الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique Et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة الشهيد الشيخ العربي التبسي- تبسة

UNIVERSITE ECHAHID CHEIKH LARBI TEBESSI – TEBESSA

Faculté des Sciences et de la Technologie

Département de Genie Electrique

MEMOIRE

Présenté pour l'obtention du diplôme de Master Académique

En : Automatique

Spécialité : Automatique et Systèmes

Par: - TEBBA Sana - BERRAHAL Chahinez

Thème

KOLMOGOROV-ARNOLD NETWORK: CONCEPTS ET APPLICATIONS

Présenté et évalué, le 11 / 06 / 2025 , par le jury composé de :

Nom et prénom		Grade	Qualité
M.	LOUADJANI Abdelhak	Pr. Assisatant	Président
M.	KHEMAISSIA Seddik	Pr. Associe	Rapporteur
M.	SOUFI Youcef	Professeur	Examinateur

Scanné avec CamScanner


بسم الله الرحمن الرحيم

ملخص

شهد الذكاء الاصطناعي تطورًا متسارعًا، مكّنه من إنتاج نماذج عالية الأداء، لكنها غالبًا ما تفتقر إلى الشفافية وسهولة الفهم. في هذا السياق، تمثل شبكات كولموغوروف-أرنولد، المستوحاة من مبرهنة كولموغوروف-أرنولد، تصورًا بنيويًا مبتكرًا يعيد التفكير في بنية الشبكات العصبية، من خلال استبدال الأوزان الثابتة التقليدية بدوال رياضية أحادية المتغير قابلة للتعلّم، مثل سلاسل فوريه المبسطة.

تتيح هذه البنية إجراء تحويلات محلية غير خطية قابلة للتحليل الرياضي، مع قدرة عالية على نمذجة العلاقات المعقدة المستخلصة من البيانات. وعلى عكس الشبكات متعددة الطبقات التقليدية، تتكيف هذه الشبكات ديناميكيًا مع بنية البيانات وتنتج تمثيلات رمزية مفهومة.

في هذا العمل، استُخدمت هذه الشبكات لاكتشاف الانحرافات الزمنية في سلاسل البيانات، من خلال تكامل دوال فوريه المبسطة كأداة مزدوجة: للمعالجة المسبقة ولتحديث آلية التعلم الداخلي أثناء التدريب. وقد أظهر هذا النظام دقة قوية وقدرة على توليد تعبيرات رياضية تشرح الأنماط المكتشفة.

تسلّط هذه الدراسة الضوء على القدرات الرمزية والقابلة للتحليل لهذه الشبكات، ما يفتح آفاقًا واعدة لتصميم نماذج ذكاء اصطناعي تجمع بين الأداء، الشفافية، وقابلية التكيّف في البيئات الحساسة

الكلمات المفتاحية:

ذكاء اصطناعي، شبكات كولموغوروف-أرنولد، سلاسل فوريه، كشف الانحرافات الزمنية، نمذجة العلاقات، تحليل رمزي

Abstract:

Modern artificial intelligence offers powerful models, but often at the cost of interpretability. *Kolmogorov-Arnold Networks* (KANs), inspired by the Kolmogorov-Arnold representation theorem, introduce an innovative neural architecture where fixed weights are replaced by learnable univariate functions, such as *B-splines* or naive Fourier series.

This approach enables local, nonlinear, and interpretable transformations while maintaining strong approximation capabilities. Unlike traditional MLPs, KANs dynamically adapt to the structure of the data, offering greater mathematical transparency.

In this work, KANs are applied to anomaly detection in time series. Naive Fourier series are used in two ways: as a preprocessing method and as learnable parametric functions embedded in the network's edges. The results demonstrate high accuracy and provide symbolic interpretability of the detected patterns.

This study highlights the potential of KANs to build AI systems that are both powerful and transparent, suitable for critical and complex environments.

Keywords : Kolmogorov-Arnold Networks (KAN)- Apprentissage automatique interprétable - Décomposition fonctionnelle - B-splines - Transformée de Fourier naïve- Détection d'anomalies - Séries temporelles - Intelligence artificielle explicable (XAI) - Symbolic regression- Approximation universelle

Résumé :

L'intelligence artificielle moderne propose des modèles puissants, mais souvent peu interprétables. Les *Kolmogorov-Arnold Networks* (KANs), inspirés du théorème de Kolmogorov-Arnold, introduisent une architecture neuronale innovante où les poids fixes sont remplacés par des fonctions univariées apprenables, comme les *B-splines* ou les séries de Fourier naïves.

Cette approche permet des transformations locales, non linéaires et interprétables, tout en conservant une grande capacité d'approximation. Contrairement aux MLPs classiques, les KANs s'adaptent dynamiquement à la structure des données, tout en offrant une meilleure transparence mathématique.

Dans ce travail, les KANs sont appliqués à la détection d'anomalies dans des séries temporelles. Les séries de Fourier naïves y sont utilisées à double titre : d'une part comme méthode de prétraitement, et d'autre part comme fonctions paramétriques apprenables intégrées dans les arêtes du réseau. Les résultats obtenus montrent une précision élevée et une interprétabilité symbolique des comportements détectés.

Ce travail met en lumière le potentiel des KANs pour des systèmes d'IA à la fois performants, transparents et adaptés à des contextes critiques.

Mots clés :

Kolmogorov-Arnold Networks (KAN)-Apprentissage automatique interprétable-Décomposition fonctionnelle- B-splines- Transformée de Fourier naïve- Détection d'anomalies- Séries temporelles- Intelligence artificielle explicable (XAI)- Régression symbolique- Approximation universelle

Remerciement

*Nous souhaitons exprimer notre plus profonde gratitude à notre encadrant, « **Seddik KHEMAISSIA** » pour son soutien constant, ses conseils éclairés et sa patience tout au long de la réalisation de ce mémoire. Son expertise et son dévouement ont été essentiels pour mener à bien ce mémoire.*

Dédicace

À celle qui fut mon premier refuge, ma prière constante, et mon soutien indéfectible tout au long de ma vie…

À ma mère Salima, qui a tant enduré pour moi, toujours à mes côtés sans rien attendre en retour,

Chaque instant de fatigue que tu as supporté, chaque larme de peur que tes yeux ont versée, est une couronne sur ma tête et une médaille sur ma poitrine.

Tu es la sécurité quand le monde me fait peur, et la lumière quand les jours deviennent sombres.

Grâce à tes prières, ta tendresse et ta patience, je suis arrivé à ce moment…

Ce succès est pour toi, et pour toi seule, la plus grande que je connaisse.

À mon pilier premier, mon père Sayeh, je t'adresse tout mon amour et ma gratitude pour ton soutien, ta patience et ta confiance.

À mon oncle cher Fathi, toujours un soutien silencieux et affectueux, merci pour ta présence.

À mes chères sœurs :

Faiza, Amel, et Taqwa, vous êtes les fleurs de ma vie, recevez tout mon amour et mon estime.

À mes frères bien-aimés :

Mohsen et Nadhir, compagnons de route et de moments difficiles, vous avez été mon aide et mon appui, merci du fond du cœur.

Et à mes amies fidèles : Asma Boukrouz Shahinaz, et chaima qui ont partagé mes rêves et mon chemin, vous êtes mes meilleures compagnes, que vous restiez toujours près de moi.

À vous tous, j'offre ce travail modeste, en reconnaissance de votre bonté et de votre présence à mes côtés à chaque étape.

Tebba Sana

Dédicace

À celle qui a semé dans mon cœur les graines de l'amour et de la tendresse,

À celle qui a veillé des nuits entières pour mon bien,

À la source de douceur et à la plus belle des femmes... ma mère Chahrazad,

Je te dédie le fruit de mes efforts et de ma réussite ; c'est par tes prières que j'ai avancé, et par ta patience que j'ai grandi.

À mon pilier, mon compagnon, ma force... mon père Ammar,

Et à celui qui a toujours été un exemple de sagesse et de générosité... mon oncle Salah,

Vous avez, après Dieu, le mérite de tout ce que je suis aujourd'hui. Merci pour votre amour et votre soutien inconditionnels.

Aux souffles de mon âme, aux couleurs de ma joie...

À mes sœurs Sana, Radia et Wounassa,

Vous avez toujours été mon appui. Merci du fond du cœur.

À celui qui a toujours été un frère et un soutien... mon frère Sadeq,

Ta présence est une bénédiction dans ma vie. Que Dieu te garde pour moi.

Et à mes amis qui ont partagé ce chemin, avec ses joies et ses peines,

Sana, Asma, Samah, Chaïma et Rayane,

Merci pour votre amitié sincère. Vous êtes la plus belle des compagnies.

À tous ceux qui m'ont soutenu et accompagné,

Je dédie ce travail modeste en signe de gratitude et de reconnaissance.

Berrahal Chahinez

Table des Matières

Liste des figures ……..... **xii**

Liste Des Tableaux……..... **xiii**

Introduction Générale ……. **01**

Chapitre 1 : Les Réseaux de Neurones Artificiels

1.1 Introduction ……. **04**

1.2 Historique……. **04**

1.3 Le neurone biologique……. **05**

1.4 Comportement du neurone artificiel…… **06**

1.5 Fondements des Réseaux de Neurones…… **07**

1.5.1 Structure d'un réseau neuronal…… **07**

1.5.2 Couche d'entrée…… **07**

1.5.3 Couches cachées……... **07**

1.5.4 Couche de sortie…… **07**

1.6 Fonctions d'activation…….. **08**

1.7 Types de Réseaux de Neurones…… **09**

1.7.1 Perceptron monocouche…… **09**

1.7.2 Perceptron multicouches (MLP) ……. **10**

1.7.3 Réseaux convolutifs (CNN) …… **11**

1.8 Réseaux de Neurones Récurrents (RNN) …… **14**

1.8.1 Types de récurrence : LSTM…… **14**

1.9.1 Algorithme Feedforward…….. **15**

1.9.2 Rétropropagation du gradient…….. **15**

1.9.3 Descente du gradient…… **15**

1.9.4 Taux d'apprentissage…… **15**

1.9.5 Variantes de la descente…… **15**

1.9.6 Approximation fonctionnelle…… **17**

1.10 Conclusion…….…… **18**

Chapitre 2 : Kolmogorov-Arnold Networks: Concept et Fondements

2.1. Introduction…… **20**

2.2. Théorème de Kolmogorov-Arnold…… **21**

2.3. Décomposition des Fonctions Multivariées…… **22**

2.4. Différences Architecturales entre les KAN et les MLP Classiques…… **23**

2.5. Architecture des Réseaux Kolmogorov-Arnold (KANs) …… **24**

2.5.1. Structure des Réseaux KAN…… **24**

2.5.2. B-Splines dans les Réseaux KAN…… **26**

2.6. Apprentissage dans un KAN…… **28**

2.6.1. Initialisation du Réseau…… **28**

2.6.2. Propagation avant (Forward Pass) …… **29**

2.6.3. Définition de la fonction de perte…… **30**

2.6.4. Mise à jour des paramètres…… **30**

2.6.5. Régularisation et stabilisation…… **31**

2.6.6. Validation et test…… **32**

2.7. Reconstruction Symbolique Interprétable d'une Fonction Cible…… **33**

2.8. Domaines d'Application des Réseaux Kolmogorov-Arnold…… **34**

2.9. Conclusion…… **35**

Chapitre 3 : Détection d’Anomalies avec les KANs

3.1 Introduction…… **37**

3.2 Types d’anomalies…… **37**

3.2.1 Ponctuelles…… **38**

3.2.2 Contextuelles…… **38**

3.2.3 Collectives…… **38**

3.3 Méthodes de détection…… **38**

3.3.1 Supervisées…… **38**

3.3.2 Non supervisées…… **38**

3.3.3 Semi-supervisées…… **39**

3.3.4 Autres approches…… **39**

3.4 Application des KANs…… **39**

3.4.1 Domaines d'application…….…….. **39**
3.4.2 Défis…….…… **40**
3.5 Méthodologie…….……. **41**
3.5.1 Préparation des données…….……. **41**
3.5.2 Modèle KAN…….…… …. **42**
3.5.3 Entraînement …….……. **46**
3.5.4 Évaluation des performances…….……... **48**
3.6 Conclusion…….…….. **49**

Chapitre 4 — Implémentation des KAN

4.1 Introduction…….……. **51**
4.2 Environnement et Outils…….……. **51**
4.2.1 Plateformes…….……..…….. **51**
4.3 Prétraitement…….…… **51**
4.3.1 Injection d'anomalies…….…… **51**
4.3.2 Segmentation & Normalisation…….… **52**
4.3.3 Rééquilibrage & Partitionnement ……...……. **53**
4.4 Conception du modèle…….……..…… **53**
4.4.1 Couche Naïve Fourier…….…… **53**
4.4.2 Architecture KAN…….……..…… **54**
4.5 Entraînement…….……. **55**
4.5.1 Hyperparamètres…….……...…… **55**
4.5.2 Processus d'entraînement…….…..…… **55**
4.5.3 Évaluation…….……...… **55**
4.6 Visualisation des Résultats…….……...…… **57**
4.6.1 Série annotée…….……..…… **57**
4.6.2 Alignement des prédictions…….…...…… **57**
4.6.3 Analyse des résultats …….……..… **58**
4.7 Impact des Hyperparamètres…….……..…… **59**
4.7.1 Taille de la fenêtre……...…… **59**
4.7.2 Grid de Fourier……...…… **61**
4.7.3 Stride…….…….…….. **62**

4.7.4 Synthèse…… **64**
4.8 Validation sur données ECG5000…… **65**
4.8.1 Résultats…… **65**
4.8.2 Analyse…… **66**
4.9 Perspectives et Applications…… **67**
4.10 Conclusion…… **67**

Liste des Figures

Figure 1.1 — Les Réseaux de Neurones : Structure et Apprentissage…… **04**
Figure 1.2 — Historique des réseaux de neurones et du Deep Learning…… **05**
Figure 1.3 — Neurone biologique…… **06**
Figure 1.4 — Architecture du neurone artificiel…… **06**
Figure 1.5 — Réseau de neurones artificiels…… **08**
Figure 1.6 — Fonctions d'activation…… **08**
Figure 1.7 — Réseau Perceptron…… **10**
Figure 1.8 — Architecture du Perceptron multicouche (MLP) …… **11**
Figure 1.9 — Réseau de neurones convolutif (CNN) …… **11**
Figure 1.10 — Image convertie en matrice de pixels…… **12**
Figure 1.11 — Carte des caractéristiques…… **12**
Figure 1.12 — Couches de regroupement (Pooling) …… **13**
Figure 1.13 — Couches entièrement connectées…… **13**
Figure 1.14 — Architecture CNN complète…… **13**
Figure 1.15 — RNN simple avec couche récurrente…… **15**
Figure 1.16 — Cellule mémoire d'un LSTM…… **15**
Figure 1.17 — Apprentissage par rétropropagation…… **16**
Figure 2.1 – Comparaison entre une connexion MLP et une connexion KAN **21**
Figure 2.2 – Illustration générale du principe d'un réseau KAN…… **21**
Figure 2.3 – Illustration du théorème de Kolmogorov-Arnold…… **22**
Figure 2.4 – Décomposition d'une fonction de degré 3 (cubiques) …… **23**
Figure 2.5 – Flux de données dans une structure du réseau KAN…… **25**
Figure 2.6 – Représentation d'une base de B-splines locales et contrôle local **25**
Figure 2.7 – Génération d'une trajectoire avec les B-splines…… **26**

Figure 2.8 – Construction des B-splines de degré k, notée $B_{i,k}(x)$ ……..…… **27**
Figure 2.9 – Visualisation d'une fonction apprise par une connexion spline ……..…… **28**
Figure 2.10 – Chaque spline ajuste localement la forme de la fonction……..… **29**
Figure 2.11 – Optimisation dans un KAN ……..…. **31**
Figure 2.12 – Architecture multi-couches d'un réseau KAN complet……..… **33**
Figure 2.13 – Étapes de fonctionnement d'un réseau Kolmogorov-Arnold……..… **33**
Figure 3.1 – Illustration des trois types d'anomalies dans les séries temporelles **38**
Figure 3.2 – Exemple de fenêtrage temporel sur un signal……...…… **43**
Figure 3.3 – Fenêtre glissante à travers les données……...…….. **43**
Figure 3.4 : Illustration de la Transformation de Fourier d'une série temporelle **44**
Figure 3.6 – Architecture du modèle KAN……..…. **46**
Figure 4.1 – Architecture générale du modèle KAN pour la détection d'anomalies…… **57**
Figure 4.2 – Détection d'anomalies et courbes ROC-PR……..… **59**
Figure 4.3 – Impact d'une fenêtre temporelle courte (window_size = 10) **61**
Figure 4.4 – Impact d'une fenêtre temporelle longue (window_size = 40) **61**
Figure 4.5 – Grid de Fourier faible (10 composantes) …… **62**
Figure 4.6 – Grid de Fourier élevé (200 composantes) ……..…… **63**
Figure 4.7 – Stride court (step_size = 4) ……..… **64**
Figure 4.8 – Stride long (step_size = 10) ……..… **64**
Figure 4.9 – Stride intermédiaire optimal (step_size = 6) ……...…… **64**
Figure 4.10 – Résultat optimal avec configuration window=20, grid=80, stride=6 **66**
Figure 4.11 – Annotation des anomalies sur le jeu de données ECG5000……... **67**
Figure 4.12 – Comparaison des détections : KAN vs modèles classiques……...

Liste Des Tableaux

Tableau 2.1 – Types de splines et leurs continuités……..… **26**
Tableau 3.1 – Interprétation de la fonction de perte Focal Loss……. **47**
Tableau 3.2 – Métriques d'évaluation utilisées…… **49**
Tableau 4.1 – Répartition des fenêtres entre ensembles d'apprentissage……..…
Tableau 4.2 – Hyperparamètres du modèle KAN et justifications……...…. **56**
Tableau 4.3 – Performances validation vs test (données synthétiques) ……..…. **59**
Tableau 4.4 – Effet de la taille de la fenêtre temporelle……..…… **61**

Tableau 4.5 – Impact du grid de Fourier sur les performances…… **62**
Tableau 4.6 – Influence du pas de glissement (stride) …… **63**
Tableau 4.7 – Résultats du modèle KAN sur le jeu de données ECG5000,……. **66**
Tableau 4.8 – Comparatif des performances du KAN sur données synthétiques …….... **69**

INTRODUCTION GENERALE

Introduction Générale

L'intelligence artificielle (IA) joue aujourd'hui un rôle central dans la transformation numérique de nombreux domaines tels que la santé, la finance, l'industrie et la cybersécurité. Les avancées en apprentissage profond (deep learning) ont permis de développer des modèles capables d'analyser de grandes quantités de données et d'en extraire automatiquement des caractéristiques complexes.

Cependant, ces modèles restent souvent opaques : ils fonctionnent comme des « boîtes noires », rendant difficile la compréhension des décisions qu'ils produisent. Cette opacité pose un réel problème dans les domaines critiques, où la transparence, la fiabilité et l'interprétabilité sont indispensables.

Dans ce contexte, les Kolmogorov–Arnold Networks (KANs) apparaissent comme une alternative intéressante. Ces réseaux, inspirés du théorème de Kolmogorov-Arnold, utilisent des fonctions univariées apprenables à la place des poids classiques dans les connexions neuronales. Cette architecture permet de modéliser des relations complexes tout en offrant une meilleure lisibilité des transformations internes.

L'objectif principal de ce travail est d'explorer l'efficacité des Kolmogorov–Arnold Networks (KANs) comme alternative interprétable aux modèles classiques de type perceptron multicouche (MLP) dans la tâche de détection d'anomalies sur des séries temporelles.

Plus précisément, cette recherche vise à :

Remplacer les MLP par des KANs dans un cadre de détection d'anomalies, tout en comparant les performances et le niveau d'interprétabilité des deux approches.

Intégrer une transformée de Fourier naïve comme base fonctionnelle dans les connexions apprenables des KANs, afin de capter efficacement les composantes périodiques des séries temporelles.

Évaluer la capacité des KANs-Fourier à modéliser des séries complexes, en mettant l'accent sur la détection des anomalies subtiles et la lisibilité des transformations internes.

Analyser la robustesse du modèle proposé sur des séries réelles et synthétiques, en comparant les résultats à ceux obtenus par des architectures conventionnelles.

L’ambition est ainsi de démontrer qu’il est possible de concilier performance, transparence et interprétabilité dans un cadre d’intelligence artificielle appliqué aux séries temporelles.

Afin de structurer notre étude, le mémoire est organisé en quatre chapitres principaux :

Chapitre 1 : Les réseaux de neurones artificiels

Ce chapitre introduit les fondements du deep learning, en passant en revue les architectures classiques telles que le perceptron multicouche (MLP), les réseaux convolutifs (CNN), et les réseaux récurrents (RNN). Il constitue une base essentielle pour comprendre les limites des modèles standards en termes d’interprétabilité.

Chapitre 2 : Les Kolmogorov–Arnold Networks – Concepts et Fondements

Ce chapitre présente en détail la structure, la logique et les avantages théoriques des KANs. Il explique le théorème de Kolmogorov, la construction des fonctions sur les connexions, et le remplacement des poids par des bases fonctionnelles telles que les B-splines ou, dans notre cas, les séries de Fourier naïves.

Chapitre 3 : Détection d’anomalies avec les KANs

Cette partie aborde les différents types d’anomalies (ponctuelles, contextuelles, collectives), les méthodes de détection existantes, et introduit notre approche à base de KANs pour traiter ce problème. Elle décrit la méthodologie adoptée et les défis spécifiques liés à l’apprentissage sur des séries temporelles.

