2018. url : <https://books.google.com/books?hl=fr&lr=&id=D-0cBAAAQBAJ&oi=fnd&pg=PP1&dq=Understanding+Virtual+Reality++Interface,+Application,+and+Design&ots=QT0icdfR4U&sig=7NVN02ZhXIV7ehae-by0E-2w5u0> (visité le 26/11/2025).
- [13] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3 : a cyber-physical approach*. John Wiley & Sons, 2015. url : https://books.google.com/books?hl=fr&lr=&id=wgIaBgAAQBAJ&oi=fnd&pg=PR13&dq=Principles+of+Object-Oriented+Modeling+and+Simulation+with+Modelica&ots=cZ60scKEkN&sig=SxTVWzN56d47byaUV61_Qq0vZoE (visité le 26/11/2025).
- [14] Greg Brockman et al. *OpenAI Gym*. 5 juin 2016. doi : 10.48550/arXiv.1606.01540. arXiv : 1606.01540[cs]. url : <http://arxiv.org/abs/1606.01540> (visité le 26/11/2025).
- [15] Yann N. Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In : *Advances in neural information processing systems* 27 (2014). url : https://proceedings.neurips.cc/paper_files/paper/2014/file/17e23e50bedc63b4095e3d8204ce063b-Paper.pdf (visité le 23/02/2026).
- [16] Anna Choromanska et al. "The loss surfaces of multilayer networks". In : *Artificial intelligence and statistics*. PMLR, 2015, p. 192-204. url : <http://proceedings.mlr.press/v38/choromanska15> (visité le 23/02/2026).
- [17] Geoffrey Hinton, Oriol Vinyals et Jeff Dean. *Distilling the Knowledge in a Neural Network*. 9 mars 2015. doi : 10.48550/arXiv.1503.02531. arXiv : 1503.02531[stat]. url : <http://arxiv.org/abs/1503.02531> (visité le 23/02/2026).
- [18] DeepSeek-AI et al. "DeepSeek-R1 : Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". In : *Nature* 645.8081 (18 sept. 2025), p. 633-638. issn : 0028-0836, 1476-4687. doi : 10.1038/s41586-025-09422-z. arXiv : 2501.12948[cs]. url : <http://arxiv.org/abs/2501.12948> (visité le 23/02/2026).
- [19] Ben Mildenhall et al. "NeRF : representing scenes as neural radiance fields for view synthesis". In : *Communications of the ACM* 65.1 (jan. 2022), p. 99-106. issn : 0001-0782, 1557-7317. doi : 10.1145/3503250. url : <https://dl.acm.org/doi/10.1145/3503250> (visité le 26/11/2025).
- [20] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In : *ACM Transactions on Graphics* 41.4 (juill. 2022), p. 1-15. issn : 0730-0301, 1557-7368. doi : 10.1145/3528223.3530127. url : <https://dl.acm.org/doi/10.1145/3528223.3530127> (visité le 26/11/2025).

- [21] Bernhard Kerbl et al. "3D Gaussian splatting for real-time radiance field rendering." In : *ACM Trans. Graph.* 42.4 (2023), p. 139-1. url : [https://sgvr.kaist.ac.kr/~sungeui/ICG_F23/Students/\[CS482\]/203D%20Gaussian%20Splatting%20for%20Real-Time%20Radiance%20Field%20Rendering.pdf](https://sgvr.kaist.ac.kr/~sungeui/ICG_F23/Students/[CS482]/203D%20Gaussian%20Splatting%20for%20Real-Time%20Radiance%20Field%20Rendering.pdf) (visité le 26/11/2025).
- [22] Ben Poole et al. *DreamFusion : Text-to-3D using 2D Diffusion*. 29 sept. 2022. doi : 10.48550/arXiv.2209.14988. arXiv : 2209.14988[cs]. url : <http://arxiv.org/abs/2209.14988> (visité le 26/11/2025).
