

Mémoire de fin d'études
pour l'obtention du diplôme d'Ingénieur d'État en Informatique
Option : Systèmes Informatiques (SQ)

Optimisation des architectures en deep learning en utilisant les techniques de plongement de graphes

Réalisé par : M. KACEMI Souhib
M. TAIBI Mohamed Kamel Eddine *Encadré par :* M. AIT ALI YAHIA Yacine (ESI)
Mme. AMROUCHE Karima (ESI)

Soutenu le 20 Septembre 2023, Devant le jury composé de :

Mme. Fatima BENBOUZID-SI TAYEB : ESI - Président
M. Hachemi Nabil DELLYS : ESI - Rapporteur
Mme. Sara GHORAB : ESI - Examinateur

Promotion : 2022/2023

Dédicace

“

À mes chers parents,

À mes chers frères et soeurs et leurs enfants,

À tous mes amis,

À mes professeurs,

Merci.

”

- Souhib

Remerciements

Nous tenons à exprimer nos profondes gratitude pour tous ceux qui ont contribué à la réalisation de ce mémoire de master.

Nos sincères remerciements vont à nos directeurs de mémoire qui nous ont offert un encadrement de qualité, un soutien inestimable et des conseils judicieux tout au long de ce parcours académique.

Nous tenons également à remercier tous les enseignants qui nous ont prodigué leur savoir et leurs compétences pour que nous puissions réussir dans nos études.

Nous exprimons également nos profonds remerciements à **M. Ali TFAILY** et **M.Souhib KACEMI** pour leur support et leurs orientations.

Nos remerciements s'adressent également à nos familles et nos amis pour leur soutien, leur encouragement et leur amour indéfectibles.

Enfin, nous sommes reconnaissants envers toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce travail de recherche. Merci infiniment pour votre précieuse aide et vos précieux conseils tout au long de cette aventure.

Résumé

La maintenance prédictive est cruciale aujourd’hui pour anticiper les pannes avant qu’elles ne surviennent, permettant ainsi de réduire les coûts et le temps de maintenance. Les moteurs électriques, largement utilisés dans l’industrie, les transports et les usines, sont sujets à de nombreux défauts électriques et mécaniques, rendant leur maintenance coûteuse mais indispensable.

Ce mémoire explore l’application du machine learning pour la maintenance prédictive des moteurs électriques. La qualité des données est un élément clé du machine learning. Pour enrichir notre jeu de données sur différents moteurs présents sur le marché, nous utilisons l’intelligence artificielle générative pour créer des données similaires aux données réelles, sous forme de séries temporelles. Des techniques avancées telles que les auto-encodeurs variationnels (VAE), les réseaux adverses génératifs (GAN) et les grands modèles de langage (LLM) sont employées pour générer ces données.

Le travail réalisé a permis de produire des séries temporelles très proches des données réelles, lesquelles sont utilisées par un modèle de classification pour prédire de manière fiable si un moteur va tomber en panne ou non. Les résultats obtenus démontrent que notre solution est capable de générer des données de haute qualité et de prédire efficacement les pannes des moteurs électriques, offrant ainsi une approche prometteuse pour la maintenance prédictive dans divers secteurs industriels.

Mots clés : Apprentissage profond, Apprentissage automatique, Réseaux neuronaux profonds, Maintenance prédictive, Séries temporelles, Réseaux adverses génératifs (GAN), Auto-encodeurs variationnels (VAE), Grands modèles de langage (LLM)

Abstract

Predictive maintenance is crucial today for anticipating failures before they occur, thereby reducing costs and maintenance time. Electric motors, widely used in industry, transportation, and factories, are subject to numerous electrical and mechanical faults, making their maintenance costly but indispensable.

This thesis explores the application of machine learning for the predictive maintenance of electric motors. Data quality is a key element of machine learning. To enrich our dataset on various motors available on the market, we use generative artificial intelligence to create data similar to real data, in the form of time series. Advanced techniques such as variational autoencoders (VAE), generative adversarial networks (GAN), and large language models (LLM) are employed to generate these data.

The work carried out has produced time series very close to real data, which are used by a classification model to reliably predict whether a motor will fail or not. The results obtained demonstrate that our solution is capable of generating high-quality data and effectively predicting electric motor failures, thus offering a promising approach for predictive maintenance in various industrial sectors.

Keywords : Deep learning, Machine learning, Deep neural networks, Predictive maintenance, Time series, Generative adversarial networks (GAN), Variational autoencoders (VAE), Large language models (LLM)

Table des matières

Dédicace	I
Remerciements	II
Résumé	III
Abstract	IV
Introduction générale	1
I Etude bibliographique	3
1 Apprentissage profond	4
1.1 Introduction	4
1.2 Réseau de neurones artificiels	5
1.3 Connexions et poids	5
1.4 Fonction d'activation	6
1.5 Couches dans un réseau de neurones	8
1.6 Types de réseaux de neurones	10
1.6.1 Réseaux feedforward	10
1.6.2 Réseaux de neurones récurrents (RNN)	10
1.6.3 Réseaux de neurones convolutifs (CNN)	13
1.6.4 Réseaux résiduels (ResNet)	15
1.7 Le processus d'apprentissage	16
1.7.1 Descente de gradient	16
1.7.2 Propagation de l'erreur	17
1.7.3 Hyperparameters	17
1.8 Types d'apprentissage	18
1.8.1 Apprentissage supervisé	18
1.8.2 Apprentissage non-supervisé	19
1.8.3 Apprentissage semi-supervisé	19
1.8.4 Apprentissage par renforcement	20
1.9 Catégories de données	20
1.10 Défis de l'apprentissage profond	20
1.11 Conclusion	21
2 Intelligence Artificielle Générative (GenAI)	23

Table des matières

2.1	Introduction	23
2.2	Les Auto-Encodeurs Variationnels (VAE)	23
2.2.1	Introduction	23
2.2.2	Fonctionnement	24
2.2.3	Applications des Auto-Encodeurs Variationnels (VAE)	25
2.3	Réseaux Antagonistes Génératifs (GANs)	27
2.3.1	Introduction aux Réseaux Antagonistes Génératifs	27
2.3.2	Fonctionnement des GANs	27
2.3.3	Applications des GANs	29
2.4	Diffusion models	31
2.4.1	Introduction	31
2.4.2	Fonctionnement des Modèles de Diffusion	31
2.4.3	Applications des Modèles de Diffusion	34
2.4.4	Conclusion	35
3	Maintenance prédictive des moteurs électriques	36
3.1	Introduction	36
3.2	Types de Maintenance	37
3.2.1	Maintenance Corrective	37
3.2.2	Maintenance Préventive	37
3.2.3	Maintenance Prédictive	37
3.3	Maintenance prédictive pour les moteurs électriques	38
3.3.1	Moteurs électriques	39
3.4	Traitement de Signal Pour les moteurs électriques	41
3.5	Données : Séries Temporelles	44
3.6	Conclusion	45
II	Contribution	46
4	Organisme d'accueil	47
4.1	Introduction	47
4.2	Présentation de l'organisme d'accueil	47
4.3	Services	48
4.4	L'organigramme de Schneider Electric	48
4.5	Conclusion	49
5	Conception	50
5.1	Introduction	50
5.2	Problématique et besoins	50
5.3	Notions	52
5.4	Données utilisées	53
5.5	Solution proposée	57
5.5.1	TimeGAN	57
5.6	Conclusion	59
6	Réalisation et tests	60

Table des matières

6.1	Introduction	60
6.2	Technologies utilisées	60
6.2.1	Python	60
6.3	Bibliothèque utilisées	61
6.3.1	NumPy	61
6.3.2	Pandas	61
6.3.3	Plotly	62
6.3.4	Pytorch	62
6.4	Outils utilisés	63
6.4.1	Google Colab	63
6.4.2	Google Drive	64
6.5	Environnement de développement	64
6.6	Tests et résultats	65
6.6.1	Méthodes D'évaluation	65
6.6.2	Tests et résultats	66
6.7	Conclusion	68
7	Conclusion et perspectives	69
7.1	Perspectives	69
7.2	Appréciation personnelle	69
	Bibliographie	73