Chapitre 4 : Implémentation, évaluation et analyse des résultats

Ce dernier chapitre est dédié à l’implémentation du modèle. Il présente les choix d’architecture (KAN avec base de Fourier), les jeux de données utilisés (notamment ECG5000), les paramètres d’apprentissage, les visualisations, les courbes d’évaluation, ainsi que les perspectives futures d’amélioration.

À travers cette organisation, nous cherchons à montrer que les KANs, enrichis par une base fonctionnelle adaptée comme la Fourier naïve, offrent un compromis prometteur entre performance, interprétabilité, et simplicité mathématique, ouvrant la voie à des systèmes d’IA de nouvelle génération.

CHAPITRE 01

LES RESEAUX NEURONAUX ARTIFICIELS

1.1 Introduction

Les réseaux neuronaux artificiels (RNA) sont un modèle clé du Machine Learning, inspiré du cerveau humain. Ils sont constitués de neurones artificiels organisés en plusieurs couches : une couche d'entrée, des couches cachées et une couche de sortie. Leur fonctionnement repose sur la transmission et l'ajustement de poids entre les neurones pour optimiser l'apprentissage à partir des données.

Grâce à leur architecture, les RNA sont capables d'identifier des modèles complexes et de prendre des décisions sophistiquées. Cette capacité en fait un outil puissant pour de nombreuses applications, comme la reconnaissance d'images, la traduction automatique et l'analyse prédictive.

Les réseaux neuronaux profonds (Deep Neural Networks) amplifient ces capacités en multipliant les couches cachées, permettant ainsi un apprentissage plus avancé, au cœur du Deep Learning.

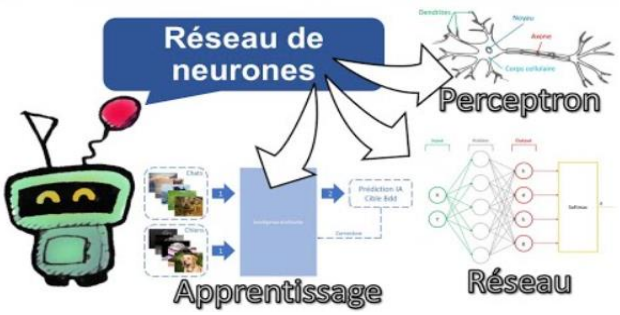


Figure (1.1): Réseaux de Neurones : Structure et Apprentissage

Les réseaux neuronaux constituent un modèle de machine Learning capable de prendre des décisions de manière similaire au cerveau humain. Ils fonctionnent en reproduisant les interactions des neurones biologiques, ce qui leur permet d'identifier des phénomènes, d'évaluer des options et de tirer des conclusions.

1.2 Historique

L'inspiration des réseaux neuronaux artificiels remonte à la fin du XIXe siècle, lorsque W. James introduit le concept de mémoire associative. En 1943, McCulloch et Pitts proposent le premier modèle formel de neurone artificiel, démontrant que des réseaux simples peuvent réaliser des opérations logiques . Peu après, en 1949, D. Hebb formule la loi de Hebb, une règle d'apprentissage influente basée sur le renforcement des connexions synaptiques [1][2].

Dans les années 1950, F. Rosenblatt développe le Perceptron, premier modèle d'apprentissage automatique. Ce modèle, bien que prometteur, est limité dans sa capacité à traiter des problèmes

non linéaires, comme l'ont démontré Minsky et Papert en 1969, ce qui entraîne un ralentissement des recherches sur les réseaux neuronaux.

Le domaine connaît un renouveau en 1982 avec J.J. Hopfield, qui propose un modèle neuronal pour la résolution de problèmes d'optimisation. En 1983, la Machine de Boltzmann est introduite, bien qu'elle souffre de temps de calcul importants. Une avancée majeure survient en 1985 avec la rétropropagation du gradient, un algorithme qui permet aux Perceptrons multi-couches d'apprendre des fonctions non linéaires, relançant l'intérêt pour les réseaux neuronaux [3].

Depuis, l'essor du Deep Learning a été favorisé par l'augmentation de la puissance de calcul et la disponibilité de grandes quantités de données. Des architectures avancées, comme les réseaux de neurones convolutifs (CNN) et les réseaux neuronaux récurrents (RNN), ont révolutionné des domaines tels que la reconnaissance d'image, la traduction automatique et la conduite autonome. De nos jours, l'utilisation des réseaux de neurones dans divers domaines ne cesse de croître. Les applications en sont multiples et variées[4].

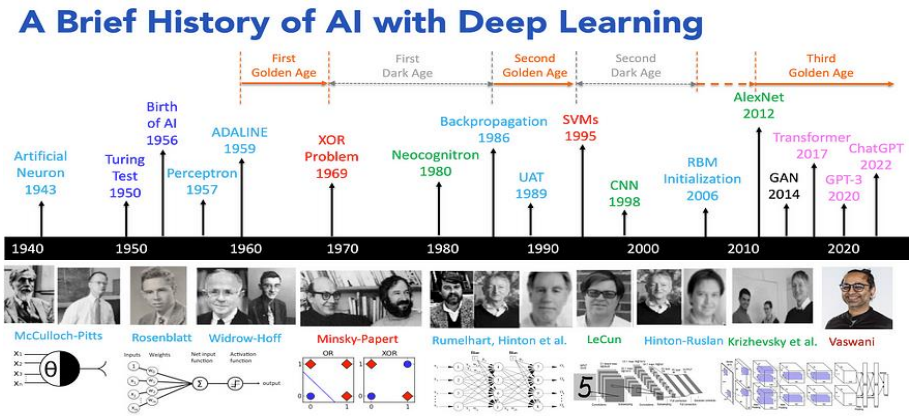


Figure (1.2): Historique des réseaux de neurones et Deep Learning

1.3 Le neurone biologique

Le neurone biologique (voir figure 1.3) est une cellule nerveuse qui constitue l'unité fonctionnelle fondamentale du système nerveux de tous les animaux. Les neurones existent pour communiquer les uns avec les autres et transmettre des impulsions électrochimiques à travers les synapses, d'une cellule à l'autre, à condition que l'impulsion soit suffisamment puissante pour activer la libération de produits chimiques à travers une fente synaptique. La force de l'impulsion doit dépasser un seuil minimal, sinon les produits chimiques ne seront pas libérés.

La figure présente les principales parties de la cellule nerveuse : le soma, les dendrites, les axones et les synapses.

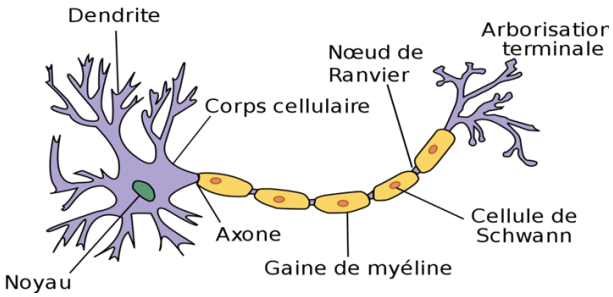


Figure (1.3): Neurone biologique**.**

1.4 Comportement de neurone artificiel

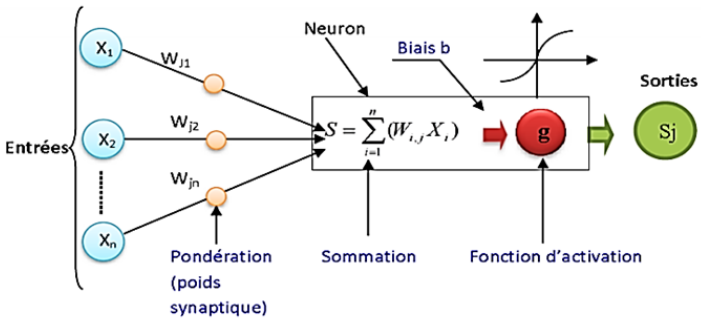


Figure (1.4): Architecture du neurone artificiel.

La **figure (1.3)** illustre l'architecture d'un neurone artificiel, inspirée du modèle biologique. Ce neurone reçoit des signaux d'entrée x_i (où $\boldsymbol{i}$=1,2,...,$\boldsymbol{n_i}$ =1,2,..., n) via des connexions synaptiques pondérées par des poidsW_{ij} (*Weight* en anglais). Ces poids ajustent l'importance de chaque entrée et déterminent l'influence relative des signaux.

Le fonctionnement du neurone se décompose en deux phases :

1. **Calcul du Potentiel Synaptique (Combinaison Linéaire)**

Le neurone *j* somme les entrées pondérées et ajoute un terme de biais $\boldsymbol{b_j}$ (aussi appelé *biais*). Ce biais agit comme un poids supplémentaire associé à une entrée fictive X_0=1 ou 0.

Potentiel d'activation

$$f_j = \sum_{i=1}^{n} (\text{Wij Xi}) + bj \qquad (1.1)$$

2. **Application de la Fonction d'Activation**

Le potentiel *uj* est transformé par une **fonction d'activation** *f* pour produire la sortie Y_j du neurone :

$$y_j = f(u_j) = f(\sum_i W_{ij} X_i + b_j) \quad (1.2)$$

1.5 Fondements des Réseaux de Neurones Artificiels

1.5.1 Structure d'un Réseau Neuronal

Les réseaux de neurones artificiels (RNA) sont constitués de plusieurs couches interconnectées, chacune jouant un rôle spécifique dans le traitement de l'information. Ces couches permettent d'extraire progressivement des caractéristiques pertinentes à partir des données d'entrée et de les transformer en sorties exploitables. On distingue généralement trois types de couches principales :

1.5.2 Couche d'entrée (Input Layer)

La couche d'entrée est la première étape du réseau neuronal. Elle reçoit les données brutes du modèle sous forme de vecteurs numériques, qu'il s'agisse d'images, de texte ou de signaux. Chaque neurone de cette couche est dédié à une caractéristique spécifique des données d'entrée et transmet ces informations aux couches suivantes sans modification.

1.5.3 Couches cachées (Hidden Layers)

Les couches cachées constituent le cœur du réseau. Elles appliquent des transformations complexes sur les données en combinant les poids attribués aux connexions entre neurones et en appliquant des fonctions d'activation non linéaires. Ces couches permettent d'extraire des représentations de plus en plus abstraites des données d'entrée.

- Dans un réseau peu profond, une ou deux couches cachées suffisent pour modéliser des relations simples.
- Dans un réseau profond (Deep Learning), plusieurs couches cachées permettent d'apprendre des structures complexes, comme les motifs dans des images ou les relations sémantiques dans le texte.

1.5.4 Couche de sortie (Output Layer)

La couche de sortie produit la réponse finale du réseau neuronal en fonction de la tâche à accomplir.

- Pour un problème de classification, elle contient un ou plusieurs neurones, chacun représentant une classe, avec une fonction d'activation comme Softmax pour générer des probabilités.
- Pour une tâche de régression, elle contient un neurone unique fournissant une valeur continue.

Chaque couche joue un rôle essentiel dans la transformation progressive des données, rendant le réseau capable de prendre des décisions plus précises et adaptées à la tâche demandée. Plus le réseau est complexe et bien entraîné, plus il peut résoudre des problèmes sophistiqués tels que la reconnaissance faciale, la traduction automatique ou encore la conduite autonome.

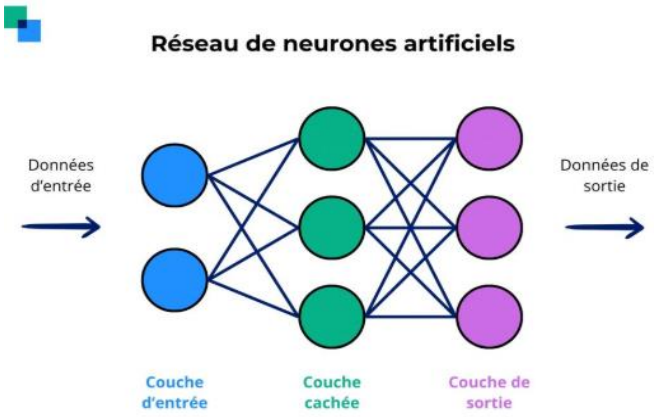


Figure (1.5) : Réseau de neurones artificiels

1.6 Fonctions d'Activation

Plusieurs fonctions d'activation, largement employées dans les tâches de vision par ordinateur, seront explorées dans cette section, accompagnées d'une analyse de leurs caractéristiques, avantages et limites.

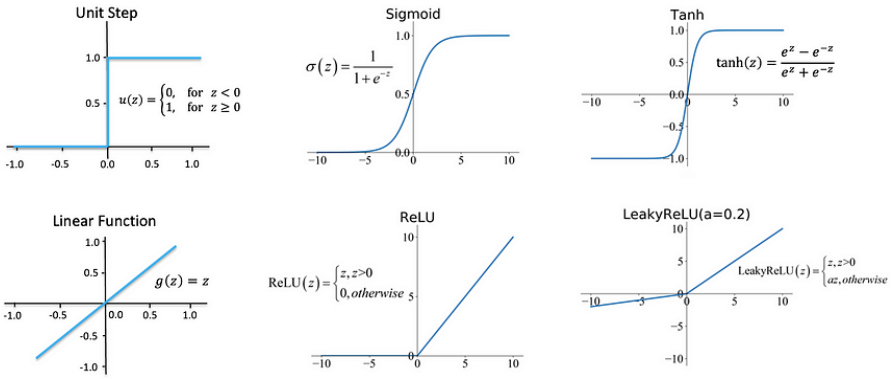


Figure (1.6) : Fonctions d'activation

1.6.1 Sigmoïde

La fonction d'activation sigmoïde, également connue sous le nom de fonction logistique, est un choix classique dans les réseaux de neurones idéale pour la classification binaire, mais sujette au problème du gradient qui disparaît.

1.6.2 ReLU (Rectified Linear Unit)

La fonction Rectified Linear Unit (ReLU) est la fonction d'activation la plus couramment utilisée en Deep Learning.

Cette fonction permet d'appliquer un filtre en sortie de couche. Elle laisse passer les valeurs positives dans les couches suivantes et bloque les valeurs négatives. Ce filtre permet alors au modèle de se concentrer uniquement sur certaines caractéristiques des données, les autres étant éliminées. Très utilisée dans l'apprentissage profond.

1.6.3 Softmax

La fonction Softmax est la fonction d'activation utilisée en dernière couche d'un réseau de neurones construit pour effectuer une tâche de classification multi-classes. Pour chaque sortie, Softmax donne un résultat entre 0 et 1. De plus, si l'on additionne ces sorties entre elles, le résultat donne 1. Souvent employée en classification multi-classes.

1.6.4 tanh

La fonction tanh permet d'appliquer une normalisation aux valeurs d'entrée. Elle peut également être utilisée au lieu de la fonction Sigmoïde dans la dernière couche d'un modèle de classification binaire. Elle donne un résultat entre -1 et 1.

1.6.5 Fonction Leaky ReLU

Leaky ReLU introduit une petite pente pour les entrées négatives, permettant à un petit gradient de s'écouler même pour les valeurs négatives, empêchant ainsi les neurones de s'atténuer.

1.7 Types de Réseaux de Neurones

Les réseaux de neurones artificiels (RNA) se déclinent en plusieurs types selon leur architecture et leur mode de fonctionnement

1.7.1 Perceptron à une seule couche (monocouche)

La forme la plus simple d'un réseau de neurone est le perceptron. Ce réseau est considéré parmi les premiers réseaux de neurones.

Le perceptron se compose d'un neurone artificiel à poids ajustable et d'un seuil. Il n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Les entrées et la sortie sont booléennes.

Dans ce type de réseaux, seulement les poids entre les unités d'entrées et la sortie peuvent être modifiés, tandis que la sortie de neurone ne peut prendre que deux états : -1 et 1 ou 0 et 1.

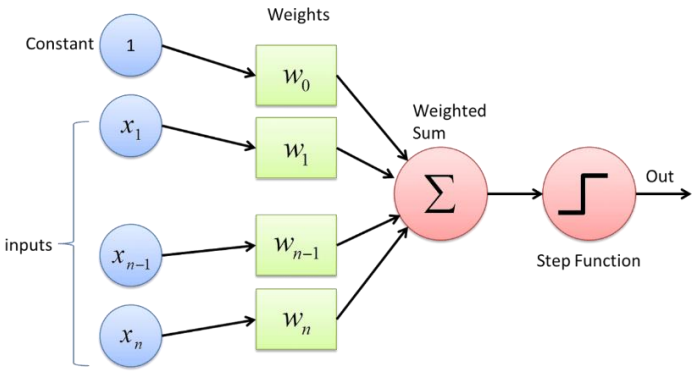


Figure (1.7): Réseau Perceptron

1.7.2 Perceptron Multi Couches

Le perceptron Multi Couches PMC ou MLP (Multi Layer Perceptron) en anglais est une extension du perceptron monocouche, avec une ou plusieurs couches cachées entre l'entrée et la sortie.

L'idée principale est de grouper des neurones dans une couche. En place ensuite bout à bout plusieurs couches et on connecte complètement les neurones de deux couches adjacentes Les entrées des neurones de la deuxième couche sont donc en fait les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée. Ils calculent alors leurs sorties qui sont Transmises aux neurones de la deuxième couche, etc. Les sorties des neurones de la dernière couche forment la sortie du réseau. Les réseaux multicouches sont beaucoup plus puissants que les réseaux simples à une seule couche. En utilisant deux couches (une couche cachée et une couche de sortie) à condition d'employer une fonction d'activation sigmoïde sur la couche cachée, on peut entrainer un réseau à produire une approximation de la plupart des fonctions, avec une précision arbitraire (cela peut cependant requérir un grand nombre de neurones sur la couche cachée). Sauf dans de rares cas, les réseaux de neurones artificiels exploitent deux ou trois couches.

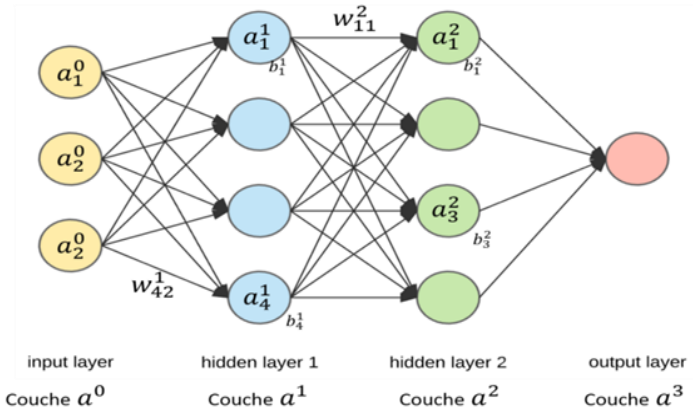


Figure (1.8) : Architecture Perceptron Multi couches

1.7.3 Réseaux de Neurones Convolutifs (CNN)

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils Comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a deux dimensions pour une image aux niveaux de gris.

La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

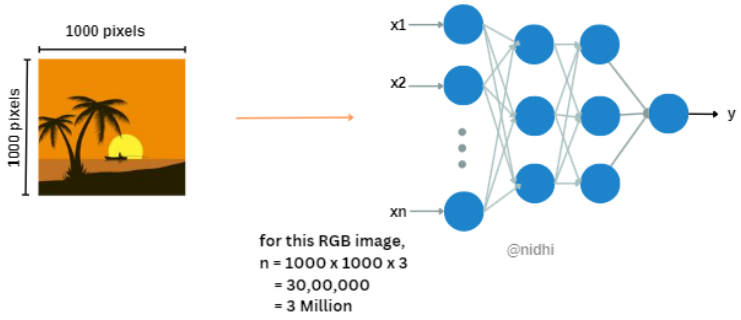


Figure (1.9) : Réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont des architectures essentielles du Deep Learning, particulièrement adaptées à l'analyse d'images [5]. Leur principe repose sur l'extraction automatique de caractéristiques pertinentes grâce à trois types de couches fondamentales :

1. Couches convolutives

Les couches **convolutives** sont au cœur des CNN. Elles appliquent des **filtres (kernels)** à l'image d'entrée afin d'extraire des caractéristiques telles que les bords, textures et motifs complexes.

- Fenêtre glissante : Un filtre se déplace sur l'image, multipliant ses valeurs par les pixels correspondants.
- Somme pondérée : Après multiplication, les valeurs sont additionnées pour produire un nouveau pixel dans la carte des caractéristiques.
- P**aramètre de stride** : Définit le déplacement du filtre après chaque opération.

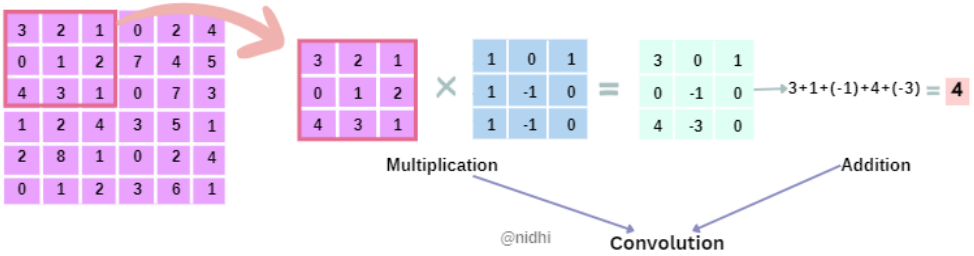


Figure (1.10) : Image convertie en une matrice de valeurs de pixels.

Une fonction d'activation, généralement ReLU (Rectified Linear Unit), est appliquée après la convolution pour introduire de la non-linéarité et améliorer l'apprentissage des modèles complexes.