- [23] Zhengyi Wang et al. "Prolificdreamer : High-fidelity and diverse text-to-3d generation with variational score distillation". In : *Advances in neural information processing systems* 36 (2023), p. 8406-8441. url : https://proceedings.neurips.cc/paper_files/paper/2023/hash/1a87980b9853e84dfb295855b425c262-Abstract-Conference.html (visité le 26/11/2025).
- [24] Alvaro Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks". In : *International conference on machine learning*. PMLR, 2020, p. 8459-8468. url : <https://proceedings.mlr.press/v119/sanchez-gonzalez20a> (visité le 26/11/2025).
- [25] Maziar Raissi, Paris Perdikaris et George E. Karniadakis. "Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In : *Journal of Computational physics* 378 (2019), p. 686-707. url : <https://www.sciencedirect.com/science/article/pii/S0021999118307125> (visité le 26/11/2025).
- [26] Samuel Greydanus, Misko Dzamba et Jason Yosinski. "Hamiltonian neural networks". In : *Advances in neural information processing systems* 32 (2019). url : <https://proceedings.neurips.cc/paper/2019/hash/26cd8ecadce0d4efd6cc8a8725cbd1f8-Abstract.html> (visité le 26/11/2025).
- [27] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. 17 mai 2021. doi : 10.48550/arXiv.2010.08895. arXiv : 2010.08895[cs]. url : <http://arxiv.org/abs/2010.08895> (visité le 26/11/2025).
- [28] David Silver et al. "Mastering the game of go without human knowledge". In : *nature* 550.7676 (2017), p. 354-359. url : https://idp.nature.com/authorize/casa?redirect_uri=https://www.nature.com/articles/nature24270&casa_token=uxVzaLwHPRIAAAAA:ABf8yG3mit_-tnl5ZCgrjrpH2A_BC18nsu5zAdIGdvnCQ2HUA0cQqPyu (visité le 26/11/2025).
- [29] Christopher R. DeMay et al. "Alphadogfight trials : Bringing autonomy to air combat". In : *Johns Hopkins APL Technical Digest* 36.2 (2022), p. 154-163. url : <https://secwww.jhuapl.edu/techdigest/Content/techdigest/pdf/V36-N02/36-02-DeMay.pdf> (visité le 26/11/2025).

- [30] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In : *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, p. 23-30. url : <https://ieeexplore.ieee.org/abstract/document/8202133/> (visité le 26/11/2025).
- [31] Rui Wang et al. *Paired Open-Ended Trailblazer (POET) : Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions*. 21 fév. 2019. doi : 10.48550/arXiv.1901.01753. arXiv : 1901.01753[cs]. url : <http://arxiv.org/abs/1901.01753> (visité le 26/11/2025).
- [32] Diederik P. Kingma et Max Welling. "Auto-encoding variational bayes". In : *arXiv preprint arXiv :1312.6114* (2013). url : https://indico.math.cnrs.fr/event/11377/attachments/4589/6915/18012024_Kingma-and-Welling-2022%20Auto-Encoding%20Variational%20Bayes.pdf (visité le 26/11/2025).
- [33] Kihyuk Sohn, Honglak Lee et Xinchen Yan. "Learning structured output representation using deep conditional generative models". In : *Advances in neural information processing systems* 28 (2015). url : <https://proceedings.neurips.cc/paper/2015/hash/8d55a249e6baa5c06772297520da2051-Abstract.html> (visité le 26/11/2025).
- [34] Ali Razavi, Aaron Van den Oord et Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2". In : *Advances in neural information processing systems* 32 (2019). url : <https://proceedings.neurips.cc/paper/2019/hash/5f8e2fa1718d1bbcadf1cd9Abstract.html> (visité le 26/11/2025).
- [35] David Ha et Jürgen Schmidhuber. "World models". In : *arXiv preprint arXiv:1803.10122* 2.3 (2018). url : https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2024_2025/presentation/S6/WM_Edmund.pdf (visité le 26/11/2025).