Table des figures

1.1	Le fonctionnement d'un neurone artificiel [McCulloch et al. 1943].	7
1.2	Les fonctions d'activations couramment utilisées [Junxi Feng et al. 2019]. .	8
1.3	Schéma simple d'un réseau de neurones feedforward [Muniasamy et al. 2020].	10
1.4	Exemple d'un réseau de neurones récurrent [Kumaraswamy 2021].	11
1.5	Architecture d'un bloc LSTM [Fawaz et al. 2019a].	12
1.6	Le fonctionnement d'un réseau neuronal convolutif [Alharbi et al. 2021]. .	13
1.7	Exemple du fonctionnement d'une couche convulsive [Kimura et al. 2019].	14
1.8	Exemples de pooling maximal et pooling moyen [Hu et al. 2022].	15
1.9	Un bloc régulier (gauche) et un bloc résiduel (droite) [Dong et al. 2022]. .	15
1.10	Fonction de taux d'erreur [Amini et al. 2018].	17
2.1	Le fonctionnement d'un neurone artificiel [Kingma et al. 2012].	25
2.2	Un modèle de réseau générateur adversaire (GAN) [Jie Feng et al. 2020]. .	29
2.3	Quelques images générées par le modèle StyleGAN sur le site 'This Person Does Not Exist'[P. Wang 2019].	30
2.4	Chaîne de Markov du processus de diffusion [Ho et al. 2020].	31
2.5	Le processus d'entraînement d'un UNet pour prédire le bruit ajouté à l'image. [Nichol et al. 2021].	32
2.6	Illustration des Processus de Diffusion : Forward Process et Reverse Process.	33
2.7	Exemple d'images générée par DALL-E2. [Ramesh et al. 2022].	35
3.1	Stratégies de maintenance montrant les différentes étapes du processus de réparation avant et après la défaillance potentielle de l'équipement. [Mrozek et al. 2023].	38
3.2	Principaux Composants d'un moteur asynchrone triphasé (Machine à Induction).	40
3.3	Principaux Composants d'un moteur asynchrone triphasé (Machine à Induction).	42
3.4	Exemples de tracés de séries temporelles discrètes (à gauche) et continues (à droite) [Brophy et al. 2023]	44
5.1	Schneider Electric ATV930D30N4 AC Speed Drive, 3 HP.	51
5.2	Organization de data [Yoon et al. 2019].	54
5.3	Organization de data	54
5.4	Organization de data.	55
5.5	Organization de data [Yoon et al. 2019].	56
5.6	Organization de data [Yoon et al. 2019].	57
5.7	Architecure de TimeGAN[Yoon et al. 2019].	58
5.8	Architecure de TimeGAN[Yoon et al. 2019].	59

Table des figures

6.1	Python.	60
6.2	NumPy.	61
6.3	Pandas.	62
6.4	Plotly.	62
6.5	Pytorch.	63
6.6	Google Colab.	63
6.7	Google Drive.	64
6.8	Seq.	66
6.9	Seq.	67
6.10	Seq.	67
6.11	Seq.	68

Liste des tableaux

3.1	Comparaison entre les Types de Maintenance	38
6.1	Configuration du modèle et du jeu de données	66
6.2	Configuration du modèle et du jeu de données	67

Liste des algorithmes

1	Normalization Nominale	57
---	----------------------------------	----

Liste des sigles et acronymes

AI	<i>Artificial Intelligence</i>
GenAI	<i>Generative Artificial Intelligence</i>
ReLU	<i>Rectified Linear Unit</i>
ANN	Artificial Neural Network
DL	<i>Deep Learning</i>
ML	<i>Machine Learning</i>
GAN	<i>Generative Adversarial Networks</i>
LLM	<i>Large language models</i>
VAE	<i>Variational Autoencoders</i>
CNN	<i>Convolutional Neural Networks</i>
RNN	<i>Recurrent Neural Network</i>
LSTM	<i>Long Short-Term emory</i>
MDP	<i>Markov Decision process</i>
NLP	<i>Natural language Processing</i>
GD	<i>Gradient Descent</i>
MLP	<i>MultiLayer Perceptron</i>

Introduction générale

L'apprentissage profond, ou deep learning, est une branche de l'intelligence artificielle (IA) qui a transformé de nombreux secteurs et industries, en particulier ces dernières années. Grâce à ses capacités avancées, l'apprentissage profond a permis des avancées significatives dans des domaines tels que la reconnaissance d'image, la compréhension du langage naturel et la génération de données. En tant que composant essentiel de l'intelligence artificielle générative, l'apprentissage profond est utilisé pour créer divers types de données, y compris des textes, des images, des données tabulaires et des données séquentielles.

Les architectures d'intelligence artificielle générative, telles que les GANs (Generative Adversarial Networks), les LLMs (Large Language Models) et les autoencodeurs variationnels (VAE), jouent un rôle crucial dans la génération de données synthétiques. Ces modèles sont particulièrement efficaces pour augmenter les ensembles de données existants, ce qui est essentiel pour entraîner d'autres modèles de machine learning avec des ensembles de données plus diversifiés et représentatifs.

Un domaine d'application particulièrement intéressant est celui des moteurs électriques, qui sont largement utilisés aujourd'hui et jouent un rôle crucial dans divers secteurs de l'industrie, notamment le transport. Ces moteurs, provenant de multiples fabricants et marques, nécessitent une maintenance prédictive pour garantir leur bon fonctionnement et prolonger leur durée de vie. Cependant, la diversité des marques et des modèles de moteurs pose un défi en termes de collecte de données suffisantes et variées pour chaque type de moteur.

Dans ce contexte, l'IA générative peut être utilisée pour générer des données synthétiques qui couvrent un large éventail de moteurs électriques. En généralisant sur toutes les variétés de moteurs existants, l'IA générative permet d'augmenter les ensembles de données, ce qui est crucial pour entraîner des modèles de classification et de prédiction plus précis et robustes. Ces modèles peuvent ensuite être utilisés pour effectuer une maintenance prédictive efficace, réduisant ainsi les temps d'arrêt et les coûts de maintenance.

Ce travail se concentrera sur l'application de l'IA générative pour la génération de données de type séries temporelles. Nous viserons à généraliser ces données pour qu'elles représentent une large gamme de moteurs électriques. L'objectif final est de faciliter la maintenance prédictive de ces moteurs en utilisant des ensembles de données augmentés et diversifiés, permettant ainsi d'améliorer la fiabilité et l'efficacité des systèmes de maintenance.

Introduction générale

Dans cet article, nous présentons les méthodes d'intelligence artificielle générative dans le contexte de l'augmentation de dataset pour la maintenance prédictive. Nous utilisons notamment les réseaux adversatifs génératifs (GAN) et les modèles de diffusion. Les GAN, qui sont généralement composés d'un générateur et d'un discriminateur, permettent de générer des données synthétiques où le générateur crée des données et le discriminateur évalue la qualité de ces données. Par ailleurs, nous abordons les modèles de diffusion qui génèrent des données à partir d'un bruit gaussien. Nous détaillons les différentes étapes nécessaires pour générer des séries temporelles et discutons des méthodes d'évaluation pour apprécier la qualité des données générées par ces modèles. En outre, nous appliquons des techniques de traitement du signal pour visualiser les données dans le domaine fréquentiel.

Nous commençons par une revue de la littérature sur l'apprentissage profond et les différentes architectures existantes. Nous y présentons également des modèles génératifs tels que les autoencodeurs variationnels (VAE), les GAN et les modèles de langage de grande taille (LLM). Le dernier chapitre de cette revue bibliographique est consacré aux bases de la maintenance prédictive, aux composants des moteurs électriques, ainsi qu'aux techniques de traitement du signal comme la transformation de Fourier rapide (FFT).

partie I

Etude bibliographique

Chapitre 1

Apprentissage profond

1.1 Introduction

L'apprentissage profond (*deep learning* en anglais) est une branche de l'intelligence artificielle (IA) qui s'intéresse à la résolution des problèmes intuitifs, c'est-a-dire des tâches qui sont faciles à réaliser par les humains mais difficiles à décrire formellement. Ce sont des problèmes qui semblent automatiques, comme la reconnaissance des mots parlés ou des visages dans les images. L'apprentissage profond permet aux ordinateurs d'apprendre des concepts complexes en rassemblant de l'expérience. Cela permet d'éviter la spécification formelle des connaissances dont l'ordinateur a besoin [GOODFELLOW et al. 2016].

L'apprentissage profond utilise des réseaux de neurones profonds pour résoudre ces problèmes. Ces réseaux sont des modèles computationnels qui imitent le fonctionnement du cerveau humain [MCCULLOCH et al. 1943, ROSENBLATT 1958]. Ils sont constitués de plusieurs couches de neurones artificiels cachées qui traitent les données d'entrée.