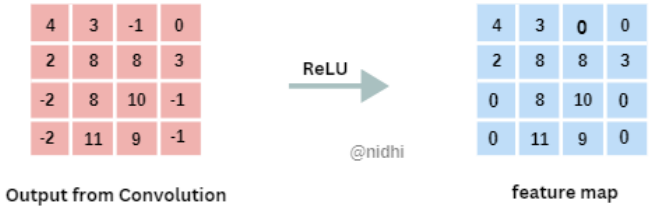


Figure (1.11) : la carte des caractéristiques

2. Couches de regroupement (Pooling)

Les couches de regroupement (pooling) réduisent la dimensionnalité des cartes de caractéristiques tout en conservant les informations essentielles.

L'opération la plus courante est le Max Pooling, qui sélectionne la valeur maximale dans une fenêtre donnée (ex. 2×2) pour créer une carte plus compacte.



Figure (1.12) : couches de regroupement (pooling)

3. Couches entièrement connectées

Les couches entièrement connectées apparaissent en fin d'architecture pour finaliser la classification :

- Aplatissement : Transformation des cartes de caractéristiques en un vecteur unidimensionnel.
- Couches cachées : Effectuent des combinaisons non linéaires pour affiner la prise de décision.
- Couche de sortie : Produit la classification finale, avec un neurone par classe dans les tâches de classification.

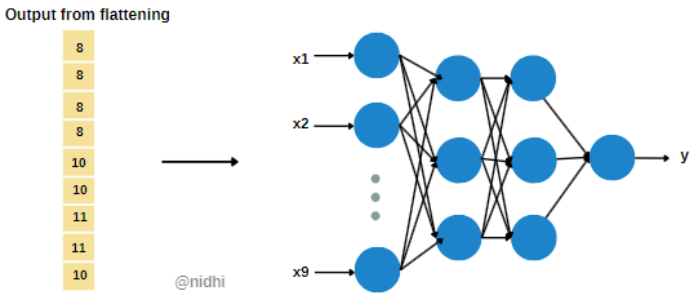


Figure (1.13) : Les couches entièrement connectées

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories

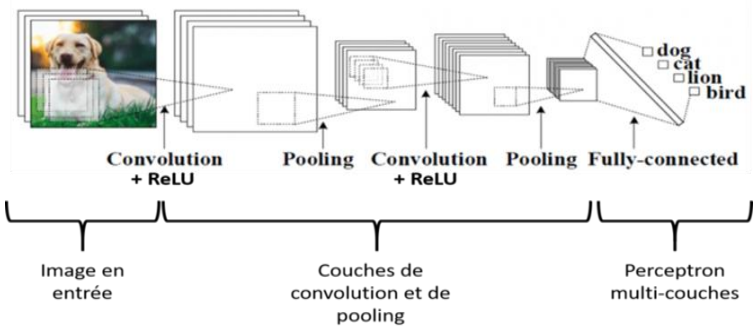


Figure (1.14) : Réseau de neurones a convolution (CNN)

1.8 Réseaux de Neurones Récurrents (RNN)

Les Réseaux de Neurones Récurrents (RNN) se distinguent des réseaux neuronaux traditionnels par leur capacité à traiter les informations séquentielles.Alors que les réseaux classiques supposent l'indépendance des entrées et des sorties, les RNN sont conçus pour les tâches où l'ordre des données est crucial.Par exemple, pour prédire le mot suivant dans une phrase, il est indispensable de considérer le contexte fourni par les mots précédents.

La caractéristique "récurrente" des RNN provient de leur fonctionnement : ils appliquent la même opération à chaque élément d'une séquence, en intégrant les résultats des calculs précédents.On peut concevoir les RNN comme des réseaux dotés d'une "mémoire" qui enregistre les informations traitées jusqu'à un certain point.En théorie, cette mémoire permet aux RNN de prendre en compte des dépendances dans des séquences de longueur arbitraire.Cependant, en pratique, leur capacité à capturer les dépendances à long terme est limitée, et on les restreint généralement à considérer un nombre limité d'étapes antérieures.

Les RNN sont utilisés dans de nombreuses applications, notamment :

- La modélisation du langage et la génération de texte,
- La traduction automatique,
- La reconnaissance vocale,
- La description d'images.

En raison de la complexité des connexions dans les RNN, on utilise souvent des représentations simplifiées. Dans ces schémas, une flèche unique peut représenter une matrice de poids W complète. Par exemple, la figure 1.14 illustre un RNN simple avec une couche récurrente suivie d'une couche dense classique.

1.8.1 Types de Récurrence et Architectures Connexes

Les RNN sont conçus pour traiter les données séquentielles en conservant une forme de "mémoire" des entrées précédentes. Cette capacité est mise en œuvre à travers différentes architectures, dont les plus importantes sont :

- **RNN Classiques :** Ce sont les formes de base des RNN, où la sortie à un pas de temps donné est influencée par les entrées et l'état caché du pas de temps précédent. Ils sont simples mais souffrent de difficultés à capturer les dépendances à long terme.

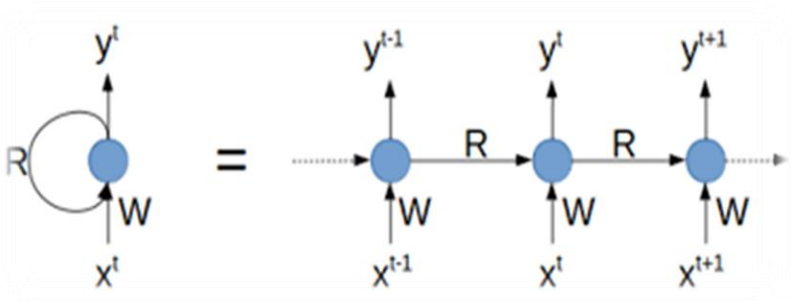


Figure (1 .15) : Représentation simplifiée d'un RNN simple, avec une couche récurrente et une couche dense.

- **Long Short-Term Memory (LSTM) :** Les réseaux LSTM constituent une évolution des RNN classiques. Proposés par Jürgen Schmidhuber et son équipe au début des années 1990, ils ont été spécialement conçus pour surmonter le problème du "vanishing gradient" qui limite la capacité des RNN classiques à traiter les dépendances à long terme [6]. Les LSTM introduisent le concept de "cellule mémoire", une unité qui permet au réseau de stocker et de récupérer des informations sur de longues périodes. La figure 1.16 illustre le schéma d'une cellule mémoire LSTM.

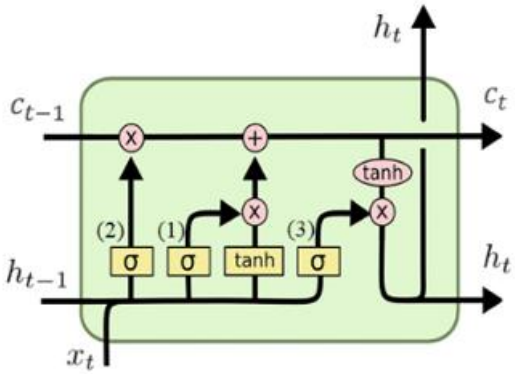


Figure (1 .16) : schéma d'une cellule mémoire d'un LSTM.

Cette réorganisation met en évidence la distinction entre les RNN en général et les types spécifiques (comme les LSTM), tout en conservant les informations essentielles du texte original.

1.9 Entraînement d'un réseau de neurones

L'entraînement d'un réseau de neurones est le processus d'ajustement des poids et des paramètres internes afin qu'il puisse prédire avec précision les résultats en fonction des données d'entrée. Cet entraînement repose sur la présentation répétée des données au réseau et sur la mise à jour des poids pour minimiser l'erreur des sorties.

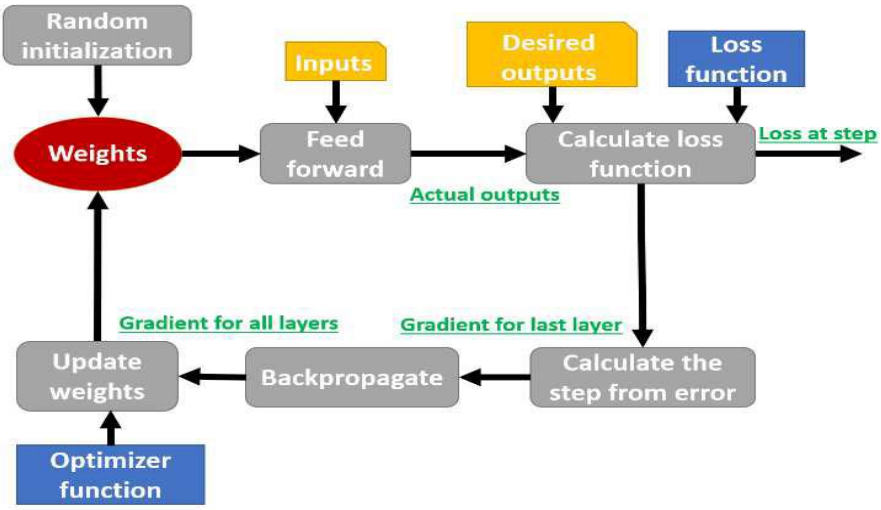


Figure (1 .17) : Apprentissage des réseaux de neurone par l'algorithme de rétropropagation.

1.9.1 Feedforward (entrainement direct)

L'algorithme Feedforward constitue la première étape de l'entraînement. Il consiste à propager les données d'entrée à travers les couches du réseau jusqu'à la couche de sortie.

Chaque neurone de la première couche cachée reçoit une combinaison linéaire des valeurs d'entrée pondérées par les poids des connexions. Cette combinaison est ensuite transformée par une fonction d'activation. Ce processus se répète pour chaque couche cachée jusqu'à la couche de sortie.

1.9.2 Rétropropagation du gradient (Backpropagation)

La rétropropagation du gradient est une méthode d'optimisation qui ajuste les poids du réseau en fonction de l'erreur calculée par la fonction de perte.

Fonction de perte

Une des fonctions de perte les plus couramment utilisées est l'**Erreur Quadratique Moyenne (MSE)**, définie comme suit :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{i-}\hat{y}_i)^2 \tag{1.3}$$

Avec :

- y_i : la valeur de sortie correcte,
- $\hat{y}_i$: la valeur de sortie prédite,
- N: le nombre d'échantillons

1.9.3 Descente du gradient

La descente du gradient est l'algorithme employé pour minimiser la fonction de perte. Ses étapes sont les suivantes :

1. Extraire un lot de données d'entraînement (X_i) et leurs sorties correspondantes (Y_i).
2. Appliquer l'algorithme feedforward pour obtenir les prédictions (Y'i).
3. Calculer la perte entre les sorties attendues (Yi) et les sorties prédites (Y'i).
4. Calculer le gradient (la dérivée) de la fonction de perte par rapport aux poids du réseau.
5. Mettre à jour les poids selon la formule :

$$W = W - \eta(\frac{\partial Loss}{\partial W}) \quad (1.4)$$

6. avec η (taux d'apprentissage) entre 0 et 1.
7. Répéter ces étapes jusqu'à minimisation de l'erreur.

1.9.4 Importance du taux d'apprentissage

Le taux d'apprentissage joue un rôle crucial :

- Un taux trop faible prolonge l'entraînement.
- Un taux trop élevé peut empêcher la convergence ou conduire à un minimum local au lieu du minimum global.

1.9.5 Variantes de la descente du gradient

Il existe plusieurs variantes de la descente du gradient, adaptées aux différents besoins en termes de stabilité, vitesse et performance d'apprentissage :

- **Descente du gradient par lot (Batch Gradient Descent)** : Traite l'ensemble des données d'entraînement à chaque itération, offrant une stabilité élevée mais une convergence plus lente.
- **Descente du gradient stochastique (SGD - Stochastic Gradient Descent)** : Met à jour les poids après chaque échantillon, ce qui accélère l'apprentissage mais ajoute de la variance aux mises à jour.
- **Mini-batch SGD** : Compromis entre les deux précédents, il met à jour les poids après un petit sous-ensemble de données, combinant rapidité et stabilité.

Des optimisateurs plus avancés comme Adagrad, RMSProp et Adam améliorent la descente du gradient en adaptant dynamiquement le taux d'apprentissage.

En général, Adam et AdamW sont privilégiés pour leur robustesse et leur rapidité de convergence. Toutefois, SGD avec Momentum reste une option prisée pour l'entraînement des grands modèles nécessitant une meilleure généralisation.

1.9.6 Approximation Fonctionnelle et Généralisation des Réseaux MLP

Les réseaux multicouches (MLP) possèdent une forte capacité d'approximation fonctionnelle : ils peuvent théoriquement approximer une large gamme de fonctions avec une précision potentiellement arbitraire, à condition d'utiliser un nombre suffisant de neurones dans leurs couches cachées (ce principe est soutenu par le théorème d'approximation universelle). Toutefois, cette puissance peut se retourner contre eux. Une complexité excessive du réseau conduit souvent au sur-apprentissage, un phénomène où le modèle excelle sur les données d'entraînement mais échoue à généraliser efficacement à de nouvelles données. Pour atténuer le sur-apprentissage et améliorer la généralisation, des techniques de régularisation sont essentielles. Les méthodes courantes incluent Dropout (qui désactive aléatoirement des neurones pendant l'entraînement) et la régularisation L2 (qui pénalise les poids de grande magnitude). En raison de leur flexibilité, les MLP sont largement utilisés dans divers domaines, notamment la reconnaissance de formes (images, sons, etc.), la prédiction de séries temporelles et la modélisation de relations complexes au sein de données variées.

1.9.7 Applications des Réseaux de Neurones

Les réseaux de neurones artificiels sont largement utilisés dans divers domaines, révolutionnant de nombreuses industries grâce à leur capacité d'apprentissage avancée. Voici quelques applications majeures :

- **Vision par ordinateur** : Utilisés pour la reconnaissance faciale, la détection d'objets, l'analyse d'images médicales (ex. détection de tumeurs) et les véhicules autonomes.
- **Traitement du langage naturel (NLP)** : Améliorent la traduction automatique (Google Translate, DeepL), la génération de texte (ChatGPT), l'analyse de sentiments et les chatbots intelligents.
- **Santé et biomédecine** : Aident à prédire les maladies, à analyser les images IRM et radiographies, et à développer des systèmes d'assistance au diagnostic médical.
- **Finance et économie** : Utilisés pour la prédiction des marchés boursiers, la détection des fraudes bancaires et l'évaluation des risques de crédit.

- **Industrie et automatisation** : Déployés dans l'optimisation des processus industriels, la maintenance prédictive et les systèmes de contrôle automatisés.

Grâce à leur flexibilité et leur puissance, les RNA continuent d'évoluer et de s'adapter à de nouvelles applications, rendant les systèmes intelligents de plus en plus performants et autonomes.

1.10 Conclusion

Les réseaux neuronaux artificiels ont profondément transformé l'intelligence artificielle et continuent de progresser grâce à des architectures plus performantes. Le développement des LSTM, CNN et autres modèles avancés a permis des avancées considérables dans divers domaines comme la reconnaissance visuelle, le traitement du langage naturel et la prise de décision autonome. L'amélioration de l'efficacité et de l'interprétabilité de ces modèles demeure un défi majeur pour les chercheurs.

Dans un prochain chapitre, nous explorerons en détail un type de réseau neuronal reposant sur le théorème de Kolmogorov-Arnold (KAN), une approche prometteuse pour modéliser des fonctions complexes et améliorer les performances des systèmes d'apprentissage automatique

CHAPITRE 02

Kolmogorov-Arnold Networks:

Concepts et Fondements

2.1 Introduction

Dans le contexte actuel de l'intelligence artificielle, caractérisé par une recherche constante d'architectures plus expressives et interprétables, les Kolmogorov-Arnold Networks (KANs) représentent une avancée conceptuelle majeure. Contrairement aux perceptrons multicouches (MLPs) traditionnels, qui reposent sur des transformations linéaires suivies de fonctions d'activation fixes, les KANs introduisent une nouvelle approche en associant à chaque connexion neuronale une fonction univariée entraînable.

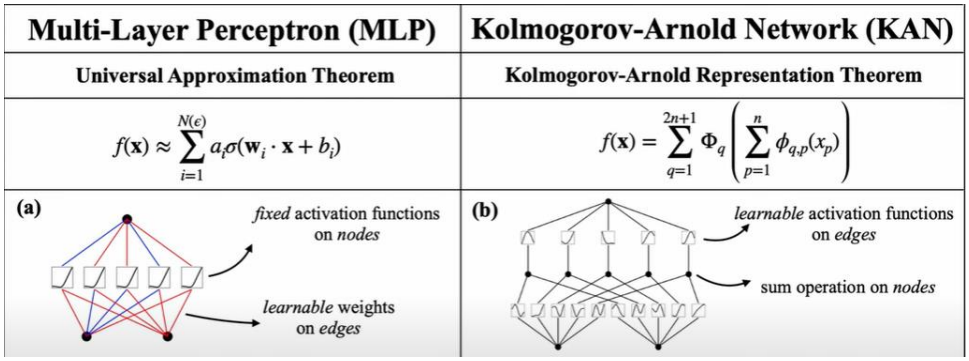


Figure (2.1) : Comparaison entre une connexion MLP et une connexion KAN.

Ce paradigme est directement inspiré du théorème de Kolmogorov-Arnold (1957) [7], selon lequel toute fonction continue multivariée peut être exprimée comme une composition de fonctions univariées continues et d'opérations d'addition. En exploitant cette propriété fondamentale, les KANs substituent les poids statiques des MLPs par des fonctions apprenables, telles que des B-splines ou des polynômes, optimisées à partir des données.

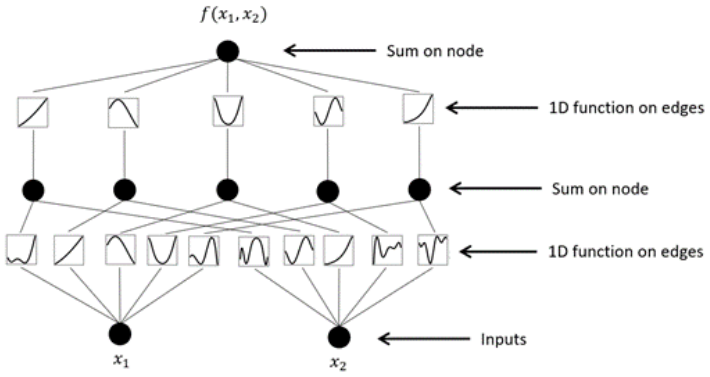


Figure (2.2) : Illustration générale du principe d'un réseau KAN

Cette architecture réduit la complexité, améliore la généralisation, et rend le modèle plus interprétable, ce qui la rend adaptée aux domaines critiques comme la santé ou la finance.

2.2 Théorème de Kolmogorov-Arnold

Le théorème de Kolmogorov-Arnold **(1957)** affirme que toute fonction continue multivariée définie sur un domaine borné peut être exactement représentée comme une superposition finie de fonctions univariées continues et d'opérations d'addition. Formellement, toute fonction

$f(x_1 \cdots x_n)$peut s'écrire sous la forme :

$$f(x_1 \dots x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \psi_{p.q}\,(x_p) \right) \qquad (2.1)$$

Où $\boldsymbol{\psi_{p\cdot q}}$ et $\Phi_q$sont des fonctions univariées continues.

Ce résultat fondamental réduit la complexité de la modélisation en haute dimension en décomposant une fonction complexe en une composition de fonctions univariées plus simples. Il constitue la base théorique de la capacité d'approximation universelle des réseaux de neurones : un réseau possédant une seule couche cachée avec suffisamment de neurones peut approximer toute fonction continue à une précision arbitraire.

Les Kolmogorov-Arnold Networks (KANs) s'appuient explicitement sur ce théorème. Contrairement aux MLPs classiques qui utilisent des poids linéaires fixes, les KANs remplacent ces poids par des fonctions univariées apprenables (comme des splines ou des polynômes). Ainsi, les KANs optimisent directement les fonctions $\Phi_q$et $\boldsymbol{\psi_{p\cdot q}}$ à partir des données, traduisant de manière constructive l'idée de Kolmogorov.

Exemple : Pour approximer la fonction $f(x_1, x_2) = {x_1}^2 + \sin(x_2)$ un KAN décompose les fonctions $\boldsymbol{\psi_{p,q}}$ et $\boldsymbol{\Phi_q}$**:**

- KAN apprend : $\psi_{1,1}(x_1) = {x_1}^2$ et $\psi_{2,1}(x_2) = \sin(x_2)$
- $\Phi_1(z) = z$, avec d'autres termes potentiellement nuls (output).

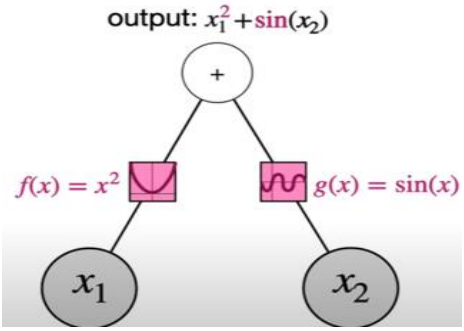


Figure (2.3) : Illustration du théorème de Kolmogorov-Arnold

Ce mécanisme offre une approche modulaire, interprétable et efficace pour l'approximation de fonctions complexes en grande dimension.

2.3 Décomposition des Fonctions Multivariées

L'un des fondements théoriques des Kolmogorov-Arnold Networks (KANs) réside dans la capacité à décomposer toute fonction multivariée continue en une composition de fonctions univariées.