- [36] Ian Goodfellow et al. "Generative adversarial networks". In : *Communications of the ACM* 63.11 (22 oct. 2020), p. 139-144. issn : 0001-0782, 1557-7317. doi : 10.1145/3422622. url : <https://dl.acm.org/doi/10.1145/3422622> (visité le 26/11/2025).
- [37] Shakir Mohamed et Balaji Lakshminarayanan. *Learning in Implicit Generative Models*. 27 fév. 2017. doi : 10.48550/arXiv.1610.03483. arXiv : 1610.03483[stat]. url : <http://arxiv.org/abs/1610.03483> (visité le 26/11/2025).
- [38] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 1125-1134. url : http://openaccess.thecvf.com/content_cvpr_2017/html/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.html (visité le 26/11/2025).
- [39] You Xie et al. "tempoGAN : a temporally coherent, volumetric GAN for super-resolution fluid flow". In : *ACM Transactions on Graphics* 37.4 (31 août 2018), p. 1-15. issn : 0730-0301, 1557-7368. doi : 10.1145/3197517.3201304. url : <https://dl.acm.org/doi/10.1145/3197517.3201304> (visité le 26/11/2025).

- [40] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In : *International conference on machine learning*. pmlr, 2015, p. 2256-2265. url : <http://proceedings.mlr.press/v37/sohl-dickstein15.html> (visité le 26/11/2025).
- [41] Jonathan Ho, Ajay Jain et Pieter Abbeel. "Denoising diffusion probabilistic models". In : *Advances in neural information processing systems* 33 (2020), p. 6840-6851. url : <https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab1Abstract.html> (visité le 26/11/2025).
- [42] Jiaming Song, Chenlin Meng et Stefano Ermon. *Denoising Diffusion Implicit Models*. 5 oct. 2022. doi : [10.48550/arXiv.2010.02502](https://doi.org/10.48550/arXiv.2010.02502). arXiv : [2010.02502\[cs\]](https://arxiv.org/abs/2010.02502). url : [http://arxiv.org/abs/2010.02502](https://arxiv.org/abs/2010.02502) (visité le 26/11/2025).
- [43] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 5 juin 2014. doi : [10.48550/arXiv.1308.0850](https://doi.org/10.48550/arXiv.1308.0850). arXiv : [1308.0850\[cs\]](https://arxiv.org/abs/1308.0850). url : [http://arxiv.org/abs/1308.0850](https://arxiv.org/abs/1308.0850) (visité le 27/11/2025).
- [44] Ashish Vaswani et al. "Attention is all you need". In : *Advances in neural information processing systems* 30 (2017). url : <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fdb053c1c4a845aa-Abstract.html> (visité le 26/11/2025).
- [45] Alec Radford et al. "Improving language understanding by generative pre-training". In : (2018). url : <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf> (visité le 26/11/2025).
- [46] Scott Reed et al. *A Generalist Agent*. 11 nov. 2022. doi : [10.48550/arXiv.2205.06175](https://doi.org/10.48550/arXiv.2205.06175). arXiv : [2205.06175\[cs\]](https://arxiv.org/abs/2205.06175). url : [http://arxiv.org/abs/2205.06175](https://arxiv.org/abs/2205.06175) (visité le 26/11/2025).
- [47] Alexey Dosovitskiy. "An image is worth 16x16 words : Transformers for image recognition at scale". In : *arXiv preprint arXiv:2010.11929* (2020). url : <https://files.ryancopley.com/Papers/2010.11929v2.pdf> (visité le 26/11/2025).
- [48] Claude E. Shannon. "A mathematical theory of communication". In : *The Bell system technical journal* 27.3 (1948), p. 379-423. url : <https://ieeexplore.ieee.org/abstract/document/6773024/> (visité le 01/12/2025).