Il existe trois grandes catégories d'apprentissage automatique : *supervisé*, *non-supervisé* et *semi-supervisé*. Dans l'apprentissage supervisé, on utilise un ensemble de données étiquetées, tandis que dans l'apprentissage non-supervisé, on ne dispose pas d'un ensemble de données étiquetées. L'apprentissage semi-supervisé est une combinaison d'apprentissage supervisé et non-supervisé. Dans l'apprentissage semi-supervisé, un ensemble de données est étiqueté, mais la majorité des données sont non étiquetées [GOODFELLOW et al. 2016, BISHOP 2016].

Dans l'apprentissage profond, plusieurs types d'architecture existent, chacune adaptée à des tâches spécifiques. Parmi les plus courants, on trouve : les *réseaux de neurones convolutionnels* (CNN), les *réseaux de neurones récurrents* (RNN), les *réseaux de neurones générateurs adversaires* (GANs) et les *réseaux de neurones de transformation* (Transformer) [GOODFELLOW et al. 2016].

Dans ce chapitre, nous allons expliquer brièvement les différentes notions en relation avec l'apprentissage profond, telles que les couches du réseau, les fonctions d'activation, les types de réseaux, les connexions et les poids, le processus d'apprentissage et les types d'apprentissage.

1.2 Réseau de neurones artificiels

Un réseau de neurones artificiels (*Artificial Neural Network* en anglais) est un modèle de traitement de l'information construit de couches de neurones interconnectées qui traitent les données d'entrée en les transmettant à travers des poids de connexion qui peuvent être ajustés par un processus d'apprentissage [AGGARWAL 2018]. Ce réseau s'inspire du fonctionnement des neurones biologiques du cerveau.

Chaque neurone dans les couches cachées du réseau reçoit des signaux d'entrée à partir des neurones précédents, les somme, et les transmet aux neurones de la couche suivante à travers une fonction d'activation. Les réseaux de neurones peuvent avoir plusieurs couches cachées, qui permettent de modéliser des relations non linéaires complexes entre les données d'entrée et de sortie. Ces réseaux neuronaux peuvent compter jusqu'à 150 couches, d'où le nom "profond". [GOODFELLOW et al. 2016].

Les réseaux de neurones peuvent faire des prédictions précises sur des données nouvelles qui ne sont pas vues pendant l'entraînement. Ils peuvent donc apprendre des relations complexes entre les données d'entrée et de sortie, ce qui leur permet de généraliser et de prédire les sorties pour de nouvelles données. Cependant, la qualité des prédictions dépend fortement de la qualité et de la quantité des données d'entraînement. Si les données d'entraînement sont mauvaises ou insuffisantes, les prédictions pour de nouvelles données peuvent être inexactes [GOODFELLOW et al. 2016].

Les réseaux de neurones artificiels sont généralement caractérisés par :

- **Traitement parallèle** : les réseaux de neurones sont capables d'effectuer plusieurs calculs simultanément. Cela les rend bien adaptés aux tâches nécessitant des calculs à grande échelle, telles que la reconnaissance d'images, la reconnaissance de la parole, et la traduction automatique [GOODFELLOW et al. 2016].
- **Apprentissage hiérarchique** : les modèles d'apprentissage profond sont généralement structurés en plusieurs couches . Chaque couche possède un niveau d'abstraction différent. Cela permet au modèle d'apprendre des motifs et des relations complexes dans les données, et plus le réseau est profond, plus la capacité du modèle à découvrir ces relations est grande [GOODFELLOW et al. 2016].
- **Grandes quantités de données** : les modèles d'apprentissage profond nécessitent de grandes quantités de données pour s'entraîner efficacement. En effet, les modèles comportent un grand nombre de paramètres qui ne peuvent être réglés qu'à partir d'une grande quantité de données [GOODFELLOW et al. 2016].

1.3 Connexions et poids

Un réseau de neurones est constitué de nœuds et des connexions entre eux [AGGARWAL 2018]. Chaque nœud possède un **ensemble d'entrées** (qui sont souvent les sorties des nœuds de la couche précédente), un **poids** et une valeur ajoutée appelée le **biais**. Dans

les réseaux neuronaux, le biais est un paramètre supplémentaire qui est ajouté à chaque neurone pour ajuster sa sortie. Il permet au réseau de déplacer la fonction d'activation horizontalement[GOODFELLOW et al. 2016].

Lorsque des signaux entrent dans un neurones, chaque signal est multiplié par le poids associé à son entrée, puis additionné avec les autres résultats. Le biais est ensuite ajouté au résultat final et ce dernier est transmet vers les entrées des neurones de la couche suivante en passant par une fonction d'activation (voir la figure 1.1) [McCULLOCH et al. 1943].

On peut dire que la taille du réseau de neurones est définie par le nombre de ses paramètres et le nombre de ses couches, qui sont des variables appelées **hyperparamètres**. Par contre, les poids et le biais sont des paramètres entraînables. Au début de l'entraînement, on affecte à ces deux paramètres des valeurs aléatoires, et au fur et à mesure, les valeurs de ces deux paramètres sont ajustées et modifiées afin d'obtenir les bonnes valeurs [AGGARWAL 2018, GOODFELLOW et al. 2016].

1.4 Fonction d'activation

Un neurone dans le réseau artificiel calcule la somme pondérée de ses entrées et la valeur résultante de cette opération passe par une fonction appelée **fonction d'activation** (ou **fonction de transfert**) avant d'être transférée vers les neurones de la couche suivante. La sortie de neurone est donc calculée selon la formule 1.1 [McCULLOCH et al. 1943].

$$y = f \left(\sum_{i=1}^n w_i x_i + b \right) \quad (1.1)$$

Où :

- w_i : le poids associé à l'entrée i
- x_i : la valeur associée à l'entrée i
- n : le nombre total d'entrées
- b : le biais (constante entraînable ajoutée)
- f : la fonction d'activation
- y : la sortie du neurone

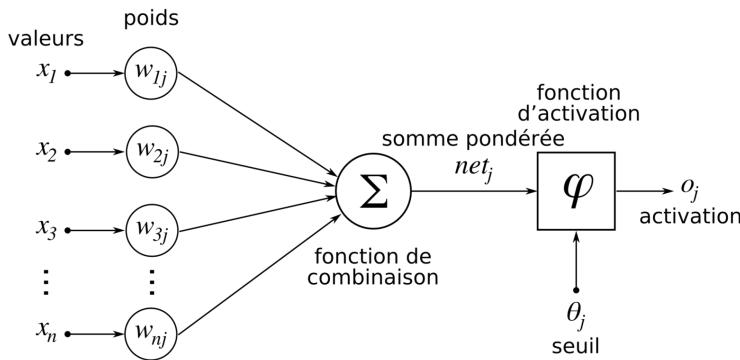


FIG. 1.1 : Le fonctionnement d'un neurone artificiel [MCCULLOCH et al. 1943].

La fonction d'activation est utilisée pour introduire de la non-linéarité dans le modèle, permettant ainsi de modéliser des relations complexes entre les données d'entrée et de sortie [GOODFELLOW et al. 2016]. Les propriétés d'une fonction d'activation doivent être vérifiées dans un problème d'apprentissage profond. Ces propriétés sont :

- **Non-linéarité** : lorsque la fonction d'activation est non linéaire, il est possible de prouver qu'un réseau neuronal à deux couches peut approximer n'importe quelle fonction continue sur un domaine compact à une précision arbitraire, ce que l'on appelle le **théorème d'approximation universelle** [GOODFELLOW et al. 2016].
- **L'intervalle** : lorsque l'intervalle des valeurs est fini, l'apprentissage de manière générale est plus efficace.
- **Differentiabilité** : cette propriété est importante quand les méthodes d'optimisation sont basées sur le gradient, car elles cherchent à optimiser l'apprentissage en se basant sur la différentiabilité de la fonction.
- **Monotonie** : Une fonction d'activation est monotone si sa sortie augmente (ou diminue) à mesure que son entrée augmente. Cela garantit que le gradient de la fonction est toujours positif ou négatif, simplifiant ainsi l'apprentissage.
- **Efficacité en termes de calcul** : les fonctions d'activation doivent être efficaces en termes de calcul, afin que le réseau puisse être utilisé dans des applications en temps réel, sans ralentir le processus de l'apprentissage.