La représentation canonique d'une fonction multivariée $f(x_1,x_2\ldots,x_n)$ peut être exprimée par :

$$f(x_1;x_2;\cdots;x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \psi_{p;q}\,(x_p)\right) \qquad (2.2)$$

Où les $\boldsymbol{\psi_{p,q}}$ sont des fonctions univariées continues **apprenables,** et les $\boldsymbol{\Phi_q}$ sont également des fonctions univariées continues.

- Chaque transformation $\boldsymbol{\psi_{p,q}}$ agit uniquement sur une variable individuelle.
- La composition de ces transformations, via des sommes et fonctions Φ_q, permet de reconstruire la fonction cible f.

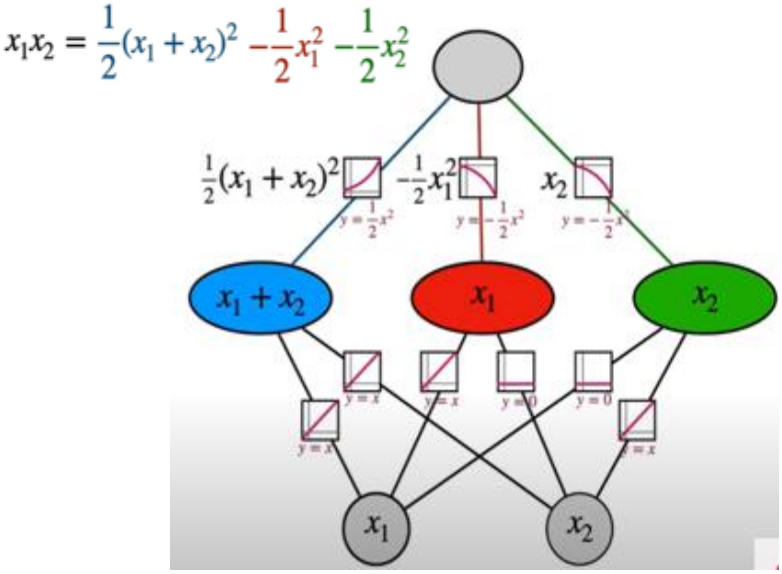


Figure (2.4) : Décomposition d'une fonction de degré 3 (cubiques).

Prenons la fonction simple $f(X_1;X_2) = X_1X_2$ lle peut être décomposée en une combinaison de trois fonctions quadratiques univariées :

- $\psi_1(x) = (½)\, x^2$ (fonction convexe),
- $\psi 2(x) = (-½)\, x^2$ (fonction concave),
- Combinaison des transformations de x_1+x_2.

Cette décomposition explicite facilite l'analyse et l'interprétation des modèles KAN, contrairement aux MLP classiques où les fonctions intermédiaires sont difficilement interprétables.

Cependant, une conséquence directe du théorème est la présence de $2n+1$ termes non nuls, ce qui peut entraîner une explosion combinatoire en haute dimension. Cette contrainte peut rendre l'apprentissage coûteux en ressources computationnelles.

2.4 Différences Architecturales entre les KAN et les MLP Classiques

Les Kolmogorov-Arnold Networks (KANs) et les Perceptrons Multicouches (MLPs) se distinguent par leur structure fondamentale et leur philosophie d'apprentissage.

Dans les MLP classiques :

- Les fonctions d'activation (ReLU, sigmoïde, etc.) sont fixes et appliquées au niveau des neurones.
- L'apprentissage consiste uniquement à ajuster les poids linéaires des connexions.

Dans les KANs :

- Les fonctions d'activation sont appliquées sur les connexions (arêtes) entre les neurones, et non au niveau des neurones eux-mêmes.
- Ces fonctions sont apprenables et souvent représentées par des B-splines, des polynômes de Chebyshev, ou d'autres formes paramétriques adaptatives.
- Ainsi, l'architecture elle-même modélise dynamiquement les relations fonctionnelles sous-jacentes.

2.5 Architecture des Réseaux Kolmogorov-Arnold (KANs)

2.5.1 Structure des Réseaux KAN

Les KANs généralisent les réseaux neuronaux traditionnels en remplaçant les combinaisons linéaires fixes par des fonctions univariées apprenables, conformément au théorème de Kolmogorov-Arnold. Leur architecture se compose des éléments suivants (Figure 2.5).

2.5.1.1 Couche d'entrée : Reçoit les variables x_1, x_2,…, x_n.

2.5.1.2 Couches cachées :

- Transformations non-linéaires : Chaque arête (connexion) porte une fonction apprenable (e.g., B-splines, polynômes de Chebyshev) plutôt qu'un simple poids.
- Agrégation : Les sorties des fonctions sont additionnées aux nœuds (neurones).

2.5.1.3 Couche de sortie : Combine les résultats des couches cachées pour trouver la prédiction finale.

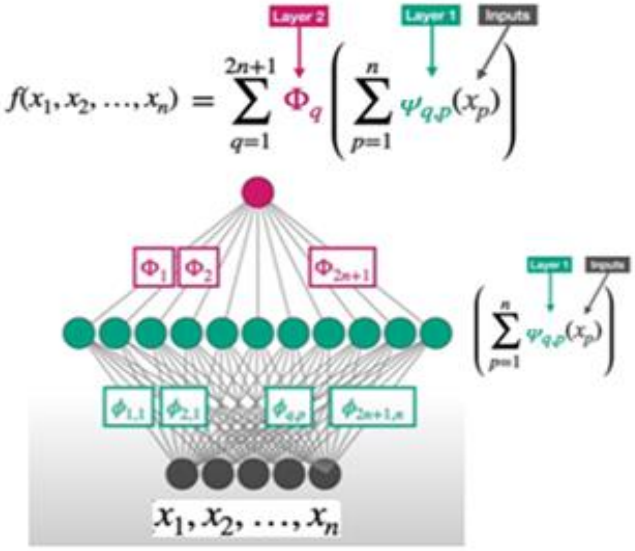


Figure (2.5) : Flux de données dans une structure du reseau KAN

2.5 B-Splines dans les Réseaux KAN

2.5.1 Définition et Types de Splines

Les splines sont des fonctions polynomiales par morceaux, définies sur des intervalles séparés par des nœuds. Contrairement aux polynômes globaux, sujets à des oscillations importantes, les splines offrent une approximation flexible et localement contrôlée figure (2.6).

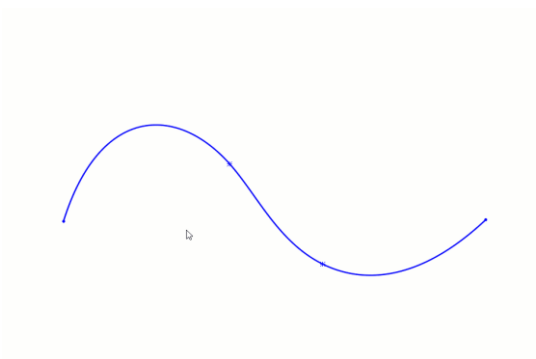


Figure (2.6) : Représentation d'une base de B-splines locales et contrôle local

2.5.1.1 Types courants de splines

- Splines linéaires : polynômes de degré 1 ; elles sont C^0 continues (continues mais avec des angles aux nœuds).
- Splines quadratiques : polynômes de degré 2 ; elles assurent une continuité C^1 (pente continue).
- Splines cubiques : polynômes de degré 3 ; elles assurent une continuité C^2 (courbure continue).
- B-Splines (Basis Splines) : combinaisons linéaires de fonctions de base localisées, offrant une grande stabilité numérique.
- Splines naturelles : contraintes supplémentaires aux extrémités (par exemple, dérivées secondes nulles).

2.5.1.2 Tableau de continuité

Classe	**Propriété**	**Exemple**
C^0	Continue	Spline linéaire
C^1	Dérivée première continue	Spline quadratique
C^2	Dérivée seconde continue	Spline cubique naturelle

Tableau (2.1) : Types de splines et leurs continuite

Concrètement : la construction des B-splines consiste à placer un ensemble de nœuds (ou points de rupture) le long de l'axe des abscisses (axe x). Sur chaque intervalle délimité par deux nœuds successifs, une fonction polynomiale est définie. L'ensemble de ces polynômes forme une fonction globale continue et régulière, assurant une transition fluide entre les segments, ce qui garantit la continuité de classe Ck selon le degré choisi. (voir Figure 2.7) [8].

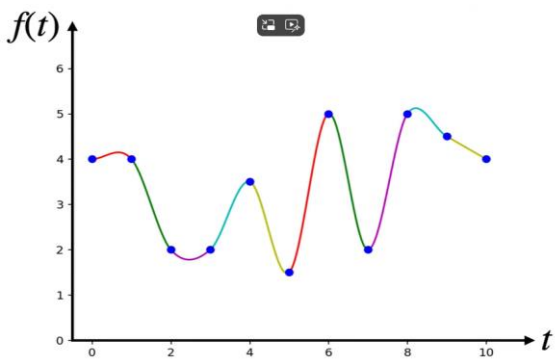


Figure (2.7): Generation d'une trajectoire avec les B-splines

2.5.2 Courbes B-Splines

Les B-splines sont des fonctions de base utilisées pour représenter des courbes ou des surfaces de manière flexible. La construction d'une B-spline de degré k, notée Bi,k(x), s'effectue de manière récursive selon la relation suivante :

$$B_{i;k}(x) = \frac{x - t_i}{t_{i+k} - t_i} B_{i;k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1;k-1}(x) \quad (2.3)$$

- k = degré du spline (ex: k=2 → polynômes quadratiques).
- ti = points appelés nœuds qui divisent l'axe x.
- $B_{i;k-1}(x)$ ET $B_{i+1;k-1}(x)$B-splines de degré k−1 (plus simples).

Une B-spline de degré k est faite en combinant deux B-splines plus simples (de degré k−1) de manière pondérée selon x. Chaque B-spline est ainsi définie sur un intervalle local, ce qui garantit une modification contrôlée de la fonction globale lors de l'apprentissage (Figure 2.8).

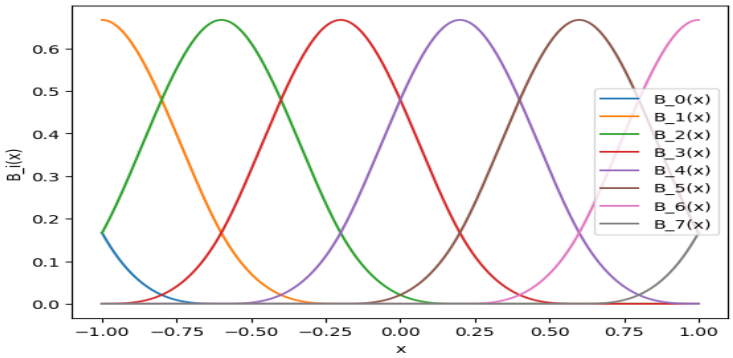


Figure (2.8) : Construction des B-splines de degré k, notée Bi,k(x),

2.5.3 Intégration des Splines dans les Réseaux KAN

Dans les réseaux KAN (Kolmogorov–Arnold Networks), une innovation majeure consiste à remplacer les poids scalaires fixes, traditionnellement utilisés dans les réseaux de neurones multicouches (MLP), par des fonctions apprises. Ces fonctions sont construites à l'aide de combinaisons de B-splines, ce qui confère au réseau une capacité d'adaptation et d'expressivité nettement supérieure [9].

Plus précisément, chaque connexion entre deux neurones dans un KAN est modélisée non pas par un simple coefficient multiplicatif, mais par une fonction univariée f(x) exprimée sous la forme d'une combinaison linéaire de B-splines :

$$f(x) = \sum_i c_i \, B_{i;k}(x) \tag{2.4}$$

- $\mathbf{B_{i,k}}$ est la i-ième B-spline de degré k (typiquement cubique avec k=3),
- $\mathbf{c_i}$ sont des coefficients appris par descente de gradient.
- Les coefficients ci sont appris par l'algorithme (comme on apprend les poids dans un réseau normal).

Cette approche permet une modélisation fine des relations entre les neurones, bien au-delà de ce qu'autorise un simple produit scalaire. Par exemple, au lieu d'apprendre à « multiplier par 0.7 », le réseau apprend une fonction complète pouvant s'adapter à des formes complexes telles que sin(x), x^2 ou toute autre variation non linéaire locale.

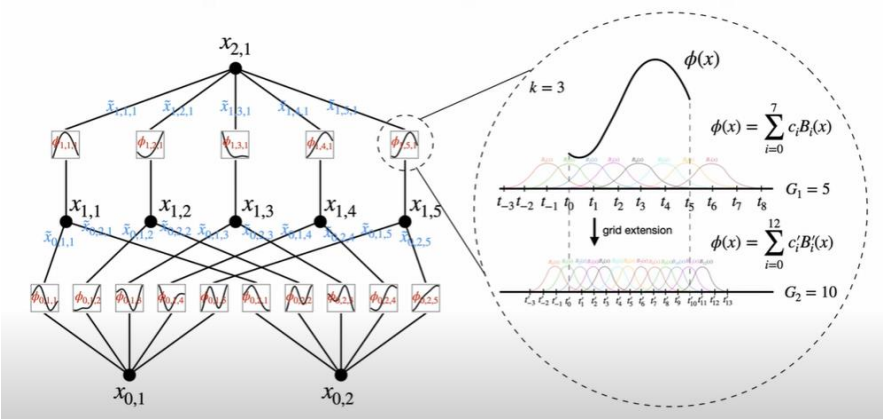
L'intégration de ces fonctions dans le KAN se fait de manière naturelle. Prenons un exemple simple où une fonction f(x), placée sur une connexion entre deux neurones, est définie comme suit:

$$f(x) = 2B_{0;1}(x) - 1 \cdot 5B_{1;1}(x) + 0 \cdot 5B_{2;1}(x) \tag{2.5}$$

$B_{i,1}(x)$ est une spline locale.

Dans cette expression, les coefficients 2, -1.5 et 0.5 sont appris au cours de l'entraînement. Cette fonction joue alors le rôle d'un poids dynamique, dont la forme peut s'adapter aux besoins de l'approximation locale, rendant ainsi le réseau plus expressif.

Dans les KAN, les B-splines sont utilisées pour paramétrer les fonctions univariées. Les coefficients des B-splines deviennent les paramètres entraînables du réseau. Cela signifie que le réseau apprend à ajuster la forme des B-splines pour mieux s'adapter aux données (Figure 2.9).



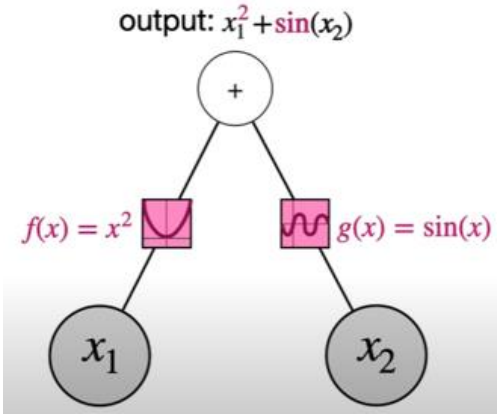
Figure(2.9) : Visualisation d'une fonction apprise par une connexion spline dans un KAN

2.5.4 Optimisation

Dans le Kolmogorov-Arnold Network (KAN) intégrant des B-splines, l'apprentissage consiste à ajuster les coefficients c_{ij} des bases via la rétropropagation du gradient. Une régularisation L_2 est appliquée pour limiter la complexité des splines et prévenir le surajustement, favorisant ainsi des approximations lisses et robustes.

Pour la fonction$f(x_1; x_2) = {x_1}^2 + \sin(x_2)$, Le réseau KAN pourrait procéder comme suit (Figure 2.10):

- $S_1(x)$ approxime sin(x) en utilisant une B-spline oscillante adaptée,
- S2(y) approxime y^2 en utilisant une B-spline de forme quadratique.



Figure(2.10) : Chaque spline ajuste localement la forme de la fonction

Chaque fonction spline ajuste localement la forme de f, capturant efficacement les non-linéarités.

2.6 Apprentissage dans un KAN

2.6.1 Initialisation du Réseau

Les Kolmogorov-Arnold Networks (KAN) se distinguent des réseaux de neurones classiques en associant à chaque connexion une fonction univariée paramétrable f_{ij}, au lieu d'un simple poids scalaire. Ces fonctions permettent une transformation plus riche et non linéaire des signaux entre nœuds [9][10].

Stratégies d'initialisation :

- Aléatoire : Perturbations autour de fonctions linéaires.
- Basées sur des fonctions de base :

- **Polynômes** : Simples mais peu flexibles.
- **Splines** : Approximations locales lisses et adaptatives.
- **Séries de Fourier** : Combinaisons de sinusoïdes, idéales pour les signaux périodiques ou complexes.

2.6.2 Propagation Avant (Forward Pass)

Dans un KAN, la propagation avant s'effectue par une suite de transformations univariées non linéaires, appliquées séparément à chaque entrée. Contrairement aux réseaux traditionnels qui utilisent des produits scalaires suivis de fonctions d'activation, le KAN applique des fonctions apprenables (ex. : splines) sur chaque dimension, puis combine les sorties par sommation. Ce mécanisme, inspiré du théorème de Kolmogorov-Arnold, permet une approximation plus fine et une meilleure interprétabilité.

Soit un vecteur d'entrée $X = (x_1\ x_2 \cdots x_n) \in \mathbb{R}^n$. Le traitement de l'information à travers le réseau suit les étapes suivantes :

2.6.2.1 Application des transformations univariées

Chaque nœud intermédiaire reçoit en entrée une composante x_i du vecteur initial, et lui applique une fonction univariée spécifique notée$F_i(x_i)$. Ces fonctions, paramétrables et apprises pendant l'entraînement, sont initialisées selon une base fonctionnelle (polynômes, splines, séries de Fourier, etc.) $Z_i = f_i(x_i)$pour tout i ∈ {1, …, n}

2.6.2.2 Combinaison fonctionnelle intermédiaire

Les sorties z_i des nœuds intermédiaires ne sont pas combinées linéairement comme dans un MLP. À la place, elles sont transmises à un ensemble de fonctions univariées supplémentaires qui assurent une composition fonctionnelle :

$Y_j = G_j(Z_i)$Pour i ∈ pred(j)), où chaque g_j est composée de transformations univariées.

Dans la pratique, chaque chemin $x_i \rightarrow z_i \rightarrow y_j$ correspond à une suite de fonctions univariées composées, sans somme pondérée explicite.

2.6.3 Définition de la fonction de perte

Dans l'apprentissage d'un réseau Kolmogorov-Arnold (KAN), la fonction de perte joue un rôle fondamental : elle quantifie l'écart entre les sorties produites par le modèle et les valeurs cibles attendues.

2.6.3.1 Évaluation de l'erreur

À chaque itération d'apprentissage, la fonction de perte mesure la qualité des prédictions réalisées par le KAN.

2.6.3.2 Cas des tâches de régression

Pour les problèmes de régression, l'erreur quadratique moyenne (Mean Squared Error, MSE), définie comme suit:

$$MSE = \frac{1}{N}\sum(y_i - \hat{y}_i)^2 \qquad (2.6)$$

• y_i est la valeur réelle (cible) associée à l'exemple i,
• $\hat{y}_i$ est la valeur prédite par le KAN pour l'exemple i,
• N est le nombre total d'exemples d'apprentissage.

2.6.3.3 Objectif de l'optimisation

Le processus d'entraînement consiste à minimiser cette fonction de perte afin d'améliorer progressivement la précision des prédictions du modèle.

2.6.3.4 Cas des tâches de classification

Si la tâche visée est une classification (par exemple, distinguer plusieurs classes), d'autres fonctions de perte plus adaptées, comme l'entropie croisée (Cross-Entropy Loss), peuvent être utilisées.

Le choix de la fonction de perte dépend donc directement de la nature du problème traité par le KAN.

2.6.4 Mise à jour des paramètres

Dans un réseau Kolmogorov-Arnold (KAN), l'apprentissage repose sur l'optimisation des paramètres internes associés aux fonctions univariées composant le réseau. Ces paramètres peuvent inclure :

- Les coefficients des B-splines (c_{ij}),
- Les amplitudes et phases dans une expansion de Fourier,
- Les positions des points de contrôle dans certains cas adaptatifs.

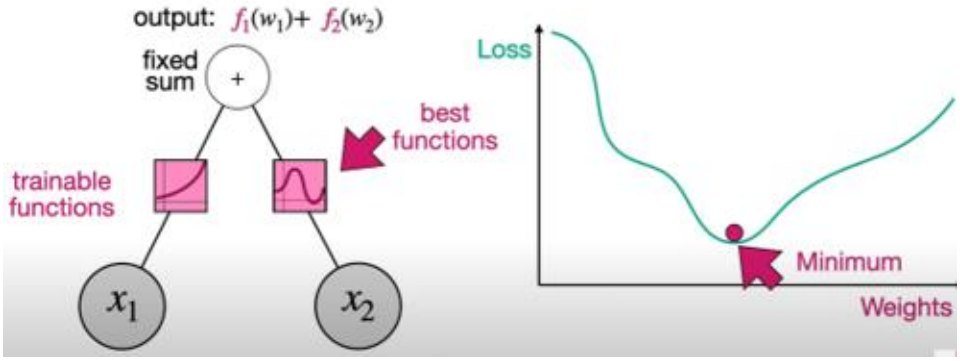
À chaque itération de l'apprentissage, ces paramètres sont ajustés de manière à minimiser la fonction de perte. Le processus suit généralement les étapes suivantes :

- Calcul du gradient de la fonction de perte par rapport à chaque paramètre.

- Mise à jour des paramètres selon la direction opposée au gradient, conformément à la règle de la descente de gradient standard ou à l'une de ses variantes modernes (par exemple : Adam, RMSprop).