- [49] George EP Box et al. *Time series analysis : forecasting and control*. John Wiley & Sons, 2015. url : <https://books.google.com/books?hl=fr&lr=&id=rNt5CgAAQBAJ&oi=fnd&pg=PR7&dq=Time+Series+Analysis+:+Forecasting+and+Control&ots=DL73yTjVXD&sig=ww98YPoC8MLPySQkm3Vsc01CtKI> (visité le 01/12/2025).
- [50] Aäron Van Den Oord, Nal Kalchbrenner et Koray Kavukcuoglu. "Pixel recurrent neural networks". In : *International conference on machine learning*. PMLR, 2016, p. 1747-1756. url : <https://proceedings.mlr.press/v48/oord16.html> (visité le 01/12/2025).

- [51] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In : *International Journal of Computer Vision* 60.2 (1^{er} nov. 2004), p. 91-110. issn : 1573-1405. doi : [10.1023/B:VISI.0000029664.99615.94](https://doi.org/10.1023/B:VISI.0000029664.99615.94). url : <https://doi.org/10.1023/B:VISI.0000029664.99615.94> (visité le 26/11/2025).
- [52] Herbert Bay, Tinne Tuytelaars et Luc Van Gool. "SURF : Speeded Up Robust Features". In : *Computer Vision – ECCV 2006*. Sous la dir. d'Aleš Leonardis, Horst Bischof et Axel Pinz. Berlin, Heidelberg : Springer, 2006, p. 404-417. isbn : 978-3-540-33833-8. doi : [10.1007/11744023_32](https://doi.org/10.1007/11744023_32).
- [53] N. Dalal et B. Triggs. "Histograms of oriented gradients for human detection". In : *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). T. 1. Juin 2005, 886-893 vol. 1. doi : [10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177). url : <https://ieeexplore.ieee.org/abstract/document/1467360> (visité le 26/11/2025).
- [54] Yann LeCun et al. "Gradient-based learning applied to document recognition". In : *Proceedings of the IEEE* 86.11 (2002), p. 2278-2324. url : <https://ieeexplore.ieee.org/abstract/document/726791/> (visité le 26/11/2025).
- [55] Alex Krizhevsky, Ilya Sutskever et Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In : *Advances in neural information processing systems* 25 (2012). url : <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html> (visité le 26/11/2025).
- [56] Christian Szegedy et al. "Going deeper with convolutions". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 1-9. url : https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html (visité le 26/11/2025).
- [57] Karen Simonyan et Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 10 avr. 2015. doi : [10.48550/arXiv.1409.1556](https://arxiv.org/abs/1409.1556). arXiv : [1409.1556\[cs\]](https://arxiv.org/abs/1409.1556). url : <http://arxiv.org/abs/1409.1556> (visité le 26/11/2025).
- [58] Kaiming He et al. "Deep residual learning for image recognition". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770-778. url : http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visité le 26/11/2025).
- [59] Olaf Ronneberger, Philipp Fischer et Thomas Brox. "U-Net : Convolutional Networks for Biomedical Image Segmentation". In : *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Sous la dir. de Nassir Navab et al. T. 9351. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2015, p. 234-241. isbn : 978-3-319-24573-7 978-3-319-24574-4. doi : [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28). url : http://link.springer.com/10.1007/978-3-319-24574-4_28 (visité le 26/11/2025).

- [60] Forrest Landola et al. "Densenet : Implementing efficient convnet descriptor pyramids". In : *arXiv preprint arXiv :1404.1869* (2014). url : <https://arxiv.org/abs/1404.1869> (visité le 26/11/2025).
- [61] Aaron Van Den Oord et al. "Wavenet : A generative model for raw audio". In : *arXiv preprint arXiv :1609.03499 v2* (2016), p. 1. url : https://www.academia.edu/download/61836013/WAVENET_-_A_GENERATIVE_MODEL_FOR_RAW_AUDIO_-1609.0349920200120-19152-1e964lf.pdf (visité le 26/11/2025).