Parmi les fonctions d'activation les plus couramment utilisées dans les réseaux de neurones, on peut citer :

- **La fonction Sigmoïde** : si la probabilité d'un résultat est comprise entre 0 et 1, la fonction sigmoïde est le meilleur choix. Cette fonction est largement utilisée grâce à son intervalle et sa différentiabilité.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1.2)$$

- **La fonction Unité linéaire rectifiée (ReLU)** : c'est une fonction qui possède une dérivée et permet la rétropropagation (backpropagation) tout en étant efficace

sur le plan informatique. Cependant, elle n'active pas les neurones en même temps, et c'est considéré comme désavantage pour cette fonction.

$$ReLU(x) = \max(0, x) \quad (1.3)$$

- **La fonction Tangente hyperbolique (Tanh)** : cette fonction est très identique à la fonction d'activation sigmoïde. Sa plage de sortie est comprise entre -1 et 1. Avec cette fonction, plus l'entrée est grande, plus la valeur de sortie sera proche de 1, et plus l'entrée est petite, plus la sortie sera proche de -1.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1.4)$$

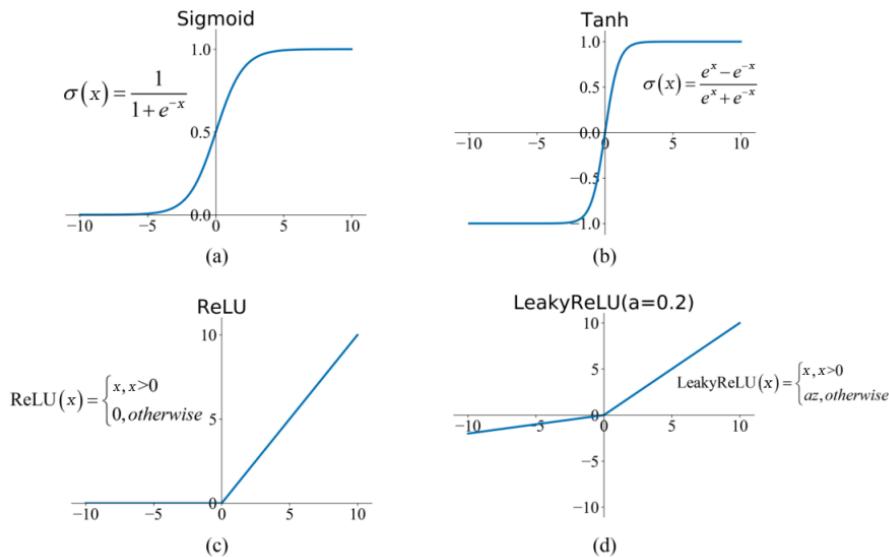


FIG. 1.2 : Les fonctions d'activations couramment utilisées [Junxi FENG et al. 2019].

1.5 Couches dans un réseau de neurones

Une couche (*layer* en anglais) est une succession verticale des neurones. Mathématiquement, elle est vue comme une composition de deux fonctions h et g où g est une fonction linéaire et h une fonction d'activation non linéaire. Cette composition de fonction est définie par l'équation 2.2 [GOODFELLOW et al. 2016].

$$y = h(g(x) + b) \quad (1.5)$$

Une couche intermédiaire est donc l'ensemble des noeuds verticaux qui sont connectés à la couche précédente et à la couche suivante. La connectivité entre les couches détermine la manière dont les informations circulent sur le réseau. La façon de connexions des noeuds entre eux est différente d'une architecture à une autre [GOODFELLOW et al. 2016], et c'est ce qui détermine le type d'une couche :

- **Couche entièrement connectée** : tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante.
- **Couche partiellement connectée** : certains neurones ne sont pas connectés aux neurones de la couche suivante.

Les couches sont le composant principal des réseaux de neurones. Elles ont plusieurs caractéristiques qui définissent leur comportement et influencent les performances globales du réseau. Ces caractéristiques sont les suivantes :

- **La matrice de poids** : Dans une couche d'un réseau de neurones, la matrice de poids est une matrice de paramètres qui représente les connexions entre les neurones d'entrée et les neurones de sortie de cette couche [AGGARWAL 2018]. Elle définit la puissance des connexions entre les neurones des différentes couches. Chaque ligne de la matrice correspond aux poids associés à un neurone d'entrée particulier, et chaque colonne correspond aux poids associés à un neurone de sortie particulier. La taille de la matrice de poids dépend du nombre de neurones d'entrée et du nombre de neurones de sortie dans la couche.

La forme générale de la matrice de poids dans un réseau de neurones peut être exprimée comme suit :

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,m} \\ w_{2,1} & w_{2,2} & \dots & w_{2,m} \\ \dots & \dots & \dots & \dots \\ w_{n,1} & w_{n,2} & \dots & w_{n,m} \end{bmatrix} \quad (1.6)$$

où $w_{i,j}$ représente le poids de la connexion entre le neurone i de la couche actuelle et le neurone j de la couche suivante et (n, m) représente la dimension de la matrice.

La matrice de poids est crucial pour la performance du réseau neuronal, puisqu'elle détermine la capacité du réseau d'apprendre et généraliser les motifs à partir des données en entrées.

- **Type de couche** : Les couches forment les blocs de construction de base des réseaux de neurones. Elle permettent d'effectuer des calculs complexes et d'apprendre des relations qui existent entre données d'entrée et de sortie. Dans le réseau neuronal, il existe trois types de couches différents :
 - **Couche d'entrée** : Cette couche est responsable de la réception des données d'entrée et de leur transmission à la couche suivante (la première couche parmi les couches cachées).
 - **Couche cachée** : Cette couche traite les entrées de la couche précédente et génère des valeurs de sortie qui sont transmises à la couche suivante. Les réseaux de neurones peuvent avoir plusieurs couches cachées, chacune effectuant différentes opérations sur les entrées.
 - **Couche de sortie** : Cette couche produit la sortie finale du réseau de neurones, qui peut être une classification, une régression ou un autre type de prédiction.

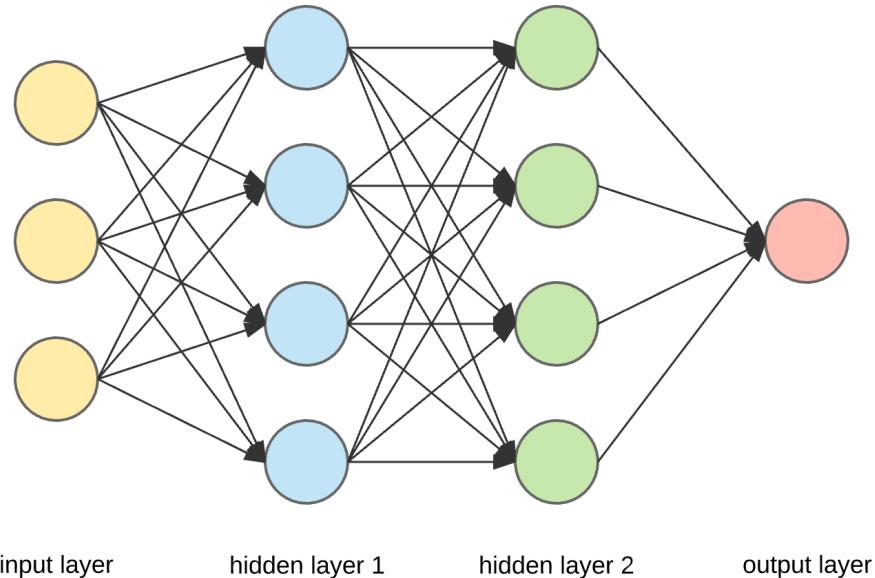


FIG. 1.3 : Schéma simple d'un réseau de neurones feedforward [MUNIASAMY et al. 2020].

1.6 Types de réseaux de neurones

Dans l'apprentissage profond, il existe plusieurs classes de réseaux de neurones, chacune avec sa propre architecture, caractéristiques, algorithme d'apprentissage et application. Dans cette section, nous allons présenter les différentes architectures de réseaux de neurones.

1.6.1 Réseaux feedforward

Les réseaux feedforward (ou réseaux entièrement connectés) sont un des types de réseau de neurones artificiels où les informations circulent dans une seule direction, de l'entrée vers la sortie [GOODFELLOW et al. 2016]. Ils sont composés d'une succession de couches interconnectées, où chaque neurone d'une couche est connecté aux neurones de la couche suivante. Dans un Réseau de ce type, les données sont introduites dans la première couche du réseau (couche d'entrée), puis elles traversent plusieurs couches cachées avant d'atteindre la couche de sortie (*la figure 1.4 est un schéma simple d'un réseau feedforward*).

Les réseaux feedforward sont indépendants de la structure, c'est-à-dire il n'existe pas d'hypothèses particulières à faire sur l'entrée, ce qui les rend largement applicables. Cependant, ils ont tendance à être moins performants que les réseaux à usage spécial. Les réseaux feedforward sont couramment utilisés dans les applications d'apprentissage supervisé. Ils peuvent également être utilisés dans des applications d'apprentissage non supervisé [AGGARWAL 2018].