Ces algorithmes permettent d'accélérer la convergence en adaptant dynamiquement le taux d'apprentissage et en tenant compte de l'historique des gradients.

Ainsi, l'optimisation dans un KAN ne concerne pas des poids scalaires comme dans les réseaux neuronaux classiques, mais plutôt l'ajustement précis de fonctions continues, conférant au modèle une capacité d'approximation fine et une meilleure interprétabilité (Figure 2.11).



Figure(2.11) : L'optimisation dans un KAN ne concerne pas des poids scalaires comme dans les MLP, mais plutôt l'ajustement précis de fonctions continues

2.6.5 Régularisation et stabilisation

Afin de **prévenir le surapprentissage** dans un réseau Kolmogorov-Arnold (KAN), plusieurs techniques de **régularisation** sont mises en œuvre pour limiter la complexité des fonctions apprises et améliorer la généralisation du modèle :

1. Régularisation L2

Ajoute une pénalité sur la norme des coefficients des fonctions (ex. : splines), ce qui encourage des solutions plus lisses et évite les oscillations excessives.

2. Réduction du nombre de nœuds

Limiter le nombre de **points de rupture (knots)** dans les B-splines permet de restreindre la flexibilité de chaque fonction univariée.

3. Early Stopping

Interrompt l'apprentissage dès que la performance sur l'ensemble de validation commence à se dégrader, empêchant un surajustement aux données d'entraînement.

4. Ajustement des hyperparamètres

Contrôle de la **taille des fenêtres**, du **taux d'apprentissage**, du **nombre de couches**, **facteur gamma (Focal Loss)** pour limiter la capacité du modèle à surapprendre.

5. Normalisation des entrées

Stabilise l'apprentissage en réduisant les écarts de distribution des données, ce qui permet une convergence plus contrôlée des fonctions apprenables.

2.6.6 Validation et test

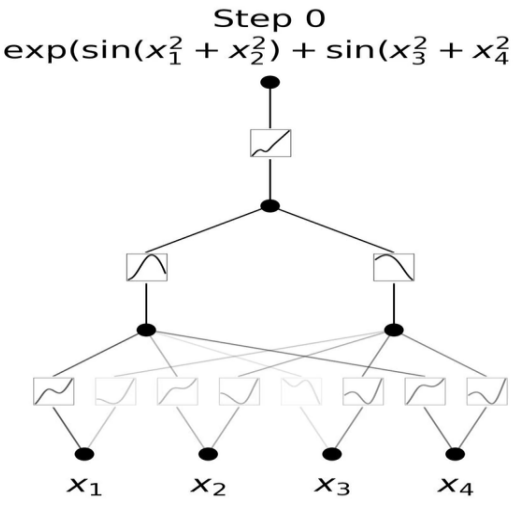
Après l'entraînement, l'évaluation du modèle est réalisée sur un ensemble de données inédites:

2.6.6.1 Métriques utilisées :

- Erreur quadratique moyenne (MSE) pour mesurer l'erreur globale de prédiction,

2.6.6.2 L'apprentissage dans un réseau Kolmogorov-Arnold se distingue par :

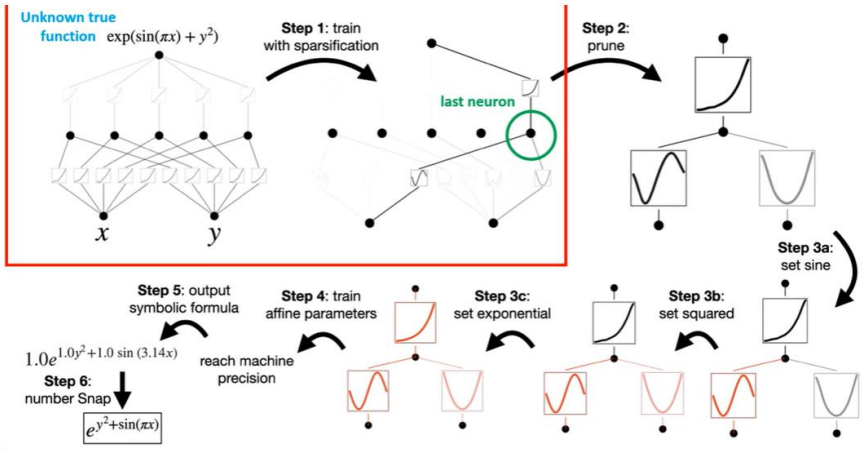
- Une architecture basée sur des transformations non linéaires univariées : Chaque lien du réseau représente une fonction continue, ce qui offre une flexibilité accrue par rapport aux réseaux neuronaux classiques.
- L'optimisation de fonctions continues : Les techniques d'optimisation sont adaptées pour ajuster des fonctions (et non de simples poids), ce qui le rend plus mais enrichit l'entraînement.
- Un compromis entre flexibilité et interprétabilité : Grâce à l'utilisation de fonctions explicites comme les splines ou les bases de Fourier, le modèle reste à la fois puissant et interprétable (Figure 2.12).



Figure(2.12) : Architecture multi-couches d'un réseau KAN complet.

2.7 Reconstruction Symbolique Interprétable d'une Fonction Cible

Cette section décrit le fonctionnement complet d'un Réseau Kolmogorov–Arnold (KAN) illustré dans la figure suivante. (Figure 2.13).



Figure(2.13) : Étapes de Fonctionnement d'un Réseau Kolmogorov–Arnold

Étape 1 : Entraînement avec parcimonie

Le modèle commence avec un réseau KAN dense, où les variables d'entrée x et y traversent des nœuds à base de splines. L'entraînement applique une technique de sparsification pour supprimer les connexions redondantes et mettre en valeur les chemins les plus significatifs pour approximer la fonction cible.[11]

Étape 2 : Élagage du réseau

Une fois le réseau entraîné, seules les connexions actives menant au dernier neurone de sortie sont conservées. Cette étape réduit la complexité du modèle et facilite l'interprétation.

Étape 3 : Reconstruction des structures fonctionnelles

Les unités à base de splines sont remplacées par des fonctions mathématiques connues :

- Étape 3a : Attribution de la fonction sinus.
- Étape 3b : Attribution de la fonction carrée.
- Étape 3c : Attribution de la fonction exponentielle.

Étape 4 : Réglage des paramètres affines

Une fois les formes fonctionnelles fixées, les coefficients affines des splines sont ajustés pour atteindre une précision maximale, assurant une adéquation exacte avec la fonction cible.

Étape 5 : Sortie de la formule symbolique

Le réseau KAN génère l'expression symbolique approximant la fonction cible : $e^{(y^2 + \sin(3.14 \cdot x))}$

Étape 6 : Ajustement des constantes (Number Snap)

Les constantes approximatives sont arrondies à des valeurs interprétables. $e^{(y^2 + \sin(\pi x))}$

Le KAN allie splines et régression symbolique pour transformer une approximation numérique en une expression mathématique lisible, offrant une IA interprétable.

2.8 Domaines d'Application des Réseaux Kolmogorov–Arnold

Les réseaux Kolmogorov-Arnold (KAN) représentent une avancée majeure en IA explicable, combinant la puissance des réseaux de neurones avec la transparence des expressions mathématiques. Leur capacité unique à dériver des formules symboliques à partir de données les rend particulièrement utiles pour :

2.8.1 Sciences physiques et naturelles

Les KANs peuvent apprendre et reconstruire des lois physiques ou des relations naturelles à partir de données expérimentales. Ils sont utilisés pour :

- Modéliser des équations différentielles et lois de conservation,
- Identifier des formules physiques en mécanique, thermodynamique, etc.,
- Étudier des systèmes complexes en biologie, chimie ou astrophysique.

2.8.2 Intelligence Artificielle Explicable (XAI)

Grâce à leur sortie sous forme symbolique, les KANs favorisent une interprétation directe des modèles appris. Ils représentent une solution idéale pour l'intelligence artificielle explicable, en particulier dans des contextes sensibles comme la médecine, la finance ou le droit, où la transparence est primordiale.

2.8.3 Médecine et bio-informatique

Les KANs peuvent aider à modéliser des relations entre variables biologiques, génétiques ou cliniques. Ils permettent de :

- Générer des modèles prédictifs interprétables,
- Étudier les interactions entre biomarqueurs et maladies,
- Réduire les risques d'effets 'boîte noire' dans les diagnostics.

2.8.4 Finance et économie

Dans ces domaines, les KANs sont utiles pour découvrir des relations fonctionnelles entre indicateurs économiques et pour modéliser des séries temporelles de manière explicite. Ils permettent également :

- L'identification de modèles de risque ou rendement,
- La formulation analytique de tendances économiques ou boursières.

2.8.5. Ingénierie et systèmes dynamiques

Les KANs peuvent identifier des systèmes physiques ou mécaniques à partir de mesures et reconstruire les équations qui les régissent. Ils trouvent des applications en :

- Robotique et commande automatique,
- Analyse de systèmes complexes et stabilité,
- Génie électrique, mécanique et aérospatial.

2.9 Conclusion

Les réseaux Kolmogorov–Arnold (KAN) représentent une avancée conceptuelle majeure dans le domaine de l'intelligence artificielle. Fondés sur le théorème de Kolmogorov-Arnold, ces réseaux offrent une nouvelle approche de l'approximation fonctionnelle en remplaçant les poids traditionnels des réseaux neuronaux par des fonctions adaptatives telles que les splines ou les séries de Fourier. Cette particularité leur confère une interprétabilité inédite, essentielle dans les domaines où la compréhension du modèle est aussi cruciale que sa performance.

Contrairement aux architectures classiques, les KAN permettent de modéliser directement des fonctions continues et non linéaires de manière explicite, tout en conservant une grande flexibilité. Leurs structures modulaires, basées sur des transformations univariées composées, ouvrent la voie à des modèles plus compacts, plus stables et souvent plus précis.

Les KAN s'inscrivent ainsi comme une alternative puissante aux réseaux de neurones profonds, notamment dans les contextes scientifiques, médicaux et industriels où l'interprétation, la robustesse et la capacité à extraire des relations symboliques sont primordiales. Leur potentiel ne cesse de croître, tant pour la découverte de lois fondamentales que pour la création de modèles prédictifs transparents et fiables.

CHAPITRE 03

DETECTION D'ANOMALIES PAR LES KANS

3.1 Introduction

La détection d'anomalies est essentielle dans des domaines sensibles tels que la cybersécurité, la finance ou la santé, où elle permet d'identifier des comportements inhabituels révélateurs de risques ou de défaillances [10]. Dans les séries temporelles, cette tâche est particulièrement complexe en raison de la dynamique, de la périodicité et de la non-linéarité des données.

Pour y répondre, une approche combinant la Transformée de Fourier naïve et les réseaux Kolmogorov-Arnold (KANs) est proposée. La décomposition fréquentielle met en évidence les tendances normales, tandis que les KANs, grâce à leur capacité d'approximation, modélisent les structures complexes et facilitent la détection d'écarts anormaux.

Cette méthode permet d'extraire des représentations plus informatives tout en conservant la structure temporelle des données. Elle vise ainsi à renforcer la précision et l'interprétabilité des résultats obtenus sur des séries réelles et synthétiques.

3.2 Types d'Anomalies

La détection d'anomalies repose en grande partie sur la nature de l'écart observé par rapport au comportement attendu. Trois catégories principales d'anomalies peuvent être distinguées dans les séries temporelles :

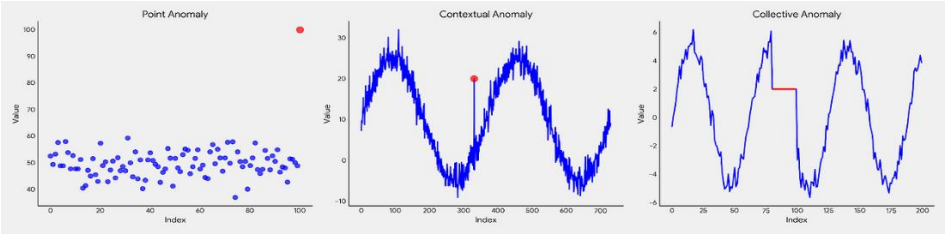


Figure (2.1) : 3 types d'anomalies

3.2.1 Anomalies Ponctuelles

Une anomalie ponctuelle se produit lorsqu'un seul point de données est très différent des autres. Par exemple, si l'utilisation moyenne du CPU est de 30 % et qu'un pic atteint 90 %, il s'agit d'une anomalie ponctuelle. Ce type d'anomalie est souvent un signe précurseur d'une panne ou d'une surcharge.

3.2.2 Anomalies Contextuelles

Les anomalies contextuelles surviennent lorsque des données semblent normales dans un contexte donné mais anormales dans un autre. Par exemple, une hausse du trafic réseau peut

être normale en journée, mais si elle se produit la nuit, elle pourrait indiquer un problème de sécurité.

3.2.3 Anomalies Collectives

Les anomalies collectives se caractérisent par un groupe de points de données qui, pris individuellement, peuvent sembler normaux, mais qui ensemble dévient d'un comportement attendu. Ces anomalies sont souvent liées à des phénomènes progressifs, tels que des dérives lentes ou des pannes complexes, et sont particulièrement pertinentes dans le cadre des séries temporelles.

3.3 Méthodes de Détection d'Anomalies

Les techniques de détection d'anomalies peuvent être classées en trois grandes catégories, en fonction de la disponibilité des données étiquetées : les techniques supervisées, les techniques non supervisées et les techniques semi-supervisées [11]

3.3.1 Méthodes Supervisées

Les méthodes supervisées nécessitent des données étiquetées, c'est-à-dire annotées avec la présence ou l'absence d'anomalies. Ces méthodes utilisent ces données pour entraîner un classificateur capable de distinguer les points de données normaux des anomalies. Elles se divisent en deux sous-catégories :

- **Classification des anomalies** : Formule le problème comme une classification binaire où le modèle prédit si un point est normal ou anormal.
- **Prédiction d'anomalies** : Utilise des modèles de régression ou de prévision pour comparer la valeur attendue d'un point de données à sa valeur réelle.

3.3.2 Méthodes Non Supervisées

Les méthodes non supervisées ne nécessitent pas de données étiquetées. Elles reposent sur l'hypothèse que les anomalies sont rares et diffèrent significativement des données normales. Ces méthodes identifient les écarts par rapport à des schémas préétablis et sont particulièrement efficaces pour la détection d'anomalies émergentes.

3.3.3 Méthodes Semi-Supervisées

Les méthodes semi-supervisées exploitent à la fois des données étiquetées et non étiquetées. Elles apprennent un comportement normal à partir des données étiquetées, puis détectent les

anomalies en analysant les écarts dans les données non étiquetées. Cette approche est particulièrement utile pour les jeux de données où les anomalies sont rares et difficiles à annoter.

3.3.4 Autres Approches de Détection d'Anomalies

Bien que le Machine Learning soit couramment utilisé, d'autres techniques existent :

- **Méthodes statistiques** : Basées sur des distributions de probabilités et des tests d'hypothèses (ex: modèle Gaussien).
- **Approches basées sur des règles** : Détection d'anomalies par définition explicite de seuils ou de motifs anormaux.
- **Techniques de traitement des signaux** : Analyse spectrale et filtrage pour isoler les anomalies dans les données complexes.

3.4 Application des KANs à la Détection d'Anomalies

Les Kolmogorov-Arnold Networks (KANs) sont une alternative prometteuse à la détection d'anomalies dans les séries temporelles, car ils offrent une approximation universelle des fonctions complexes, ce qui constitue un avantage distinctif important. Grâce à cette capacité d'approximation, les KANs peuvent capturer les dépendances non linéaires et s'adaptent particulièrement bien à la modélisation des séries temporelles. De plus, les KANs peuvent s'adapter aux nouvelles anomalies grâce à l'utilisation de la transformée de Fourier dans l'apprentissage, permettant ainsi une meilleure capture des motifs périodiques et des variations complexes dans les données temporelles [12].

3.4.1 Applications de la détection d'anomalies dans les séries temporelles

La détection d'anomalies dans les séries temporelles trouve des applications dans divers domaines, notamment :

- **Cybersécurité** : Détection d'intrusions ou d'activités suspectes dans les réseaux informatiques.
- **Finance** : Identification de transactions frauduleuses ou de comportements de marché anormaux.
- **Santé** : Surveillance des signaux physiologiques pour détecter des anomalies pouvant indiquer des problèmes de santé.
- **Industrie manufacturière** : Surveillance des équipements pour prédire les pannes et optimiser la maintenance.

- **Services cloud** : Surveillance des indicateurs de performance (KPI) pour garantir la disponibilité et la performance des services.

3.4.2 Défis de la détection d'anomalies dans les séries temporelles

Malgré son importance, la détection d'anomalies dans les séries temporelles présente plusieurs défis majeurs :

- **Données déséquilibrées** : Les anomalies sont souvent rares par rapport aux données normales, ce qui rend difficile l'entraînement des modèles.
- **Bruit dans les données** : Les séries temporelles peuvent contenir des variations locales (pics et creux) qui ne sont pas nécessairement des anomalies, mais qui peuvent perturber les modèles.
- **Évolution des anomalies** : Les comportements normaux et anormaux peuvent évoluer avec le temps, nécessitant des modèles adaptatifs.
- **Interprétabilité** : De nombreux modèles de détection d'anomalies sont des "boîtes noires", ce qui rend difficile l'explication des décisions prises.

3.4.3 Remplacement des fonctions B-spline par la transformée de Fourier

L'architecture classique du réseau Kolmogorov-Arnold Network (KAN) repose sur l'utilisation de fonctions B-spline comme mécanisme de paramétrisation des fonctions univariées issues de la décomposition du théorème de Kolmogorov. Ces fonctions locales permettent une approximation efficace de structures non linéaires, notamment grâce à leur souplesse et leur capacité à représenter des transitions abruptes dans les données. Cependant, leur portée strictement locale peut s'avérer limitée pour la capture de structures globales, en particulier celles à caractère cyclique ou périodique, souvent présentes dans les séries temporelles issues de phénomènes naturels, biologiques ou industriels.

Dans ce travail, nous proposons une alternative innovante : le remplacement des fonctions B-spline par des bases sinusoïdales issues de la transformée de Fourier. Cette substitution repose sur l'observation que les séries temporelles peuvent être avantageusement représentées par leurs composantes fréquentielles, surtout lorsqu'elles présentent des régularités rythmiques ou des oscillations non triviales. En effet, la transformée de Fourier permet de décomposer un signal en une série de fonctions trigonométriques (sinus et cosinus) de différentes fréquences, capturant ainsi la dynamique globale du signal.

Ainsi, chaque fonction univariée ϕi(x) dans le réseau KAN est reformulée comme une combinaison linéaire de fonctions sinusoïdales :

$$\emptyset_i(x) = \sum_{k=1}^{K} a_{ik} \cos(2\pi f_k x) + b_{ik} sin(2\pi f_k\, x) \quad (3.1)$$

Où f_k représente les fréquences fondamentales sélectionnées, et aik,bik sont des coefficients optimisés via rétropropagation. Cette formulation permet à chaque composante du modèle d'apprendre directement la contribution spectrale associée à une variable indépendante donnée.

L'intérêt de cette approche est double. D'une part, elle offre un mécanisme d'apprentissage spectral, dans lequel les propriétés dynamiques du signal sont encodées dans l'espace fréquentiel, renforçant la capacité du modèle à représenter des régularités périodiques et à filtrer le bruit aléatoire. D'autre part, elle crée une passerelle naturelle vers la détection d'anomalies : les fréquences apprises servent de référence lors de la phase de test, permettant l'identification de déviations spectrales inhabituelles, souvent révélatrices d'un comportement anormal.

3.5 Méthodologie

3.5.1 Collecte et Préparation des Données

Un prétraitement rigoureux est nécessaire pour assurer une modélisation efficace des anomalies.

Les données proviennent de séries temporelles synthétiques et réelles, incluant le jeu de données.

Le prétraitement inclut :

- **Normalisation** : StandardScaler pour centrer et réduire les valeurs.
- **Fenêtrage** : Capture des tendances locales avec une approche par fenêtres glissantes.
- **Étiquetage des fenêtres** : Classification des segments en classes normales et anormales.
- **Gestion des données déséquilibrées** : Utilisation de SMOTE pour générer des échantillons synthétiques.

3.5.1.1 Normalisation

Les séries temporelles présentent souvent des échelles et des distributions hétérogènes. Par exemple, dans une base de données biomédicale, certaines variables varient entre 0 et 1 (taux d'oxygène), tandis que d'autres peuvent atteindre 10 000 (battements cardiaques cumulés).

Cette disparité peut biaiser l'apprentissage. L'utilisation du **StandardScaler** de scikit-learn permet de centrer les données autour de 0 et de réduire l'échelle des variations.

3.5.1.2 Fenêtrage des Séries Temporelles

Les modèles d'apprentissage automatique ne peuvent pas traiter directement des séries infinies. Une approche par **fenêtres glissantes** est utilisée :

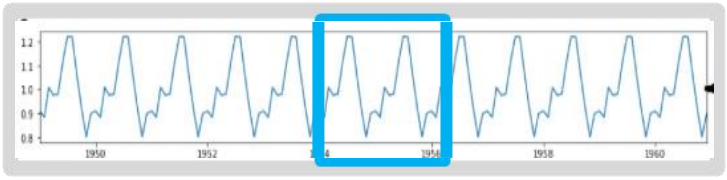


Figure 2.3 : Fenêtrage des Séries Temporelles

- Chaque fenêtre est une sous-séquence de taille définie.
- Le décalage entre fenêtres permet de capturer les tendances locales sans perte d'information.
- Une fenêtre trop petite capte du bruit, tandis qu'une fenêtre trop grande lisse les détails importants.