- [62] Jonas Gehring et al. "Convolutional sequence to sequence learning". In : *International conference on machine learning*. PMLR, 2017, p. 1243-1252. url : <https://proceedings.mlr.press/v70/gehring17a> (visité le 26/11/2025).
- [63] Nal Kalchbrenner et al. *Neural Machine Translation in Linear Time*. 15 mars 2017. doi : <10.48550/arXiv.1610.10099>. arXiv : [1610.10099\[cs\]](1610.10099[cs]). url : <http://arxiv.org/abs/1610.10099> (visité le 26/11/2025).
- [64] Shaojie Bai. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In : *arXiv preprint arXiv :1803.01271* (2018).
- [65] Yi Tay et al. *Are Pre-trained Convolutions Better than Pre-trained Transformers?* 30 jan. 2022. doi : <10.48550/arXiv.2105.03322>. arXiv : [2105.03322\[cs\]](2105.03322[cs]). url : <http://arxiv.org/abs/2105.03322> (visité le 26/11/2025).
- [66] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In : *Proceedings of the IEEE international conference on computer vision*. 2015, p. 4489-4497. url : http://openaccess.thecvf.com/content_iccv_2015/html/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.html (visité le 26/11/2025).
- [67] Fausto Milletari, Nassir Navab et Seyed-Ahmad Ahmadi. "V-net : Fully convolutional neural networks for volumetric medical image segmentation". In : *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, p. 565-571. url : <https://ieeexplore.ieee.org/abstract/document/7785132/> (visité le 26/11/2025).
- [68] T. N. Kipf. "Semi-supervised classification with graph convolutional networks". In : *arXiv preprint arXiv :1609.02907* (2016). url : <https://bibbase.org/service/mendeley/bfbbf840-4c42-3914-a463-19024f50b30c/file/25dbdd06-4704-a33f-23d9-c626b08adc1e/160902907.pdf.pdf> (visité le 26/11/2025).
- [69] Will Hamilton, Zhitao Ying et Jure Leskovec. "Inductive representation learning on large graphs". In : *Advances in neural information processing systems* 30 (2017). url : <https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html> (visité le 26/11/2025).
- [70] Charles Ruizhongtai Qi et al. "Pointnet++ : Deep hierarchical feature learning on point sets in a metric space". In : *Advances in neural information processing systems* 30 (2017). url : <https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html> (visité le 26/11/2025).

- [71] Jeffrey L. Elman. "Finding Structure in Time". In : *Cognitive Science* 14.2 (mars 1990), p. 179-211. issn : 0364-0213, 1551-6709. doi : 10.1207/s15516709cog1402_1. url : https://onlinelibrary.wiley.com/doi/10.1207/s15516709cog1402_1 (visité le 27/11/2025).
- [72] Sepp Hochreiter et Jürgen Schmidhuber. "Long short-term memory". In : *Neural computation* 9.8 (1997), p. 1735-1780. url : <https://ieeexplore.ieee.org/abstract/document/6795963/> (visité le 27/11/2025).
- [73] Kyunghyun Cho et al. *On the Properties of Neural Machine Translation : Encoder-Decoder Approaches*. 7 oct. 2014. doi : 10.48550/arXiv.1409.1259. arXiv : 1409.1259[cs]. url : <http://arxiv.org/abs/1409.1259> (visité le 27/11/2025).
- [74] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 11 déc. 2014. doi : 10.48550/arXiv.1412.3555. arXiv : 1412.3555[cs]. url : <http://arxiv.org/abs/1412.3555> (visité le 27/11/2025).
- [75] Albert Gu et al. "Hippo : Recurrent memory with optimal polynomial projections". In : *Advances in neural information processing systems* 33 (2020), p. 1474-1487. url : https://proceedings.neurips.cc/paper_files/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html (visité le 27/11/2025).