1.6.2 Réseaux de neurones récurrents (RNN)

Les réseaux de neurones récurrents (RNN) sont des architectures conçus pour fonctionner avec des données séquentielles, telles que : la reconnaissance de la parole, la recon-

naissance de la voie, l'analyse de séries chronologiques et le traitement du langage naturel [GOODFELLOW et al. 2016]. Les principales caractéristiques des RNN sont :

- **Connexions récurrentes** : Les connexions dans les réseaux récurrents sont des connexions récurrentes qui permettent à l'information de persister au fil du temps. Cela signifie que la sortie du réseau à un pas de temps est réinjectée en entrée du réseau au pas de temps suivant.
- **État caché** : Les réseaux RNN maintiennent un état caché qui représente la mémoire du réseau. Cet état est mis à jour à chaque pas de temps en fonction de l'entrée courante et de l'état caché précédent.
- **RNN bidirectionnels** : Les réseaux RNN bidirectionnels traitent la séquence d'entrée dans les sens avant et arrière, ce qui permet de capturer le contexte des pas de temps passés et futurs.

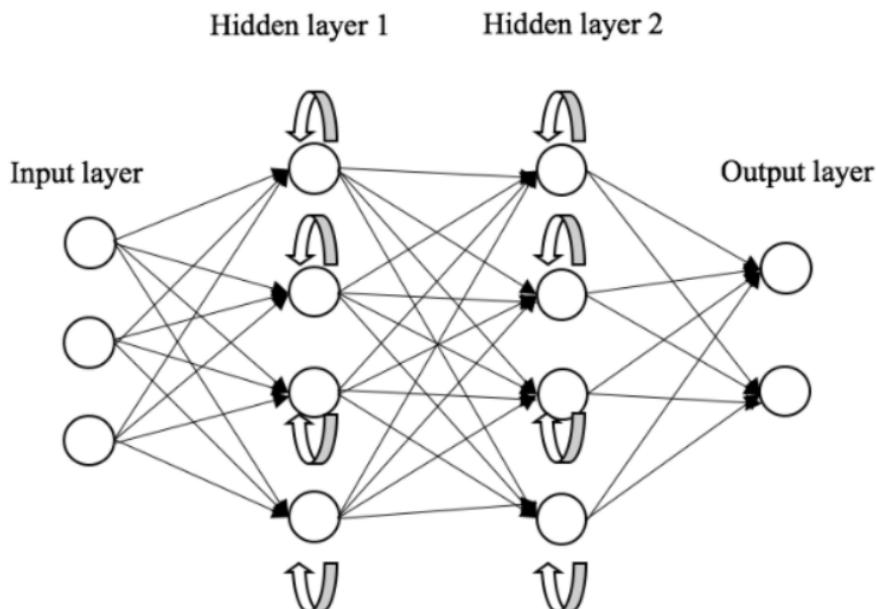


FIG. 1.4 : Exemple d'un réseau de neurones récurrent [KUMARASWAMY 2021].

La capacité à conserver une mémoire des pas de temps précédents et à gérer les dépendances à long terme rend les réseaux récurrents utiles pour les tâches qui nécessitent de comprendre le contexte de la séquence d'entrée.

Réseaux récurrents à mémoire courte et long terme (LSTM)

Les réseaux de neurones récurrents à mémoire longue à court terme, ou Long Short-Term Memory (LSTM) en anglais, représentent une amélioration significative des réseaux de neurones récurrents traditionnels, conçue pour résoudre les problèmes d'évanouissement et d'explosion du gradient. Introduits par Hochreiter et Schmidhuber en 1997 dans [HOCHREITER et al. 1997], les réseaux LSTM intègrent des cellules mémoire capables

de stocker et de conserver des informations tout au long du traitement d'une séquence. Ces cellules mémoire permettent de transporter des informations importantes grâce à trois types de portes : la porte d'entrée, la porte de sortie et la porte d'oubli. Ces portes jouent un rôle crucial en décidant quelles informations doivent être ajoutées, conservées ou oubliées.

Porte d'Entrée : décide quelles nouvelles informations doivent être stockées dans la cellule de mémoire. Elle est définie par :

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1.7)$$

où σ représente la fonction sigmoïde, W_i est le poids associé à la porte d'entrée, h_{t-1} est l'état caché précédent, x_t est l'entrée actuelle, et b_i est le biais.

Porte d'Oubli : contrôle quelles informations anciennes doivent être effacées de la cellule de mémoire. Elle est définie par :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1.8)$$

où σ est la fonction sigmoïde, W_f est le poids associé à la porte d'oubli, h_{t-1} est l'état caché précédent, x_t est l'entrée actuelle, et b_f est le biais.

Porte de Sortie : décide quelles informations de la cellule de mémoire sont utilisées pour calculer l'état caché actuel. Elle est définie par :

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (1.9)$$

où σ est la fonction sigmoïde, W_o est le poids associé à la porte de sortie, h_{t-1} est l'état caché précédent, x_t est l'entrée actuelle, et b_o est le biais.

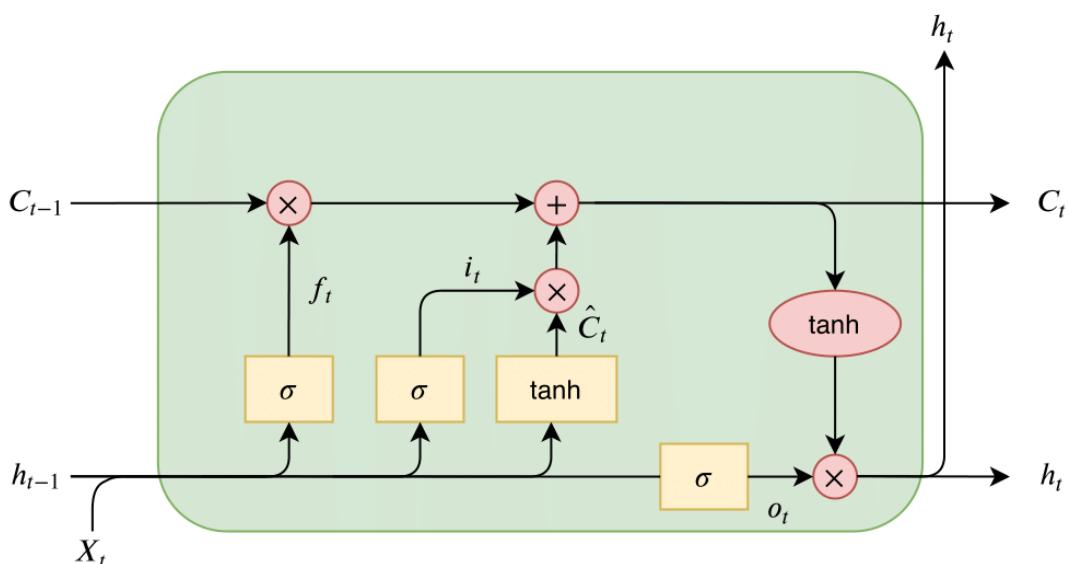


FIG. 1.5 : Architecture d'un bloc LSTM [FAWAZ et al. 2019a].

1.6.3 Réseaux de neurones convolutifs (CNN)

L'architecture des réseaux de neurones convolutifs (CNN) est une architecture spéciale qui est bien adaptée à la tâche de classification d'images. Ces réseaux comportent trois types de couches : **convolutives**, **pooling** et **d'activation** [GOODFELLOW et al. 2016].

Les couches convolutives sont appliquées à l'image en entrée pour l'extraction des caractéristiques importantes de l'image. Ensuite, ces dernières traversent des couches d'activation, qui sont responsables de l'application d'une fonction d'activation non-linéaire à ces caractéristiques. Les couches d'activation sont suivies de couches de pooling qui réduisent la taille de l'image. Les couches de pooling sont elles même suivies par une couche entièrement connectée qui donne la classification de l'image (la sortie finale du modèle) [GOODFELLOW et al. 2016].