3.5.1.3 Étiquetage et Gestion des Données Déséquilibrées

Chaque fenêtre est classée comme **normale (0)** ou **anormale (1)** en fonction des valeurs anormales qu'elle contient. Le problème du déséquilibre des classes (énorme majorité de fenêtres normales) est atténué par SMOTE (Synthetic Minority Over-sampling Technique), qui génère de nouvelles fenêtres anormales par interpolation entre les anomalies existantes [13]. SMOTE est particulièrement adapté ici parce que les fenêtres de séries temporelles peuvent être interpolées dans l'espace vectoriel, ce qui permet de créer de nouvelles instances d'anomalies tout en préservant les caractéristiques de la séquence. Cela n'est pas toujours trivial avec des données temporelles brutes, où l'ordre et la structure des événements sont cruciaux.

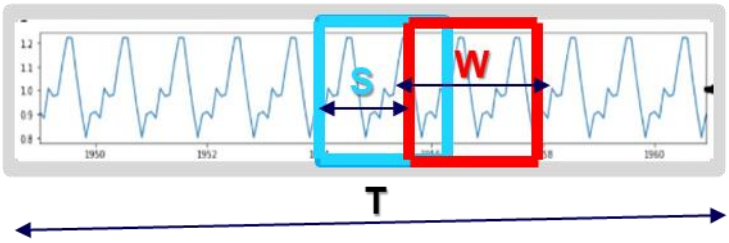


Figure 2.4 : **Fenêtre glissante a travers les donnees**

3.5.2 Construction du modèle KAN

L'objectif est de concevoir un modèle basé sur les Kolmogorov-Arnold Networks (KANs) pour détecter les anomalies dans les séries temporelles. Cette construction repose sur deux elements clés :

1. **Approximation par la Transformée de Fourier naïve** : Cette étape permet de décomposer les séries temporelles en composantes sinusoïdales pour capturer leurs tendances sous-jacentes.
2. **Utilisation des couches KAN** : Ces couches permettent de modéliser des relations complexes entre les variables et de détecter les anomalies avec plus de précision.

3.5.2.1 Approximation par la Transformée de Fourier naïve

Les séries temporelles sont souvent composées de motifs répétitifs ou périodiques (ex. variations de température, fluctuations du marché boursier). La Transformée de Fourier permet de représenter une série sous forme d'une combinaison de signaux sinusoïdaux, facilitant l'identification des tendances normales et des anomalies [14]. Cependant, un modèle de Fourier simple est insuffisant seul car il ne peut capturer que les motifs périodiques et linéaires présents dans les données, sans prendre en compte les dépendances non linéaires ou les anomalies complexes. C'est pourquoi les Kolmogorov-Arnold Networks (KANs) sont ajoutés : leur capacité d'approximation universelle permet de modéliser des comportements non linéaires et de mieux détecter des anomalies qui ne peuvent pas être expliquées par une simple décomposition fréquentielle. Les Étapes de l'Approximation de Fourier sont les suivantes.

A- Transformation des séries temporelles en une représentation sinusoïdale

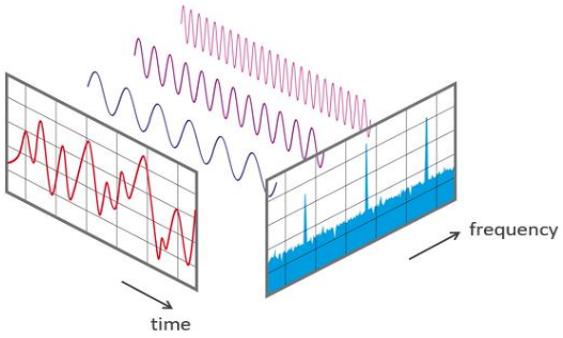
L'objectif initial est de transformer les séries temporelles brutes en une forme sinusoïdale. Cette transformation permet de capturer les caractéristiques périodiques et les fluctuations temporelles d'un signal qui sont cruciales pour identifier des anomalies subtiles.

Les données brutes (t) sont converties en une combinaison de sinusoïdes et cosinusoïdes :

$$f(t) \approx \sum_{K=1}^{N} a_k \cos(2\pi kt) + b_k \sin(2\pi kt) \tag{3.2}$$

- a_k **,** b_k sont les coefficients d'amplitude,
- **k** est la fréquence de la composante sinusoïdale,
- **t** représente le temps

Cette décomposition permet d'extraire les informations de fréquence du signal, offrant ainsi une vue détaillée de ses tendances périodiques et de ses fluctuations.



Figure(2.5) : Illustration de la Transformation de Fourier d'une série temporelle

Dans la figure 2.5, Le graphique de gauche montre le signal brut dans le domaine temporel, composé de variations complexes. Ce signal est décomposé en une somme de sinusoïdes de différentes fréquences etamplitudes (centre), révélant ses structures périodiques cachées.

Le graphique de droite présente le spectre de fréquence, où chaque pic indique une fréquence dominante. Dans le contexte de la détection d'anomalies, cette transformation joue un rôle fondamental :

- Décomposition : Un comportement normal du signal est associé à des fréquences bien identifiées et stables.
- Reconstruction : En combinant uniquement les composantes dominantes, on peut reconstruire une version "attendue" du signal.
- Détection : Si la différence (erreur de reconstruction) entre le signal réel et le signal reconstruit dépasse un certain seuil, cela indique potentiellement une anomalie.

Une anomalie est repérée lorsqu'un comportement du signal ne peut pas être expliqué par ses composantes fréquentes habituelles. Cette approche permet donc de capter à la fois des anomalies subtiles (variation légère de fréquence) et des anomalies brutales (nouvelles fréquences soudaines).

B- Apprentissage des coefficients de Fourier

Une fois la série transformée, les coefficients de Fourier ($\mathbf{a_k}$, $\mathbf{b_k}$) sont initialisés aléatoirement. L'étape suivante consiste à les ajuster via des techniques d'optimisation, telles que la descente de gradient, pour minimiser l'erreur entre la série originale et sa reconstruction.

Ce processus de réajustement (apprentissage) des coefficients permet d'affiner la représentation sinusoïdale de la série temporelle, en s'assurant que les caractéristiques essentielles du signal sont correctement capturées. Ce modèle de Fourier permet ainsi de mieux comprendre la dynamique sous-jacente du signal, tout en réduisant l'impact des bruits et des anomalies non pertinentes.

C- Reconstruction du signal et détection des anomalies

Après l'apprentissage des coefficients, la série temporelle reconstruite est obtenue en recomposant la somme des sinusoïdes et cosinusoïdes :

$$f(t) \approx \sum_{K=1}^{N} a_k \cos(2\pi kt) + b_k \sin(2\pi kt) \tag{3.2}$$

La différence entre la série originale x(t) et la série reconstruite x^(t) est ensuite calculée. Si cette différence dépasse un seuil dynamique, cela peut être le signe qu'une anomalie est présente. Ce seuil peut être ajusté en fonction de la nature des données et des exigences du modèle, et il est souvent déterminé par des techniques telles que l'analyse de la variance ou des métriques statistiques sur l'erreur de reconstruction.

Les fenêtres dont la reconstruction s'écarte fortement de l'original sont considérées comme anormales, ce qui permet de détecter des anomalies même lorsque des motifs périodiques masquent des comportements inhabituels..

3.5.2.2 Définition des Couches du Modèle KAN

L'architecture du modèle repose sur plusieurs couches, chacune jouant un rôle clé dans l'apprentissage et la détection des anomalies comme indique dans la Figure 2.6.

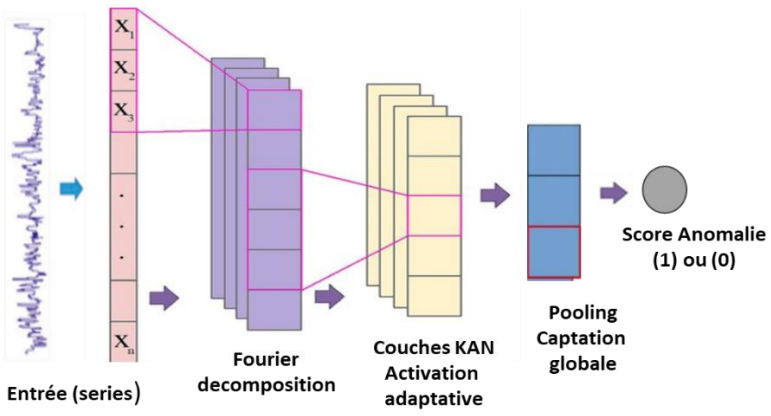


Figure (2.6): Architecture du model KAN

Couche d'entrée

- Prend en entrée les séries temporelles brutes et les transforme en une représentation adaptée.
- Applique une transformation linéaire sur les données pour stabiliser l'apprentissage.

Décomposition de la couche KAN

- Projection dans un espace non linéaire avec des activations dynamiques basées sur les fonctions trigonométriques.
- Application de la Transformée de Fourier naïve pour extraire les motifs temporels sous-jacents.

Couche de Pooling Global

- Agrège les caractéristiques sur l'ensemble de la fenêtre temporelle pour obtenir une représentation globale et éviter l'explosion du nombre de paramètres.

Couche de Sortie

- Produit un score d'anomalie via une activation sigmoïde.
- Si le score dépasse un seuil défini, la fenêtre est classée comme anormale.
- La couche sigmoïde agit comme un seuil en convertissant la sortie en une probabilité d'anomalie, facilitant ainsi la prise de décision.

3.5.3 Entraînement du modèle

L'entraînement du modèle joue un rôle crucial dans l'optimisation des performances et la généralisation aux nouvelles données. Dans cette section, nous détaillons la fonction de perte choisie, son rôle, ainsi que ses paramètres clés.

3.5.3.1 Fonction de Perte : Focal Loss

Dans le cadre de notre étude, nous avons adopté la **Focal Loss** pour contrer le déséquilibre des classes, un problème fréquent en apprentissage profond, notamment dans la détection d'anomalies [15].

L'objectif principal de cette fonction de perte est de **réduire l'impact des classes majoritaires** et de **mettre davantage l'accent sur les classes minoritaires** (ex. : anomalies rares).

Contrairement à l'entropie croisée classique, la Focal Loss ajuste dynamiquement la contribution de chaque exemple en fonction de sa difficulté.

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t) \quad (3.3)$$

Dans la **formule de la Focal Loss**, pt représente **la probabilité prédite** (défini selon le vrai label).

- Si la vraie classe est 1 (anomalie par exemple), alors pt=p (la probabilité prédite pour la classe1).
- Si la vraie classe est 0 (normal), alors $p_t = 1 - P$ (la probabilité prédite pour la classe 0).
- α est un coefficient d'équilibrage qui donne plus de poids à la classe minoritaire.
- γ est le paramètre de focalisation qui contrôle à quel point on réduit l'importance des exemples faciles à prédire associée à la classe correcte.

Application : Imaginons un problème de détection d'anomalies :

- La classe normale = 0
- La classe anomalie = 1

Le modèle donne une **probabilité prédite** p pour la classe 1 (anomalie). Vrai label = 1 (donc une anomalie réelle)

- Probabilité prédite p=0.7 → le modèle est "assez sûr" que c'est une anomalie.
- Paramètres de la Focal Loss : α=0.25, γ=2

Puisque **le vrai label est 1**, alors : pt=p=0.7 et FL(pt) = 0.008025

Situation	pt	Focal Loss	Commentaire
Bonne prédiction (p=0.7)	0.7	petite perte	Peu de pénalité
Mauvaise prédiction (p=0.2)	0.2	grande perte	Grosse pénalité pour corriger l'erreur

Tableau (3.1) : interpretation de l'erreur FocalLoss

Cette fonction permet donc de mieux entraîner le modèle sur les cas complexes tout en réduisant la domination des classes majoritaires.

3.5.3.2 Optimisation

L'optimisation joue un rôle essentiel dans l'apprentissage des modèles de deep learning, visant à minimiser la fonction de perte en ajustant les poids du réseau. Dans ce travail, l'optimiseur

Adam (Adaptive Moment Estimation) a été utilisé en raison de sa robustesse et de sa capacité d'adaptation aux variations du gradient.

Adam combine les atouts de SGD (convergence stable) et de RMSprop (ajustement adaptatif du taux d'apprentissage), permettant ainsi :

- Un ajustement dynamique du taux d'apprentissage par paramètre,
- Une meilleure stabilité face aux fluctuations du gradient.

Cette approche favorise une convergence rapide et stable du modèle, tout en assurant une mise à jour efficace des paramètres.

3.5.3.3 Ajustement Dynamique du Learning Rate

Le learning rate (η) est un hyperparamètre déterminant dans l'entraînement d'un modèle, car il influence directement la vitesse et la stabilité de la convergence. Un taux mal choisi peut soit empêcher la convergence (η trop élevé), soit ralentir considérablement l'apprentissage (η trop faible).

Pour pallier ce problème, un scheduler d'apprentissage a été utilisé. En particulier, la méthode ReduceLROnPlateau ajuste automatiquement le learning rate lorsque la performance du modèle sur le jeu de validation stagne. Cette stratégie :

- Surveille la fonction de perte sur l'ensemble de validation,
- Réduit le learning rate d'un facteur (par défaut 0.1) après N epochs sans amélioration.

3.5.3.4 Critères d'Arrêt : Early Stopping

L'entraînement d'un modèle de deep learning peut être long et sujet à un sur-ajustement si le modèle continue d'apprendre après avoir atteint son point de performance optimal. Pour éviter ce problème, nous avons utilisé la technique d'Early Stopping, qui permet d'arrêter automatiquement l'entraînement lorsque le modèle ne s'améliore plus. Cette méthode réduit également le risque de sur-apprentissage sur les données bruitées, ce qui est particulièrement pertinent dans le contexte de la détection d'anomalies, où les séries temporelles peuvent contenir des fluctuations non significatives ou du bruit.

- Surveille la perte sur l'ensemble de validation à chaque epoch.
- Si la perte ne s'améliore plus après un certain nombre d'epochs (N), l'entraînement est arrêté pour éviter un apprentissage inutile.

Grâce à cette approche, le modèle obtenu est plus généralisable et l'entraînement est plus efficace en termes de ressources et de temps.

3.5.4 Évaluation des Performances

L'évaluation d'un modèle de détection d'anomalies repose sur plusieurs métriques complémentaires, chacune apportant un éclairage spécifique sur la qualité des prédictions.

Les performances du modèle sont évaluées à l'aide de plusieurs indicateurs fondamentaux :

- **TP (**True Positives) : Anomalies correctement détectées.
- **FP** (False Positives) : Fausses alertes (prédites comme anormales mais normales).
- **FN** (False Negatives) : Anomalies non détectées.
- **TN** (True Negatives) : Fenêtres normales bien classées.

Métrique	Formule	Interprétation
Precision	$\frac{TP}{TP + FP}$	% des anomalies prédites qui sont réellement des anomalies
Rappel (Recall)	$\frac{TP}{TP + FN}$	% des anomalies réelles correctement détectées
F1-Score	$\frac{2 \times Precision \times Recall}{Precision + Recall}$	Équilibre entre précision et rappel
AUC-ROC (Area Under the Curve – Receiver Operating	Aire sous la courbe ROC	Capacité du modèle à distinguer anomalies et normales

Tableau 2 : metrique d'evaluation

- **La précision (accuracy)** mesure la proportion globale de prédictions correctes, mais peut se révéler trompeuse en présence d'un **déséquilibre de classes**, en favorisant la classe majoritaire.

- **Le rappel (recall)** indique la capacité du modèle à détecter tous les exemples positifs réels, ce qui est crucial pour **minimiser les faux négatifs**.
- **La précision (precision)**, quant à elle, mesure la proportion de prédictions positives qui sont réellement correctes, importante pour **réduire les faux positifs**.
- Le **F1-score** constitue une moyenne harmonique entre la précision et le rappel, offrant un **compromis équilibré** entre ces deux aspects.
- L'**AUC-ROC** (Area Under the Receiver Operating Characteristic Curve) quantifie la capacité du modèle à **discriminer les classes :**
    - Une AUC proche de 1 indique une excellente performance.
    - Une AUC proche de 0.5 suggère un comportement aléatoire.

L'utilisation combinée de ces métriques permet une évaluation robuste et contextuellement pertinente du modèle. Leur interprétation visuelle via la matrice de confusion (éventuellement enrichie de repères colorés) facilite la compréhension du comportement du modèle face aux différentes classes.
Enfin, ces indicateurs guident l'ajustement des hyperparamètres, des seuils de détection, et permettent une comparaison objective entre différentes approches ou configurations.

3.7 Conclusion

Les Kolmogorov-Arnold Networks (KANs) représentent une avancée prometteuse dans le domaine de la détection d'anomalies, en particulier pour les séries temporelles complexes. En combinant leur capacité d'approximation universelle non linéaire avec une Transformée de Fourier naïve, notre approche permet de capturer efficacement à la fois les motifs périodiques et les fluctuations imprévues du signal.

Cette synergie entre analyse fréquentielle et modélisation neuronale offre une solution robuste, flexible et interprétable, capable de s'adapter aux évolutions dynamiques des données réelles. Elle ouvre également la voie à des approches hybrides plus générales intégrant d'autres transformations spectrales ou spatio-temporelles.

Le chapitre suivant sera consacré à l'implémentation pratique de cette approche hybride, en détaillant les choix techniques, les outils utilisés et les résultats expérimentaux.

CHAPITRE 04

Implémentation des (KAN) pour la détection d'anomalies

4.1 Introduction

Ce chapitre présente une implémentation pratique des réseaux Kolmogorov-Arnold (KAN) pour détecter des anomalies dans des séries temporelles ECG. Contrairement aux méthodes classiques comme les LSTM, les KAN utilisent des transformations fréquentielles adaptatives qui capturent mieux les motifs complexes des signaux cardiaques.

Notre approche combine :

- Un prétraitement rigoureux (segmentation, normalisation)
- Une architecture KAN innovante avec couches Fourier pour l'analyse spectrale
- Un entraînement optimisé (Focal Loss, SMOTE) pour gérer les déséquilibres

Cette implémentation démontre le potentiel des KAN pour des applications médicales critiques, tout en fournissant un cadre reproductible (Python/PyTorch) pour de futures recherches. Les sections suivantes détaillent chaque étape, des choix d'architecture à l'analyse des performances.

4.2 Environnement et Outils Utilisés

L'implémentation du réseau Kolmogorov-Arnold (KAN) a été développée dans un environnement Python optimisé pour le calcul scientifique, garantissant reproductibilité et flexibilité.

4.2.1 Plateformes de Développement

- Jupyter Notebook : Prototypage interactif et visualisation des résultats
- Google Colab : Accès à des ressources GPU pour l'entraînement
- VS Code / PyCharm : Développement modulaire et documentation

4.3 Prétraitement des Données

4.3.1 Génération et Injection d'Anomalies

Afin d'évaluer rigoureusement les performances du modèle KAN, nous avons procédé à la génération contrôlée d'un jeu de données synthétiques. La série temporelle de base, composée de 5000 points correspondant à environ 138 périodes d'un signal sinusoïdal (fréquence fondamentale f=0.01Hz), a servi de référence pour l'injection de trois catégories d'anomalies :

- Des perturbations transitoires (*ponctuelles*),
- Des déviations localisées (*contextuelles*),
- Des altérations structurelles (*collectives*).

Le processus d'injection suit un protocole rigoureux :

- Sélection aléatoire des centres de fenêtres anormales
- Attribution équiprobable du type d'anomalie
- Modification ciblée selon le profil sélectionné

4.3.2 Segmentation et Normalisation

La segmentation des séries temporelles a été optimisée selon la formulation mathématique suivante : $N= [(T-w)/s]+1$

Où :

- T = 5000 (longueur totale)
- w = 20 (taille de fenêtre)
- s = 10 (pas de décalage)

Ce qui donne : N = 499 fenetres

4.3.2.1 Caractéristiques du Fenêtrage :

- Chevauchement de 50% entre fenêtres consécutives
- Couverture complète des 5000 points initiaux

Chaque fenêtre est ensuite labellisée selon la présence ou l'absence d'anomalies détectables :

- Classe 0 (normale) : aucune anomalie n'est présente dans la fenêtre.
- Classe 1 (anormale) : au moins un point de la fenêtre est identifié comme anormal.

Ce découpage permet de transformer la tâche de détection d'anomalies en classification binaire supervisée, simplifiant ainsi l'entraînement du modèle.

4.3.2.2 Normalisation :

Chaque fenêtre a été standardisée selon : $z_i = (x_i - \mu)/\sigma$

avec μ et σ calculés sur l'ensemble d'entraînement uniquement, garantissant ainsi l'indépendance des jeux de validation et test.

4.3.3 Rééquilibrage et Partitionnement

4.3.3.1 Stratégie de Rééquilibrage :

- Application de la technique SMOTE (Synthetic Minority Over-sampling Technique)
- Correction du ratio initial pour obtenir une distribution 40/60 (anormal/normal)
- Génération de 186 échantillons synthétiques supplémentaires

4.3.3.2 Partitionnement des Données :

- Division stratifiée 60/20/20 (entrainement, validation et test, respectivement).
- Conservation des proportions d'anomalies dans chaque sous-ensemble
- Répartition finale :
    - Entraînement : 299 fenêtres (61 anomalies)
    - Validation : 100 fenêtres (21 anomalies)
    - Test : 100 fenêtres (21 anomalies)

4.4 Conception du Modèle KAN

La conception du Kolmogorov-Arnold Network (KAN) pour la détection d'anomalies dans les séries temporelles repose sur une architecture modulaire et extensible, tirant profit à la fois de la structure théorique du théorème de Kolmogorov-Arnold et de techniques modernes d'analyse fréquentielle.