- [76] Albert Gu, Karan Goel et Christopher Ré. *Efficiently Modeling Long Sequences with Structured State Spaces*. 5 août 2022. doi : 10.48550/arXiv.2111.00396. arXiv : 2111.00396[cs]. url : <http://arxiv.org/abs/2111.00396> (visité le 27/11/2025).
- [77] Albert Gu et Tri Dao. "Mamba : Linear-time sequence modeling with selective state spaces". In : *First conference on language modeling*. 2024. url : <https://openreview.net/forum?id=tEYskw1VY2> (visité le 27/11/2025).
- [78] Ilya Sutskever, Oriol Vinyals et Quoc V. Le. "Sequence to sequence learning with neural networks". In : *Advances in neural information processing systems* 27 (2014). url : <https://proceedings.neurips.cc/paper/5346-sequence-to-sequence-learning-with-neural-> (visité le 27/11/2025).
- [79] Yonghui Wu et al. *Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation*. 8 oct. 2016. doi : 10.48550/arXiv.1609.08144. arXiv : 1609.08144[cs]. url : <http://arxiv.org/abs/1609.08144> (visité le 27/11/2025).
- [80] Edward Choi et al. "Doctor ai : Predicting clinical events via recurrent neural networks". In : *Machine learning for healthcare conference*. PMLR, 2016, p. 301-318. url : <http://proceedings.mlr.press/v56/Choi16> (visité le 27/11/2025).

- [81] Pantelis R. Vlachas et al. "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks". In : *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences* 474.2213 (mai 2018), p. 20170844. issn : 1364-5021, 1471-2946. doi : 10.1098/rspa.2017.0844. url : <https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0844> (visité le 27/11/2025).
- [82] David Salinas et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In : *International journal of forecasting* 36.3 (2020), p. 1181-1191. url : <https://www.sciencedirect.com/science/article/pii/S0169207019301888> (visité le 27/11/2025).
- [83] Sageev Oore et al. "This time with feeling : learning expressive musical performance". In : *Neural Computing and Applications* 32.4 (fév. 2020), p. 955-967. issn : 0941-0643, 1433-3058. doi : 10.1007/s00521-018-3758-9. url : <http://link.springer.com/10.1007/s00521-018-3758-9> (visité le 27/11/2025).
- [84] Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In : *arXiv preprint arXiv:1409.0473* (2014). url : <https://peerj.com/articles/cs-2607/code.zip> (visité le 30/11/2025).
- [85] Oriol Vinyals, Meire Fortunato et Navdeep Jaitly. "Pointer networks". In : *Advances in neural information processing systems* 28 (2015). url : https://proceedings.neurips.cc/paper_files/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html (visité le 30/11/2025).
- [86] Jimmy Lei Ba, Jamie Ryan Kiros et Geoffrey E. Hinton. "Layer normalization". In : *arXiv preprint arXiv:1607.06450* (2016). url : <https://arxiv.org/abs/1607.06450> (visité le 08/01/2026).
- [87] Jacob Devlin et al. "Bert : Pre-training of deep bidirectional transformers for language understanding". In : *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics : human language technologies, volume 1 (long and short papers)*. 2019, p. 4171-4186. url : https://aclanthology.org/N19-1423/?utm_campaign=The+Batch&utm_source=hs_email&utm_medium=email&_hsenc=p2ANqtz-_m9bbH_7ECE1h31Z3D61TYg52rKpifVNjL4fvJ85uqggrXsWDBTI (visité le 30/11/2025).
- [88] Tom Brown et al. "Language models are few-shot learners". In : *Advances in neural information processing systems* 33 (2020), p. 1877-1901. url : https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html?utm_source=transaction&utm_medium=email&utm_campaign=linkedin_newsletter (visité le 30/11/2025).
- [89] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In : *Journal of machine learning research* 21.140 (2020), p. 1-67. url : <http://www.jmlr.org/papers/v21/20-074.html> (visité le 30/11/2025).