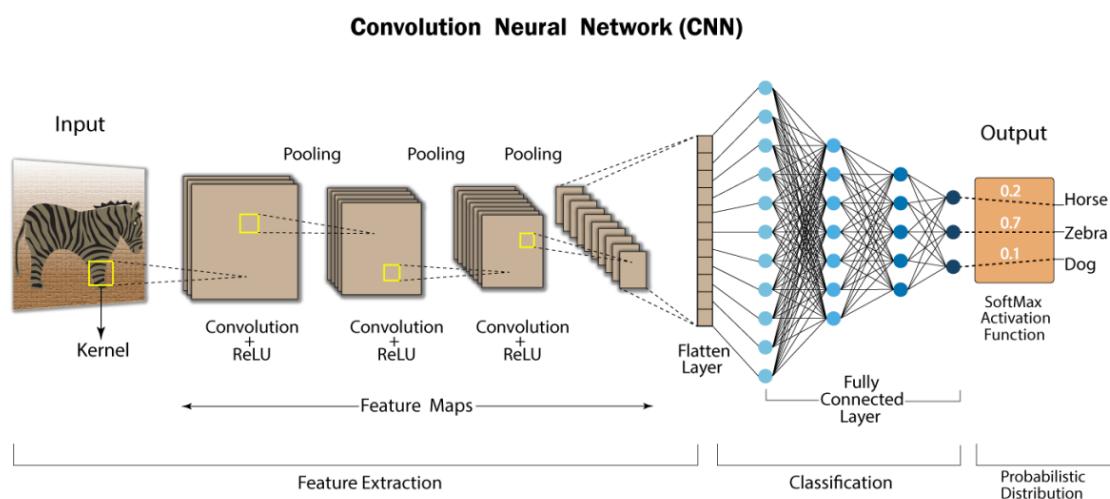


FIG. 1.6 : Le fonctionnement d'un réseau neuronal convolutif [ALHARBI et al. 2021].

Couche convulsive

Une couche convulsive est un élément constitutif des réseaux de neurones convolutifs (CNN). Elle est utilisée pour extraire des caractéristiques à partir de données d'entrée, souvent des images, en appliquant un ensemble de filtres convolutifs appris à l'entrée.

Dans une couche convulsive, chaque filtre est convolué avec l'entrée pour produire une carte de caractéristiques. Cette opération est faite en glissant le filtre sur l'entrée et en calculant le produit scalaire à chaque position. Généralement, une couche convulsive possède trois hyperparamètres qui doivent être définis : **le nombre de filtres**, **la taille des filtres** et **la Stride**. Le nombre de filtres détermine le nombre de cartes d'entités produites, tandis que la taille des filtres détermine la taille du champ récepteur de chaque carte d'entités. La stride détermine la quantité de décalage du filtre à chaque étape.

Les couches convolutives sont suivies de fonctions d'activation, et de couches de pooling. Ces dernières permettent de réduire les dimensions spatiales des cartes d'entités. Plusieurs

couches convolutives peuvent être empilées pour créer un réseau neuronal convolutif profond. Les couches convolutives sont particulièrement efficaces pour traiter des images et d'autres données de grande dimension avec une structure spatiale, car elles peuvent apprendre automatiquement à détecter les caractéristiques importantes, telles que les bords, les coins et les textures [KIMURA et al. 2019, GOODFELLOW et al. 2016].

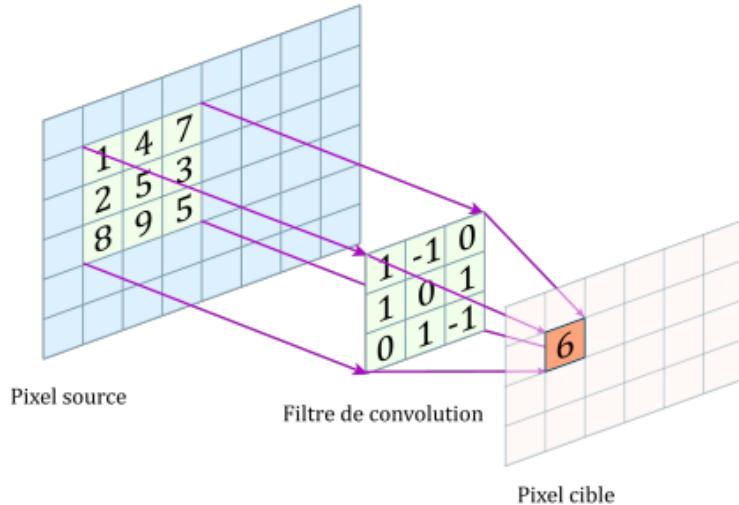


FIG. 1.7 : Exemple du fonctionnement d'une couche convulsive [KIMURA et al. 2019].

Couche pooling

Le pooling est une opération qui est utilisée pour sous-échantillonner les cartes de caractéristiques résultantes des couches convolutives en réduisant leurs dimensions spatiales tout en conservant les informations importantes. Parmi les types de pooling, on trouve le pooling maximal et le pooling moyen [GOODFELLOW et al. 2016].

Dans le pooling maximal, la valeur maximale de chaque région est sélectionnée comme valeur représentative, tandis que dans le pooling moyen, la valeur moyenne est calculée à la place. Il en résulte une carte d'entités plus petite avec une résolution spatiale réduite, qui peut être traitée plus efficacement par les couches suivantes du réseau.

Cependant, le pooling excessive peut entraîner une perte d'informations spatiales importantes. Il est donc important d'équilibrer la quantité de pooling avec les besoins du réseau et la nature de la tâche à accomplir.

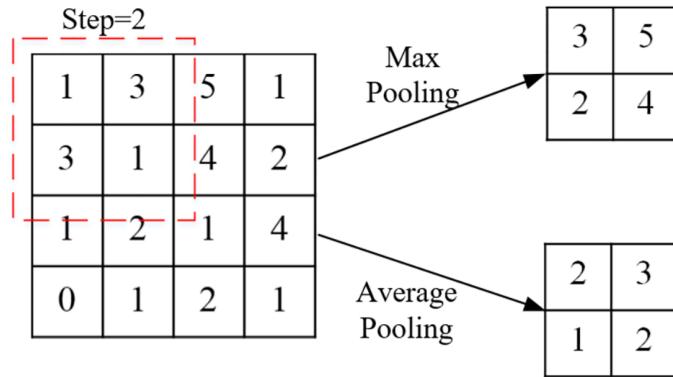


FIG. 1.8 : Exemples de pooling maximal et pooling moyen [HU et al. 2022].

1.6.4 Réseaux résiduels (ResNet)

Les réseaux résiduels ont été conçus par Microsoft afin de résoudre le problème des gradients qui disparaissent dans les réseaux de neurones très profonds, ce qui peut rendre l'entraînement difficile et diminuer significativement les performances du réseau.

Un ResNet est composé d'une ensemble de blocs résiduels, qui sont constitués de plusieurs couches avec des connexions de raccourci qui contournent une ou plusieurs couches. Ces raccourcis permettent aux gradients de circuler plus facilement à travers le réseau et évitent qu'ils ne disparaissent à mesure que le réseau devient plus profond [HE et al. 2016].

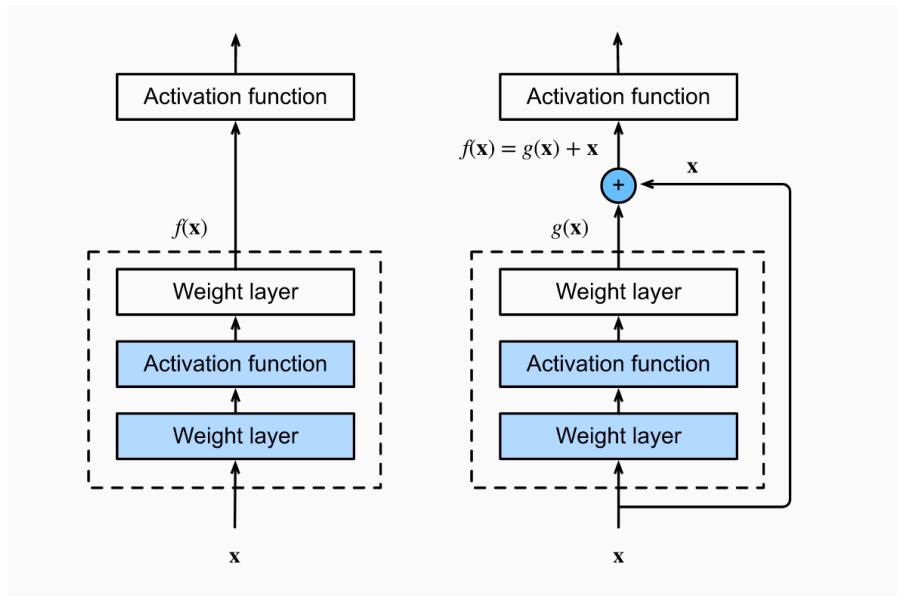


FIG. 1.9 : Un bloc régulier (gauche) et un bloc résiduel (droite) [DONG et al. 2022].