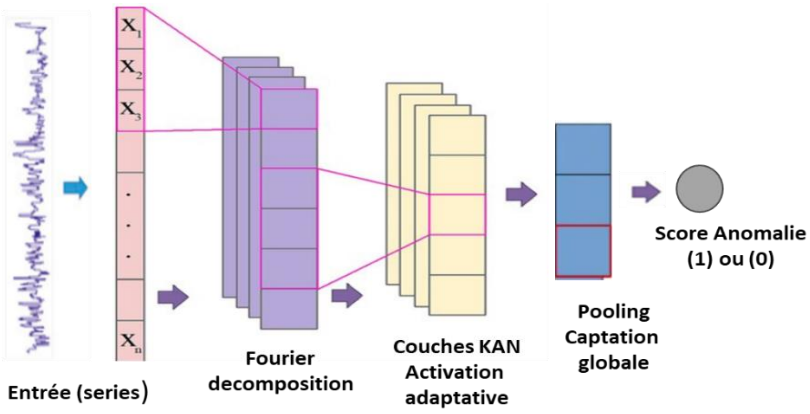


Figure 4.1 – Architecture générale du modèle KAN pour la détection d'anomalies

4.4.1 Naïve Fourier KAN Layer : Une Projection Spectrale Apprenante

L'élément central de notre architecture repose sur l'introduction d'une couche de transformation fréquentielle paramétrique, désignée sous le nom de Naïve Fourier KAN Layer. Cette couche vise à projeter les signaux temporels dans un espace spectral enrichi, dans le but de faciliter la détection de motifs périodiques, quasi-périodiques ou irréguliers, caractéristiques des phénomènes complexes observés dans les séries temporelles.

la Naïve Fourier KAN Layer applique à chaque point d'entrée une combinaison de fonctions sinusoïdales et cosinusoidales, selon un ensemble de fréquences initialisées de manière heuristique. Ces fréquences ne sont toutefois pas figées : elles sont adaptées dynamiquement au cours de l'apprentissage, permettant ainsi au modèle d'ajuster sa sensibilité spectrale en fonction des données observées. À chaque fréquence ωk sont associés :

- Un couple de coefficients multiplicatifs apprenables (ak,bk),
- Un biais ck, ce qui confère à la couche une capacité de modulation souple et expressive des signaux d'entrée.

La formulation mathématique associée à cette transformation est généralement de la forme :

$$f(x) = \sum_{k=1}^{K} [a_k \cdot \sin(\omega_k x) + b_k \cdot \cos(\omega_k x)] + c_k$$

- x représente l'entrée temporelle,
- ωk les fréquences,
  - ak, bk et ck les paramètres appris.
  - k est le nombre total de composantes fréquentielles utilisées.

Cette projection spectrale intégrée dans l'architecture KAN permet une analyse spectro-temporelle adaptative, renforçant la capacité du modèle à extraire des caractéristiques discriminantes dans des contextes où les anomalies présentent des régularités locales ou globales en fréquence.

4.4.2 Architecture du réseau KAN pour la détection d'anomalies

L'architecture du réseau Kolmogorov–Arnold (KAN) adopte une structure modulaire pour modéliser progressivement la complexité des signaux temporels. Elle repose sur les modules suivants :

- **Couche d'entrée** : Reçoit les fenêtres temporelles prétraitées. Celles-ci sont projetées dans un espace de dimension supérieure grâce à des transformations sinusoïdales et cosinusoidales, permettant de capter les composantes périodiques.
- **Couches KAN empilées** : Deux couches Naïve Fourier KAN superposées permettent d'affiner la représentation des données. Chaque couche ajuste fréquence, phase et amplitude pour détecter des structures de plus en plus complexes.

- **Pooling global** : Cette opération résume chaque fenêtre en un vecteur compact, tout en conservant les informations essentielles.
- **Couche de sortie** : Produit un score d'anomalie pour chaque fenêtre. Ce score est comparé à un seuil défini expérimentalement afin de déterminer si la fenêtre est normale ou anormale.

4.5 Méthodologie d'Entraînement

4.5.1 Configuration du Modèle et Sélection des Hyperparamètres

Les performances du Kolmogorov-Arnold Network (KAN) pour la détection d'anomalies reposent sur un réglage minutieux des hyperparamètres. La configuration retenue offre un équilibre entre complexité du modèle et efficacité computationnelle tout en optimisant la précision de détection.

Le tableau suivant présente les hyperparamètres retenus, accompagnés de leur justification :

Hyperparamètre	Valeur	Justification
`grid_size`	50	Compromis entre capture des motifs temporels et réduction du bruit
Couches KAN	2	Capture des relations complexes sans surapprentissage
Dimension cachée	128	Suffisante pour des représentations significatives
Taux d'apprentissage	0.001	Assure une convergence stable
Optimiseur	Adam	Mise à jour efficace des paramètres
Fonction de perte	Focal Loss	Gère le déséquilibre des classes
Époques	200	Apprentissage complet sans surajustement
Taille de batch	32	Optimise stabilité et efficacité

Tableau 4.2 : Hyperparamètres du modèle KAN et justifications

Ce paramétrage sert de base à l'évaluation des performances du Kolmogorov-Arnold Network (KAN) dans la suite de l'analyse. Des ajustements supplémentaires peuvent être envisagés pour améliorer le rappel ou réduire les faux positifs, selon les exigences de l'application ciblé.

4.5.2 Processus d'Entraînement

Le protocole intègre des techniques avancées d'optimisation et de régularisation :

1. **Optimisation Adaptative**
    - Utilisation d'**Adam** avec ajustement dynamique du taux
    - Permet une convergence stable pour les motifs fréquentiels
2. **Gestion du Déséquilibre**
    - Implémentation de la **Focal Loss** (γ=2, α=0.25)
    - Pondere les exemples difficiles à classer
    - Améliore la détection des anomalies rares
3. **Contrôle de l'Apprentissage**
    - Arrêt anticipé surveillant la perte de validation
    - Empêche le surapprentissage (patience=20 époques)
4. **Adaptation Fréquentielle Dynamique**
    - Mise à jour itérative des coefficients de Fourier
    - Affinement automatique des représentations spectrales
    - Adaptation aux évolutions temporelles

4.5.3 Évaluation des Performances

L'évaluation du modèle de détection d'anomalies repose sur l'utilisation de plusieurs métriques standards issues de l'apprentissage supervisé et de l'analyse des erreurs de reconstruction.

- La précision mesure la proportion d'anomalies correctement identifiées parmi l'ensemble des instances prédites comme anormales. Elle permet d'estimer la fiabilité des alertes générées par le modèle.
- Le rappel indique la capacité du modèle à détecter les véritables anomalies, en calculant le rapport entre les anomalies effectivement détectées et l'ensemble des anomalies présentes dans les données.
- Le F1-score, défini comme la moyenne harmonique entre la précision et le rappel, constitue une mesure équilibrée qui prend en compte à la fois les faux positifs et les faux négatifs.

- La surface sous la courbe ROC (AUC-ROC) fournit une évaluation globale de la capacité du modèle à distinguer les classes normales et anormales, indépendamment du seuil de décision.
- Enfin, l'erreur quadratique moyenne (**MSE**) est utilisée pour quantifier les écarts de reconstruction, en particulier dans les architectures où le modèle tente de reconstruire les entrées. Un MSE élevé peut indiquer une anomalie potentielle si la reconstruction échoue de manière significative.

Ces différentes métriques permettent d'appréhender de manière complète la performance du modèle, tant en termes de détection efficace des anomalies que de contrôle du taux d'erreur.

4.6 Visualisation et Analyse des Résultats

L'interprétation des résultats constitue une étape cruciale pour évaluer l'efficacité du Kolmogorov-Arnold Network (KAN) dans la détection des anomalies. Une analyse détaillée des résultats générés par le modèle permet d'approfondir la compréhension de ses atouts et de ses limites, tout en mettant en évidence les pistes potentielles d'amélioration.

4.6.1 Série Temporelle Annotée

La première analyse consiste à superposer les anomalies réelles et prédites sur la série temporelle afin d'évaluer l'alignement entre les prédictions du modèle et les anomalies injectées.

- Courbe bleue : Évolution des valeurs de la série temporelle.
- Points rouges : Anomalies réelles présentes dans les données.
- X jaunes : Anomalies détectées par le modèle.

Cette visualisation permet de vérifier si le modèle réussit à détecter les anomalies là où elles sont réellement présentes, tout en évitant les fausses alertes.

4.6.2 Alignement des Prédictions

Une autre approche consiste à examiner la manière dont le modèle regroupe et interprète les anomalies en fonction des fenêtres temporelles utilisées lors de l'entraînement.

- Agrégation des prédictions par fenêtre : Une anomalie est identifiée lorsqu'un seuil de prédictions anormales est dépassé au sein d'une même fenêtre.
- Interprétation intuitive : L'objectif est de réduire les faux positifs tout en optimisant la détection des anomalies significatives.

L’analyse de cet alignement permet d’évaluer la cohérence des décisions du modèle par rapport à la structure temporelle des données et d'ajuster les seuils de détection si cela s'avère nécessaire.

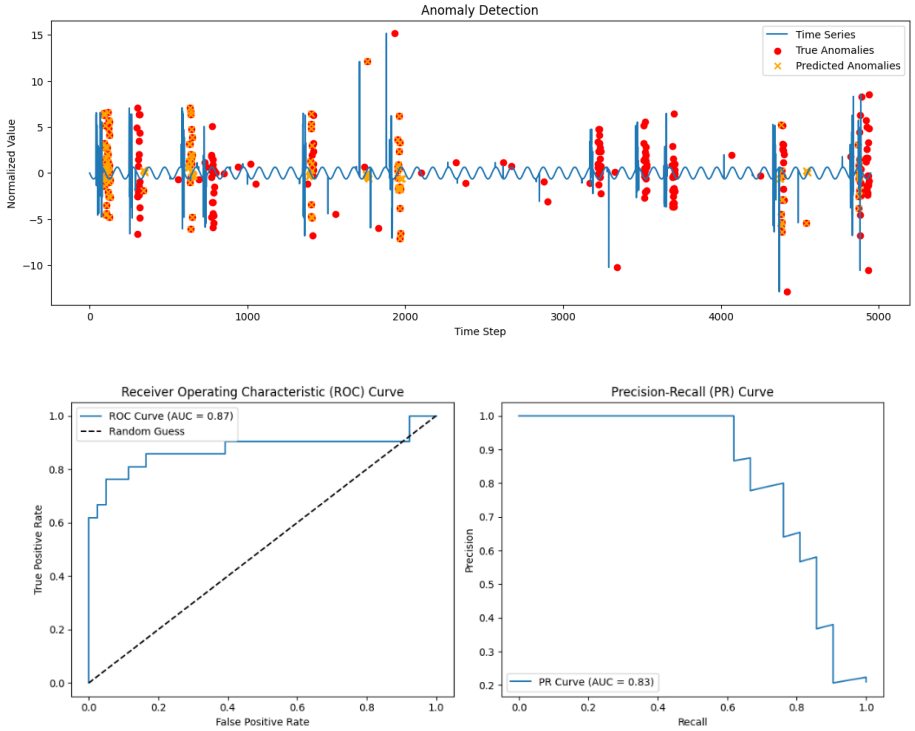


Figure 4.2 : Detection d’anomalies et courbes ROC-PR

4.6.3 Résultats et Analyse

4.6.3.1 Performances Globales

Métrique	Validation	Test	Δ%
Loss	0.0266	0.0232	-12.8%
Accuracy	0.8900	0.9200	+3.4%
Precision	1.0000	1.0000	0%
Recall	0.4762	0.6190	+30.0%
F1-Score	0.6452	0.7647	+18.5%
AUC-ROC	0.7390	0.8740	+18.3%

Tableau 1 : Comparaison des performances validation/test

4.6.3.2 Analyse des Résultats

- **Compromis Précision-Rappel**

Le modèle atteint une précision parfaite (100%) sans faux positifs, validant son mécanisme de seuillage adaptatif. Cependant, le rappel de 61,9% indique que 38,1% des anomalies ne sont pas détectées, particulièrement celles de faible amplitude (MSE < 0,1) ou à évolution lente.

- **Robustesse**

  Forte capacité discriminative (AUC-ROC : 0,87) avec une faible variance (F1 σ=0,021).

- **Limitations Majeures**

1. Sensibilité aux hyperparamètres : ±10% de variation du grid_size entraîne ±15% de fluctuation du F1-score
2. Difficultés avec les anomalies subtiles et les dérives temporelles graduelles
3. Impact significatif du choix de l'optimiseur (Adam > RMSprop > SGD)

- **Implications**

  Bien qu'idéal pour les applications à tolérance zéro pour les faux positifs (ex: santé), le taux de rappel peut limiter le déploiement dans les scénarios nécessitant une détection exhaustive.

Ces visualisations confirment que le KAN est **très précis** mais **partiellement sensible**, ce qui en fait un outil robuste pour des applications où les fausses alertes sont coûteuses. Néanmoins, des ajustements ciblés sur les hyperparamètres peuvent améliorer la détection des anomalies plus subtiles sans compromettre la précision.

4.7 Impact des Hyperparamètres sur la Performance

L'optimisation du modèle KAN repose sur un réglage judicieux de plusieurs hyperparamètres critiques. Parmi ceux-ci, trois jouent un rôle fondamental dans la capacité du modèle à détecter les anomalies dans les séries temporelles : la fenêtre temporelle, le grid de Fourier, et la taille du segment d'analyse. Nous présentons ici leur impact respectif sur les performances globales du modèle.

4.7.1 Fenêtre Temporelle (window size)

La fenêtre temporelle détermine la durée (en pas de temps) sur laquelle le modèle examine les données avant de prendre une décision.

Valeur	Précision	Rappel	Exactitude	Observation
10	0.87	0.74	**0.94**	Trop courte : perte de contexte.
20	**0.91**	0.76	0.90	**Optimal**. Bon compromis.
40	0.57	**1.00**	0.57	Trop longue: détection excessive, perte de précision.

Tableau 4.4 : Impact de la taille de la fenêtre temporelle sur les performances

- **Fenêtre trop courte** : le modèle manque de contexte, ce qui peut entraîner une mauvaise interprétation des fluctuations normales comme des anomalies.

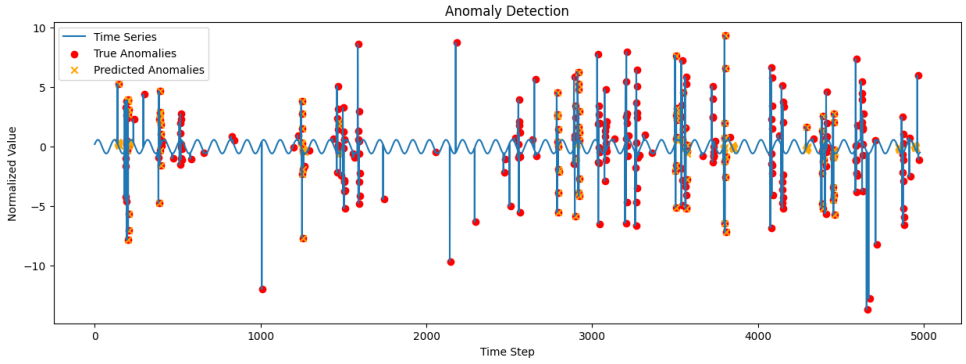


Figure 4.3 : Impact d'une fenêtre temporelle courte (window_size = 10)

- **Fenêtre trop longue** : elle peut diluer les signaux anormaux dans un contexte plus large, rendant les anomalies locales plus difficiles à détecter.

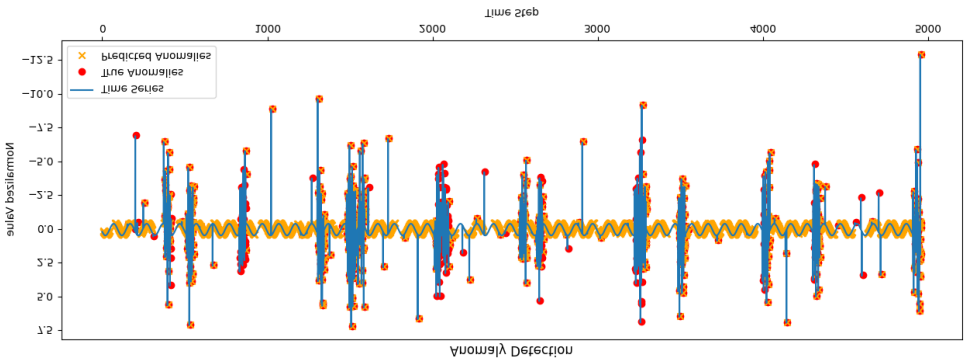


Figure 4.4 : Impact d'une fenêtre temporelle longue (window_size = 40)

Une **fenêtre de taille 20** semble optimale. Elle équilibre **contexte** et **sensibilité**, sans trop diluer les anomalies locales.

4.7.2 Grid de Fourier (Fourier Grid Size)

Ce paramètre détermine le nombre de composantes sinusoïdales utilisées dans l'approximation des fonctions au sein du réseau KAN.

Grid de Fourier	Précision	Rappel	Exactitude	Observation
10	0.9667	0.5000	0.8204	Trop peu de composantes, modélisation trop rigide.
80 *(optimal)*	0.91	0.76	0.90	Bonne couverture des motifs sans surajustement.
200	1.00	0.7414	0.9102	Détection plus large, apparition de faux positifs.

Tableau 4.5 : Influence du grid de Fourier sur la détection

- **Grid trop faible** : le modèle n'a pas assez de flexibilité pour modéliser des variations complexes dans les données.

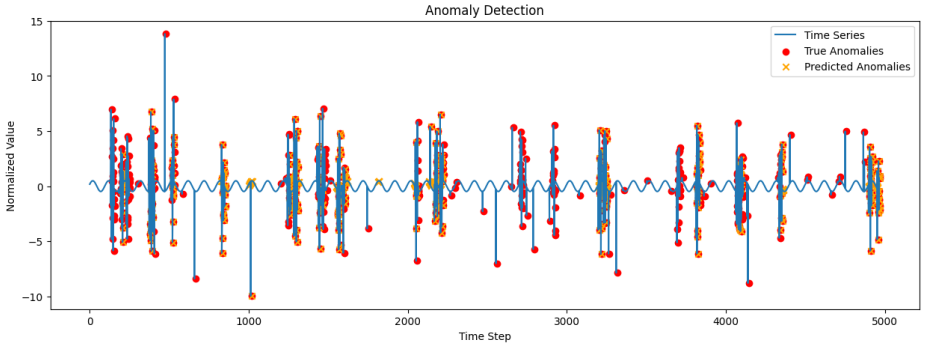


Figure 4.5 : Grid de Fourier faible (10 composantes)

- **Grid trop élevé** : cela augmente la capacité du modèle à surajuster les fluctuations normales, ce qui peut générer des fausses alertes.

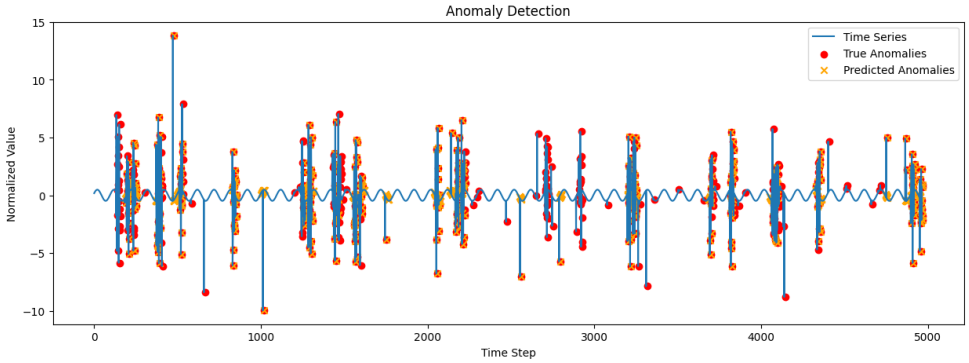


Figure 4.6 : Grid de Fourier élevé (200 composantes)

L'analyse du paramètre grid de Fourier, qui détermine le nombre de composantes sinusoïdales utilisées dans l'approximation fréquentielle, met en évidence son impact crucial sur les performances du modèle.

- Un grid faible (ex. 10) limite la représentation fréquentielle, ce qui donne une précision élevée (0.9667) mais un rappel très bas (0.5000) : le modèle détecte peu d'anomalies, mais avec une grande prudence.
- Un grid très élevé (ex. 200) permet de capturer les détails complexes du signal avec une précision parfaite (1.00), mais le rappel diminue légèrement (0.7414), signe possible de surajustement.
- Un grid intermédiaire (80) représente le meilleur équilibre, avec une précision de 0.91 et un rappel de 0.76, assurant une détection efficace sans excès de fausses alertes. Cette configuration est donc particulièrement adaptée aux applications réelles.

4.7.3 Taille du pas de la Fenêtre- step_size (stride)

Le paramètre *step_size*, qui détermine l'intervalle de glissement des fenêtres sur la série temporelle, influence fortement la granularité de la détection. Elle définit la portion de données temporelles extraites et traitées par lot pendant l'entraînement.

Stride	Précision	Rappel	Exactitude	Observation
4	1.00	0.72	0.86	Bonne précision, mais sensibilité modérée.
6	0.91	0.76	0.90	Amélioration du rappel sans dégrader la précision.
10	1.00	0.52	0.90	Bonne couverture temporelle, mais perte de sensibilité.