- [90] Qingsong Wen et al. *Transformers in Time Series : A Survey*. 11 mai 2023. doi : 10.48550/arXiv.2202.07125. arXiv : 2202.07125[cs]. url : <http://arxiv.org/abs/2202.07125> (visité le 30/11/2025).
- [91] Haoyi Zhou et al. "Informer : Beyond efficient transformer for long sequence time-series forecasting". In : *Proceedings of the AAAI conference on artificial intelligence*. T. 35. 12. 2021, p. 11106-11115. url : <https://ojs.aaai.org/index.php/AAAI/article/view/17325> (visité le 30/11/2025).
- [92] Haixu Wu et al. "Autoformer : Decomposition transformers with auto-correlation for long-term series forecasting". In : *Advances in neural information processing systems* 34 (2021), p. 22419-22430. url : <https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bcd7c19a8ac0c2b-Abstract.html> (visité le 30/11/2025).
- [93] Ailing Zeng et al. "Are transformers effective for time series forecasting?" In : *Proceedings of the AAAI conference on artificial intelligence*. T. 37. 9. 2023, p. 11121-11128. url : <https://ojs.aaai.org/index.php/AAAI/article/view/26317> (visité le 30/11/2025).
- [94] Yuqi Nie et al. *A Time Series is Worth 64 Words : Long-term Forecasting with Transformers*. 5 mars 2023. doi : 10.48550/arXiv.2211.14730. arXiv : 2211.14730[cs]. url : <http://arxiv.org/abs/2211.14730> (visité le 30/11/2025).
- [95] Chen Sun et al. "Revisiting unreasonable effectiveness of data in deep learning era". In : *Proceedings of the IEEE international conference on computer vision*. 2017, p. 843-852. url : http://openaccess.thecvf.com/content_iccv_2017/html/Sun_Revisiting_Unreasonable_Effectiveness_ICCV_2017_paper.html (visité le 01/12/2025).
- [96] Ze Liu et al. "Swin transformer : Hierarchical vision transformer using shifted windows". In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, p. 10012-10022. url : https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.html (visité le 30/11/2025).
- [97] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In : *nature* 596.7873 (2021), p. 583-589. url : <https://www.nature.com/articles/s41586-021-03819-2> (visité le 30/11/2025).
- [98] Lili Chen et al. "Decision transformer : Reinforcement learning via sequence modeling". In : *Advances in neural information processing systems* 34 (2021), p. 15084-15097. url : <https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c3Abstract.html> (visité le 30/11/2025).
- [99] Chris Olah et al. "Zoom in : An introduction to circuits". In : *Distill* 5.3 (2020), e00024-001. url : <https://distill.pub/2020/circuits/zoom-in/?ref=cold-takes> (visité le 05/02/2026).

- [100] Einar Urdshals et Jasmina Urdshals. *Structure Development in List-Sorting Transformers*. 30 jan. 2025. doi : [10.48550/arXiv.2501.18666](https://doi.org/10.48550/arXiv.2501.18666). arXiv : [2501.18666\[cs\]](https://arxiv.org/abs/2501.18666). url : <http://arxiv.org/abs/2501.18666> (visité le 05/02/2026).
- [101] Baggins Matthew. *One Attention Head Is All You Need for Sorting Fixed-Length Lists by MatthewBaggins*. itch.io. url : <https://itch.io/@matthewbaggins/one-attention-head-is-all-you-need-for-sorting-fixed-length-lists> (visité le 05/02/2026).
- [102] Matthieu CORNU, Cyrille ENDERLI et Thomas RODET. "Learning to Sort in Continuous Spaces : Neural Architectures and Iterative Training Strategies". In : *Proceedings of the 2025 33rd European Signal Processing Conference (EUSIPCO) SiG-DML-P5.2* (2025), p. 1902. url : <https://eusipco2025.org/wp-content/uploads/pdfs/0001902.pdf>.