ResNet est très efficace dans les tâches de vision par ordinateur, telles que la classification d'images, la détection d'objets et la segmentation. Il a joué un rôle important dans l'avancement de l'état de l'art en apprentissage profond et en vision par ordinateur, et continue d'être un domaine de recherche actif.

1.7 Le processus d'apprentissage

L'apprentissage est le processus itératif et continu d'ajustement des paramètres du réseau neuronal afin d'obtenir une meilleure précision du modèle [GOODFELLOW et al. 2016]. Il peut être complexe car il nécessite souvent une combinaison de techniques telles que le **prétraitement** des données, le choix de l'architecture du réseau de neurones et des hyperparamètres, ainsi que l'algorithme d'optimisation.

Ce processus d'apprentissage commence d'abord par l'initialisation aléatoire des paramètres (poids) du modèle. Ensuite, le modèle est entraîné sur un ensemble de données (**dataset**) en utilisant un algorithme d'optimisation pour ajuster les poids du modèle afin de minimiser la perte.

Lors de l'entraînement, le modèle est alimenté en entrée avec des exemples à partir du dataset d'entraînement et compare sa sortie à la sortie attendue. Ensuite, il calcule la perte en faisant la différence entre la sortie prédite et la sortie attendue. L'algorithme de **rétropropagation** de gradient ajuste les poids du modèle en calculant les gradient de la fonction de perte afin de la minimiser.

Le processus d'apprentissage continue jusqu'à ce que la performance du modèle sur le dataset de validation arrête de s'améliorer ou jusqu'à ce qu'un nombre prédéfini d'itérations d'apprentissage soit atteint. Le modèle final est ensuite utilisé pour effectuer des prédictions sur de nouvelles données.

Ce processus peut être résumé dans les points suivants :

- L'apprentissage consiste à modifier progressivement les paramètres (poids) en passant un lot de données en entrée et en évaluant le taux de perte.
- La définition de la fonction de perte est utilisée pour les modifications de réseaux dans le processus de l'entraînement.
- La modification des poids du réseau est effectuée grâce à l'algorithme de rétropropagation (backpropagation).

1.7.1 Descente de gradient

La descente de gradient est une méthode utilisée dans le processus d'optimisation. Elle est basée sur la différentiabilité d'une fonction et elle est appliquée pour calculer la fonction dérivée du premier ordre pour trouver le minimum de la fonction de perte [GOODFELLOW et al. 2016]. Sa simplicité d'application est l'un des avantages de cette méthode.

L'algorithme commence par l'initialisation aléatoire des paramètres (poids) du modèle. Ensuite, il calcule le gradient de la fonction de perte par rapport à chaque paramètre. Le gradient indique la direction dans laquelle la fonction de perte augmente le plus, donc l'algorithme met à jour les paramètres dans la direction opposée au gradient pour réduire la valeur de perte. Ce processus est répété itérativement jusqu'à ce que la valeur de perte ne puisse plus être réduite ou jusqu'à ce qu'un critère d'arrêt soit atteint. Le taux

d'apprentissage est un hyperparamètre qui contrôle la taille des mises à jour de paramètres et la vitesse de convergence de l'algorithme [GOODFELLOW et al. 2016].

Des variantes de la descente de gradient existent, telles que la descente de gradient stochastique, Adam et la descente de gradient avec moment.

1.7.2 Propagation de l'erreur

La propagation d'erreur est utilisée dans le processus d'apprentissage pour calculer le gradient de la fonction de perte par rapport aux paramètres du modèle [GOODFELLOW et al. 2016].

Dans la propagation d'erreur, l'erreur est propagée en arrière à travers les couches du modèle, en commençant par la couche de sortie et en allant vers l'entrée. Chaque couche calcule la dérivée de sa sortie par rapport à ses entrées, qui est ensuite multipliée par l'erreur propagée de la couche suivante. Ce processus est répété jusqu'à ce que l'erreur soit propagée jusqu'à la couche d'entrée, où le gradient de la fonction de perte par rapport aux paramètres du modèle est obtenu [GOODFELLOW et al. 2016].

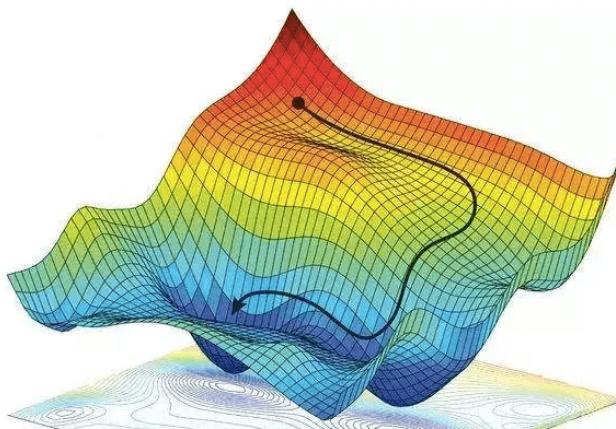


FIG. 1.10 : Fonction de taux d'erreur [AMINI et al. 2018].

1.7.3 Hyperparameters

Les hyperparamètres sont les paramètres qui sont définis avant de lancer le processus d'apprentissage et ils contrôlent l'entraînement. Les valeurs des hyperparamètres, contrairement aux paramètres du modèle, ne sont pas apprises lors de l'apprentissage [GOODFELLOW et al. 2016]. Parmi les hyperparamètres, on peut définir :

- **Taux d'apprentissage** : Il contrôle la vitesse d'apprentissage du modèle à partir des données.
- **Nombre d'époques** : Le nombre d'époques correspond au nombre d'itération sur le dataset d'entraînement.

- **Taille du batch** : Elle représente le nombre d'échantillons des données d'entraînement qui sont utilisés dans un passage avant/arrière (feedforward et backpropagation).
- **Nombre de couches** : C'est un hyperparamètre qui caractérise la profondeur du réseau.
- **Fonction d'activation** : La fonction d'activation est utilisée pour introduire la non-linéarité dans le modèle. C'est un hyperparamètre qui contrôle la sortie du neurone.
- **Initialisation des poids** : Les valeurs initiales des poids peuvent affecter de manière significative les performances du modèle. Elle affecte le processus de trouver le minimum local ou le minimum global.
- **Paramètre de régularisation** : Il est utilisé pour éviter le **surapprentissage**, c'est-à-dire éviter de construire un modèle qui est trop complexe par rapport à la quantité de données d'entraînement. Cela peut entraîner une adaptation excessive du modèle aux données d'entraînement et une mauvaise généralisation aux données inconnues.
- **Optimiseur** : L'optimiseur est l'algorithme utilisé pour mettre à jour les poids pendant l'entraînement.

Cependant, le réglage de ces hyperparamètres peut être une tâche complexe nécessitant souvent des essais et des erreurs.

1.8 Types d'apprentissage

Il existe plusieurs types d'apprentissage, et chacun de ces types a des applications spécifiques en deep learning, et peut être utilisé pour résoudre différents types de problèmes. Dans ce qui suit, nous parlons brièvement sur les quatre types d'apprentissage en deep learning les plus courants : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage par renforcement.

1.8.1 Apprentissage supervisé

Dans l'apprentissage supervisé, l'apprentissage s'effectue sur un ensemble de données étiqueté, où les étiquettes sont connues à l'avance. En d'autres termes, les données d'entrée sont accompagnées d'étiquettes de sortie ou de valeurs cibles correspondantes. Le but de l'apprentissage supervisé est d'apprendre à prédire les étiquettes à partir des données d'entrée, de sorte que lorsqu'on donne au modèle de nouvelles données en entrée, l'algorithme puisse prédire la sortie correspondante. Pendant le processus d'apprentissage, l'algorithme ajuste itérativement ses paramètres pour minimiser la différence entre la sortie prédite et la sortie réelle [AGGARWAL 2018, GOODFELLOW et al. 2016].

Ce type d'apprentissage est applicable dans plusieurs domaines, tels que la reconnaissance d'images et de la parole, le traitement du langage naturel et la modélisation prédictive dans la finance, la santé, le marketing, etc. Parmi les algorithmes d'apprentissage supervisés, nous pouvons citer la régression linéaire et les arbres de décision.

1.8.2 Apprentissage non-supervisé

L'apprentissage non supervisé se fait sur un dataset non étiqueté, sans aucune information sur les étiquettes [GOODFELLOW et al. 2016]. En d'autres termes, il n'y a pas de valeurs cibles ou d'étiquettes de sortie fournies à l'algorithme. L'algorithme est laissé à lui-même pour trouver les motifs et relations dans les données, sans recevoir d'instructions explicites sur ce qu'il faut rechercher.