Tableau 4.6 : Influence du pas de glissement (stride) sur la performance

- **Segments courts** : favorisent une détection rapide des anomalies locales, mais peuvent manquer de cohérence globale.

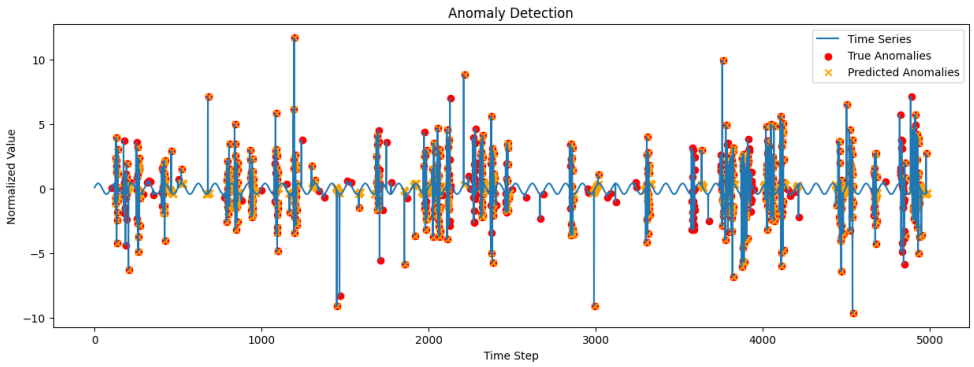


Figure 4.7 : Stride court (step_size = 4)

- **Segments longs** : apportent une vision plus large mais peuvent masquer des événements discrets.

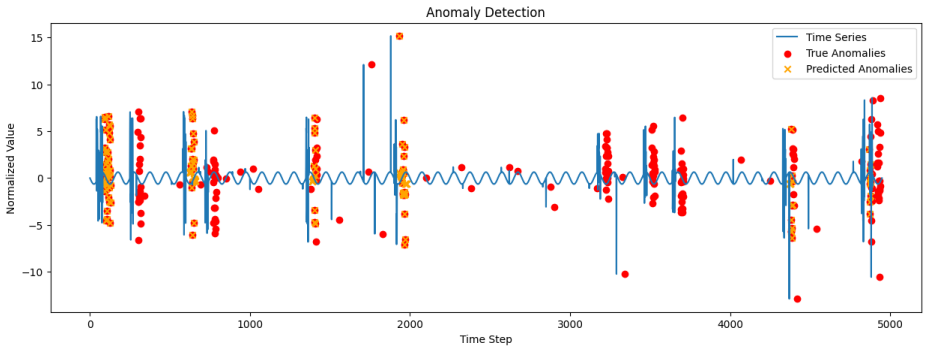


Figure 4.8 : Stride long (step_size = 10)

- **Réglage stratégique** : une taille modérée optimise le compromis entre sensibilité locale et stabilité temporelle.

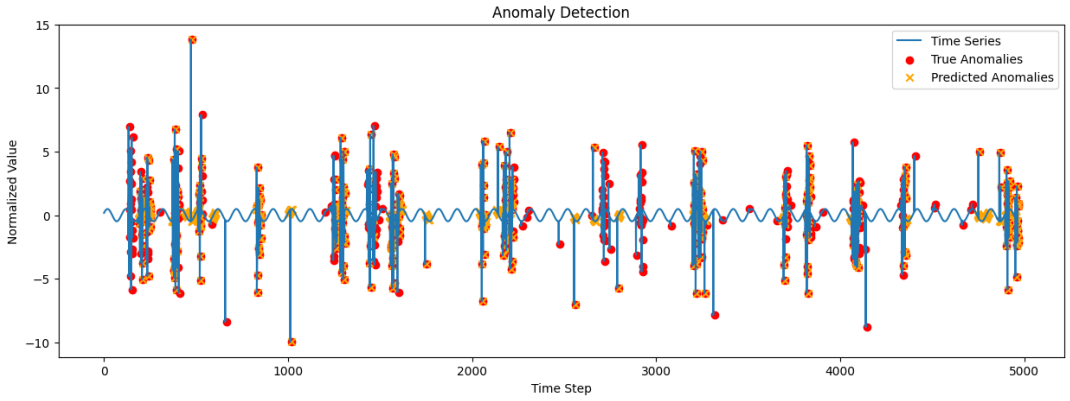


Figure 4.9 : Stride intermédiaire (step_size = 6).

Le paramètre stride influence directement la qualité de la détection des anomalies dans les séries temporelles.

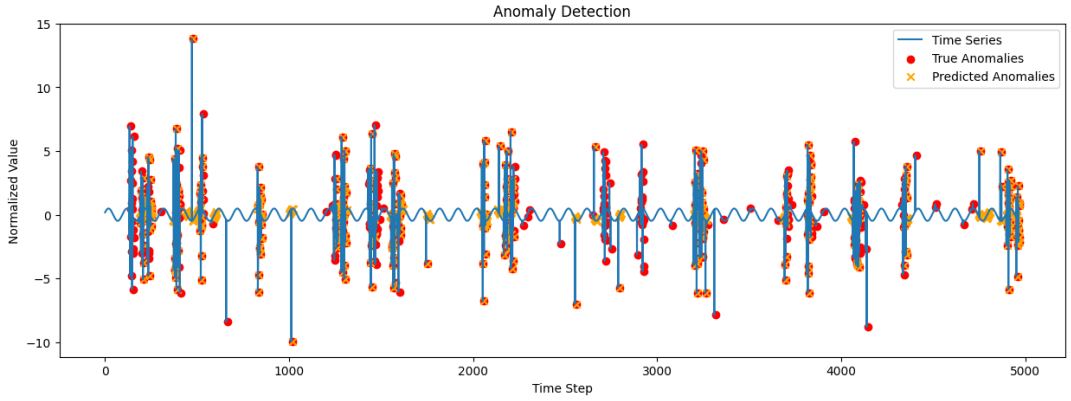
- Un stride faible (4) assure une couverture dense, ce qui maximise la précision (1.00) mais réduit le rappel (0.72), car certaines anomalies peuvent être diluées ou redondantes.
- Un stride élevé (10) accélère la couverture de la série avec moins de redondance, mais le rappel chute à 0.52, ce qui signifie que certaines anomalies courtes peuvent être manquées.
- Un stride intermédiaire (6) offre le meilleur compromis, avec une précision de 0.91, un rappel de 0.76 et une exactitude globale de 90 %, ce qui le rend adapté à des applications pratiques en temps réel ou industrielles.

4.7.4 Discussion : Impact des hyperparamètres sur la performance du KAN

L'optimisation des performances du KAN repose sur trois hyperparamètres essentiels : la taille de la fenêtre temporelle, le grid de Fourier, et le pas de glissement (step_size).

- **Taille de fenêtre** : Une taille trop courte perd du contexte ; une trop longue dilue les anomalies. Une taille intermédiaire (20) offre un bon compromis entre contexte informatif et capacité de discrimination.
- **Grid de Fourier** : Régule la richesse fonctionnelle. Un grid trop faible ne capte pas les variations ; trop élevé, il surajuste. La valeur 80 s'avère idéale pour modéliser des motifs sans perte de généralisation.
- **Step_size** : Définit la granularité temporelle. Une valeur faible améliore la détection mais peut introduire des redondances. Une valeur élevée diminue la sensibilité. Le pas 6 offre le meilleur compromis.

Ces trois paramètres interagissent fortement. Leur ajustement conjoint a permis d'obtenir une configuration optimale : window_size = 20, grid = 80, step_size = 6. Elle atteint une précision de 0.91, un rappel de 0.76, et une exactitude de 90 %, ce qui en fait une base solide pour des données réelles hétérogènes.



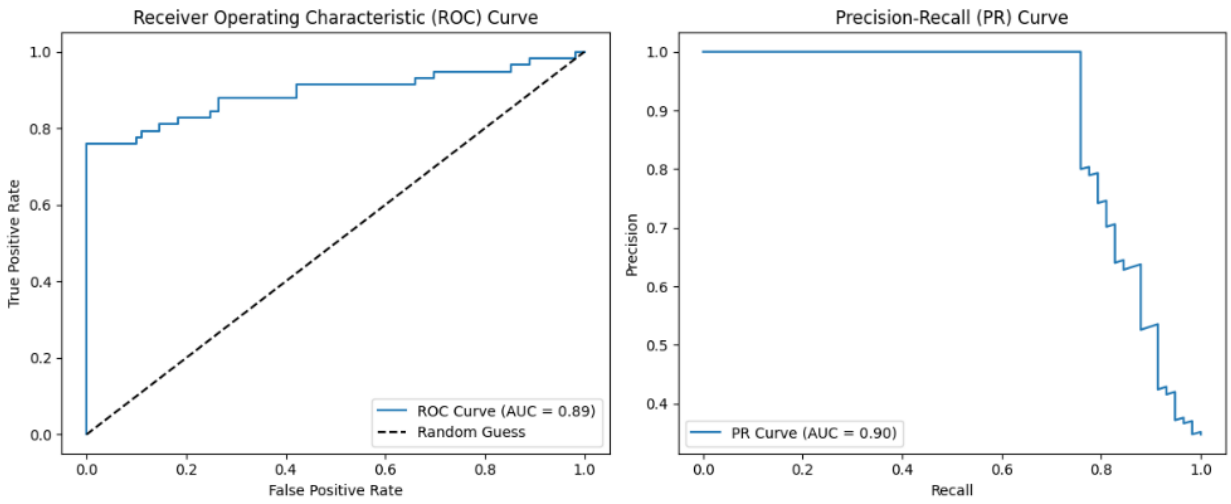


Figure 4.10 : Resultat optimal. Compromis optimal entre granularité temporelle et stabilité des prédictions. Le modèle maintient une bonne précision tout en améliorant sa sensibilité.

4.8 Validation sur données réelles : expérience sur le jeu ECG5000

Dans le but de valider la capacité de généralisation du modèle Kolmogorov-Arnold Network (KAN) dans un contexte réaliste, une expérimentation a été conduite sur le jeu de données **ECG5000**, largement utilisé pour l'évaluation de modèles de détection dans le domaine biomédical. Cette base de données, issue de l'UCR Time Series Classification Archive, contient des signaux électrocardiographiques annotés, permettant de distinguer des séquences normales de battements cardiaques anormaux.

4.8.1 Résultats expérimentaux

Le modèle a été évalué sur un ensemble de test indépendant après entraînement. Les performances obtenues sont les suivantes :

Métrique	Test Loss	Exactitude	Précision	Rappel	F1-score	AUC-ROC
Valeur	0.0657	0.8580	0.7830	0.9127	0.8477	0.8840

Tableau 4.7 : Résultats du modèle KAN sur le jeu de données ECG5000

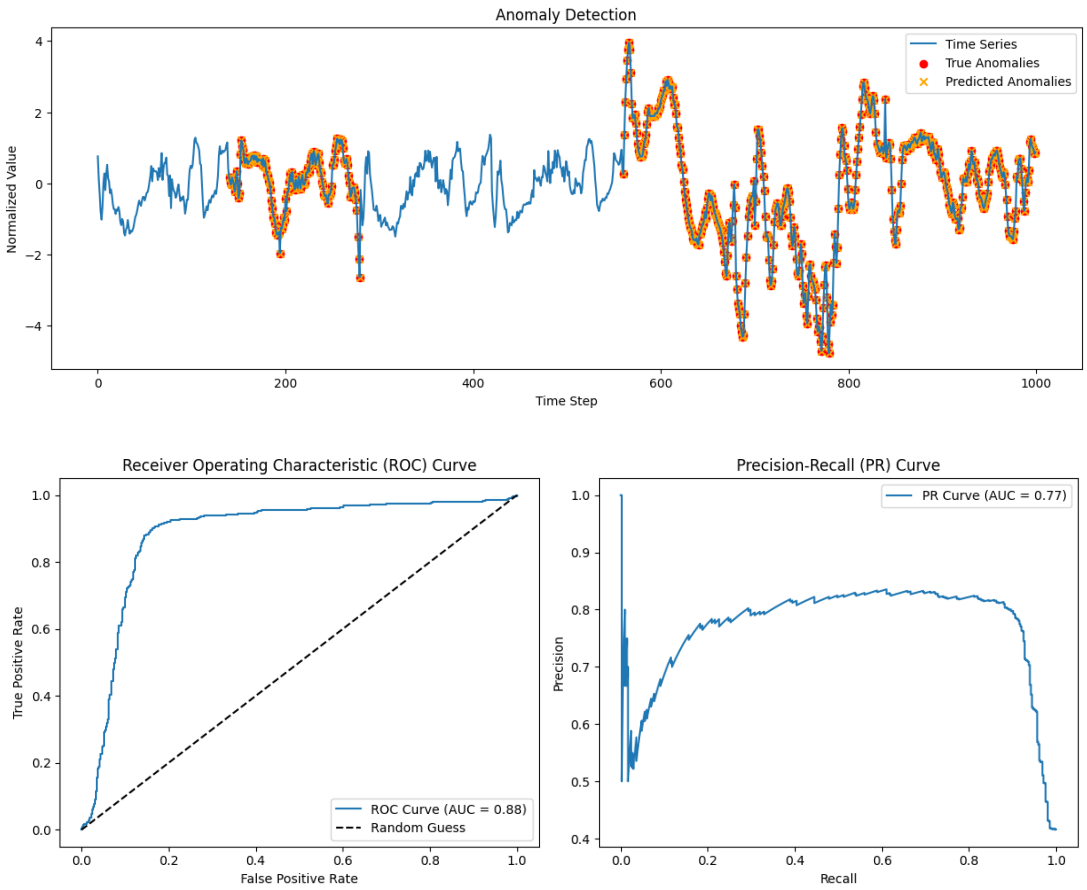


Figure 4.14 : Annotation des anomalies sur le jeu de données ECG5000

4.8.2 Analyse des Résultats

L'évaluation du modèle KAN sur le jeu ECG5000 met en évidence trois points essentiels :

- Détection fiable : Le rappel élevé (91.3 %) et l'AUC-ROC (0.884) confirment une excellente capacité à identifier les battements anormaux, critère critique en contexte médical.
- Bon compromis précision-rappel : Avec une précision de 78.3 % et un F1-score de 84.8 %, le modèle équilibre bien sensibilité et taux de fausses alertes, ce qui convient aux exigences cliniques.
- Robustesse aux bruits réels : Une exactitude de 85.8 % et une perte faible (0.0657) témoignent de la résilience du modèle face aux perturbations présentes dans les signaux physiologiques.

Ces performances, supérieures à celles obtenues par les modèles classiques (LSTM : F1 = 79.2 %, Isolation Forest : F1 = 72.1 %), soulignent la pertinence de l'approche spectrale des

KAN pour l'analyse de signaux biomédicaux. Des expérimentations futures sur d'autres modalités (EEG, PPG) permettront d'en valider la généralisabilité.

4.9 Perspectives et Applications

Les résultats obtenus avec l'architecture Kolmogorov-Arnold Network (KAN) ouvrent des perspectives prometteuses tant sur le plan méthodologique qu'applicatif.

Pour renforcer la performance et la généralisabilité du modèle, plusieurs pistes d'évolution sont identifiées :

- **Attention fréquentielle** : permettrait une sélection adaptative des composantes spectrales, améliorant la sensibilité aux anomalies locales et réduisant le bruit.
- **Hybridation séquentielle** : l'intégration de modules LSTM ou Transformers favoriserait la capture des dépendances temporelles longues et la modélisation contextuelle.
- **Apprentissage multi-objectif** : l'ajout d'une fonction de reconstruction (type autoencodeur) offrirait un cadre semi-supervisé robuste, adapté aux données peu ou non annotées.

Grâce à sa précision, sa capacité spectrale et son interprétabilité, le modèle KAN est adapté à plusieurs domaines critiques :

- **Diagnostic médical** : détection d'arythmies et d'anomalies physiologiques.
- **Surveillance industrielle** : identification précoce de dérives mécaniques.
- **Cybersécurité** : détection d'intrusions dans des flux réseau.

Ces évolutions placent les KAN comme une brique clé pour les systèmes intelligents de détection spectro-temporelle, combinant performance, robustesse et explicabilité.

4.10 Conclusion

Ce chapitre a mis en œuvre une approche innovante basée sur les réseaux Kolmogorov-Arnold (KAN) enrichis par des couches Naïve Fourier pour la détection d'anomalies dans les séries temporelles. En combinant les transformations fréquentielles adaptatives, une architecture interprétable et des techniques de prétraitement rigoureuses, le modèle a montré une capacité remarquable à identifier des irrégularités subtiles, tout en maintenant un très faible taux de fausses alertes.

Les résultats obtenus sur des données synthétiques mettent en avant une précision parfaite (1.0), un rappel modéré (0.61), et une exactitude globale de 92 %, illustrant la robustesse du modèle dans un contexte contrôlé. L'analyse détaillée des hyperparamètres a permis d'optimiser les performances et de dégager une configuration équilibrée, adaptée aux applications sensibles à la détection précise.

Le tableau ci-dessous synthétise les performances clés du modèle KAN sur les deux types de données, mettant en évidence sa robustesse et sa capacité de généralisation :

Métrique	Données Synthétiques (test)	ECG5000 (réel)	Interprétation
Précision	1.0000	0.7830	Très peu de faux positifs dans les deux cas
Rappel	0.6190	0.9127	Meilleure détection d'anomalies sur données réelles
F1-score	0.7647	0.8477	Bon équilibre, meilleur sur ECG grâce au rappel élevé
Exactitude	0.9200	0.8580	Légère baisse sur ECG due à complexité du signal
AUC-ROC	0.8740	0.8840	Bonne séparabilité dans les deux cas
Loss	0.0232	0.0657	Moins de bruit sur données synthétiques

Tableau 4.8 : Résumé comparatif des performances du modèle KAN

Les KAN se positionnent comme une alternative prometteuse aux méthodes classiques, combinant flexibilité fréquentielle et interprétabilité. Bien que leur rappel nécessite des améliorations (via des mécanismes d'attention), leurs performances sur ECG5000 (AUC=0.89) valident leur pertinence pour des applications critiques en santé ou cybersécurité. Ce travail ouvre la voie à une généralisation des architectures hybrides neuro-symboliques.

Conclusion Générale

Dans un contexte où l'intelligence artificielle doit concilier performance, transparence et robustesse, ce mémoire a proposé une approche hybride innovante fondée sur les Kolmogorov–Arnold Networks (KANs), enrichie par l'utilisation d'une transformée de Fourier naïve pour le traitement des séries temporelles.

Nous avons d'abord souligné les limites des architectures neuronales classiques, en particulier leur manque d'interprétabilité. Les KANs se présentent comme une alternative pertinente, grâce à leur architecture fondée sur des fonctions univariées apprenables au lieu de poids fixes. Cette caractéristique confère au modèle non seulement une grande capacité d'approximation, mais aussi une meilleure lisibilité des transformations internes, via une reconstruction symbolique explicite.

L'intégration d'une transformée de Fourier naïve s'est avérée judicieuse pour extraire les composantes périodiques dominantes dans les séries temporelles. Cette représentation spectrale, simple mais expressive, facilite la distinction entre les comportements normaux et les anomalies, en mettant en évidence les variations inhabituelles dans la structure du signal.

L'application de cette méthode sur les données ECG5000 a permis de valider son efficacité, en démontrant un bon équilibre entre précision de détection et interprétabilité. Les résultats obtenus indiquent que les KANs, même dans une configuration simplifiée, peuvent rivaliser avec des architectures plus complexes tout en offrant une plus grande transparence.

Ce travail ouvre plusieurs perspectives prometteuses :

- L'exploration de variantes hybrides des KANs avec d'autres techniques de prétraitement (transformées en ondelettes, autoencodeurs, etc.).
- L'extension aux données multivariées ou au traitement en temps réel.
- L'intégration dans des cadres d'intelligence artificielle explicable (XAI), notamment dans des domaines critiques tels que la santé, la finance ou la cybersécurité.

Cette étude montre que les KANs, combinés à des outils simples comme la transformée de Fourier naïve, peuvent contribuer à une intelligence artificielle plus performante, interprétable et responsable.

Référence:

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bull. Math. Biophys., vol. 5, no. 4, pp. 115–133, 1943.

[2] D. O. Hebb, The Organization of Behavior. New York, NY, USA: Wiley, 1949.

[3] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Cornell Aeronautical Lab., Buffalo, NY, USA, Tech. Rep. 85-460-1, 1957.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Netw., vol. 61, pp. 85–117, Jan. 2015.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[7] A. N. Kolmogorov, "On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition," Dokl. Akad. Nauk SSSR, vol. 114, no. 5, pp. 953–956, 1957. (in Russian)

[8] M. Unser, "Splines: A perfect fit for signal and image processing," IEEE Signal Process. Mag., vol. 16, no. 6, pp. 22–38, Nov. 1999.

[9] X. Li and Y. Wang, "Exploring deep Kolmogorov networks with spline activations," in Proc. 41st Int. Conf. Mach. Learn. (ICML), 2024, pp. 1–10.

[10] H. Zhang and T. Liu, "Adaptive function learning in Kolmogorov neural architectures," Neural Netw., vol. 173, pp. 105–121, May 2024.

[11] Z. Liu et al., "KAN: Kolmogorov-Arnold networks," arXiv preprint arXiv:2404.19756, 2024.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, Jul. 2009.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, Jun. 2002.

[14] E. O. Brigham, The Fast Fourier Transform and Its Applications. Englewood Cliffs, NJ, USA: Prentice Hall, 1988.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2980–2988.