L'utilisation de cette approche d'apprentissage peut impliquer le regroupement de points de données similaires, la découverte de structures ou de caractéristiques cachées dans les données (réduction de la dimensionnalité) ou l'identification de valeurs aberrantes ou d'anomalies dans les données.

Nous pouvons appliquer l'apprentissage non supervisé dans divers domaines, tels que la segmentation des marchés, la détection d'anomalies et l'extraction de caractéristiques. Parmi les algorithmes d'apprentissage non supervisés courants, on trouve le clustering k-means, l'analyse en composantes principales (PCA) et les auto-encodeurs.

L'un des principaux défis dans l'apprentissage non supervisé est d'évaluer la qualité des résultats, car il n'y a pas d'objectifs ou d'étiquettes explicites à comparer. Au lieu de cela, les résultats sont souvent évalués en fonction de leur utilité ou de leur interprétabilité pour une tâche ou un domaine particulier.

1.8.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé est un type d'apprentissage qui combine des éléments d'apprentissage supervisé et non supervisé. Dans l'apprentissage semi-supervisé, un petit ensemble de données étiquetées est fourni, tandis que la grande partie de données est non étiquetées [GOODFELLOW et al. 2016].

Le but dans l'apprentissage semi-supervisé est d'utiliser les données étiquetées pour guider le processus d'apprentissage sur les données non étiquetées, afin d'améliorer la précision du modèle. Cela peut être particulièrement utile dans les situations où il est difficile ou coûteux d'obtenir des données étiquetées, mais il existe une abondance de données non étiquetées disponibles.

Il existe plusieurs approches, mais une méthode courante consiste à utiliser les données étiquetées pour créer un modèle, puis à utiliser ce modèle pour faire des prédictions sur les données non étiquetées. Les étiquettes prédites sont ensuite utilisées pour améliorer le modèle, et le processus est répété de manière itérative.

L'un des défis de l'apprentissage semi-supervisé est que la qualité des résultats peut

dépendre fortement de la distribution des données non étiquetées. Si les données non étiquetées ne sont pas représentatives du domaine cible, le modèle peut mal fonctionner même avec une grande quantité de données non étiquetées.

1.8.4 Apprentissage par renforcement

L'apprentissage par renforcement se concentre sur la prise de décision. Il s'agit d'une méthode d'apprentissage dans laquelle un agent apprend à prendre des décisions en interagissant avec un environnement. L'agent doit choisir une action à partir d'un état donné, et l'environnement renvoie un signal de récompense ou de pénalité en fonction de l'action choisie. L'objectif de l'agent est de maximiser la récompense totale sur une période donnée [WIERING et al. 2012]. Dans le chapitre suivant, nous présenterons l'apprentissage par renforcement et nous en parlerons avec plus de détails.

1.9 Catégories de données

La division du jeu de données est une étape cruciale avant de commencer l'entraînement du modèle de manière. Elle permet d'éviter les problèmes de surapprentissage. Il est courant de diviser un jeu de données en trois parties distinctes : l'ensemble d'entraînement, l'ensemble de validation et l'ensemble de test [GOODFELLOW et al. 2016].

- **Ensemble d'entraînement** : Il s'agit de la partie de données utilisée pour entraîner le modèle. Il est important que ce dataset soit représentatif de l'ensemble des données et qu'il contienne une variété de cas d'utilisation différents.
- **Ensemble de validation** : Il s'agit d'un sous-ensemble de l'ensemble de données utilisé pour évaluer les performances du modèle pendant l'entraînement. L'ensemble de validation est utilisé pour régler les hyperparamètres du modèle et pour éviter le surapprentissage. Il est important qu'il soit représentatif et distinct du dataset d'entraînement.
- **Ensemble de test** : Il s'agit d'un ensemble de données utilisé pour évaluer les performances du modèle après son entraînement. L'ensemble de test est utilisé pour obtenir une estimation impartiale de la performance du modèle sur de nouvelles données inédites, donc il est nécessaire que ce dataset soit représentatif de l'ensemble des données et distinct des deux autres datasets cités précédemment.

La distinctivité des datasets assure que le modèle se généralise bien aux nouvelles données invisibles. En règle générale, l'ensemble de données est divisé en ces trois ensembles dans un rapport de 60-20-20 ou 70-15-15 respectivement.

1.10 Défis de l'apprentissage profond

L'apprentissage profond a fait des progrès remarquables ces dernières années et a obtenu des résultats excellents dans divers domaines tels que la vision par ordinateur,

le traitement du langage naturel et la reconnaissance de la parole. Cependant, il reste encore plusieurs défis à relever afin d'améliorer encore l'efficacité et l'efficience des modèles d'apprentissage profond. Certains défis majeurs incluent :

- **Rareté des données** : les modèles d'apprentissage profond nécessitent une grande quantité de données pour être entraînés efficacement. Cependant, dans de nombreux domaines, tels que l'imagerie médicale et la conduite autonome, les données sont rares et coûteuses à collecter.
- **Surapprentissage (overfitting)** : les modèles d'apprentissage profond peuvent facilement sur-adapter les données d'apprentissage, en particulier lorsque le modèle comporte un grand nombre de paramètres. Le surapprentissage peut entraîner de mauvaises performances de généralisation sur de nouvelles données.
- **Interprétabilité** : les modèles d'apprentissage profond sont souvent appelés "boîtes noires" car il peut être difficile de comprendre comment ils arrivent à leurs prédictions. Ce manque d'interprétabilité peut compliquer le débogage et l'amélioration des modèles de l'apprentissage profond.
- **Limitations matérielles** : les modèles d'apprentissage profond sont coûteux en termes de calcul et nécessitent un matériel spécialisé tel que des unités de traitement graphique (GPU) ou des unités de traitement de tenseur (TPU). Le coût de ce matériel peut constituer un obstacle pour les petits groupes de chercheurs ou les entreprises.
- **Attaques contradictoires** : les modèles d'apprentissage profond peuvent être vulnérables aux attaques contradictoires, où un attaquant manipule délibérément les données d'entrée pour amener le modèle à faire des prédictions incorrectes.

1.11 Conclusion

En conclusion, l'apprentissage profond est devenu un domaine de recherche actif de l'apprentissage automatique qui a révolutionné la façon dont nous abordons de nombreux problèmes difficiles, tels que la vision par ordinateur, le traitement du langage naturel et la robotique. Avec l'avènement d'un matériel puissant, d'ensembles de données à grande échelle et d'algorithmes sophistiqués, les modèles d'apprentissage profond ont connu un avancement remarquable dans plusieurs domaines tels que la reconnaissance d'images, la reconnaissance vocale, la traduction linguistique et la conduite autonome.

Malgré son énorme succès, l'apprentissage en profondeur fait encore face à plusieurs défis, tels que le besoin d'algorithmes d'entraînement plus efficaces et fiables, une meilleure interprétabilité, etc. L'utilisation et l'entraînement de modèles d'apprentissage profond exigent des ordinateurs assez puissants et ne peuvent pas être utilisés sur les appareils moins puissants comme les ordinateurs embarqués ou les smartphones, ce qui signifie qu'on a besoin de trouver des moyens pour optimiser ces modèles afin de les utiliser ultérieurement par les machines moins puissantes. Ces méthodes d'optimisation font l'objet du dernier chapitre.

Chapitre 1. Apprentissage profond

Dans ce qui suit, nous présenterons l'apprentissage profond et ses algorithmes. La raison pour laquelle on a réservé un chapitre complet pour l'apprentissage profond est que ce type d'apprentissage peut aider beaucoup dans l'optimisation des réseaux de neurones profonds, comme nous le verrons plus tard.

Chapitre 2

Intelligence Artificielle Générative (GenAI)

2.1 Introduction

Avec les avancées fulgurantes du deep learning, de nouvelles méthodes d'intelligence artificielle ont émergé, souvent désignées sous le terme de modèles génératifs. Ces technologies de pointe, telles que les auto-encodeurs variationnels, les réseaux adversariaux génératifs (GANs), les modèles de diffusion, les transformers et les modèles de langage de grande taille (LLM) sont capables de produire des données d'une qualité impressionnante, ressemblant de manière frappante aux données réelles. Les données générées par ces modèles sont souvent indiscernables des données authentiques, ce qui pose de nouveaux défis en matière d'identification et d'authenticité. Les modèles génératifs permettent la génération de textes, d'images, de séries temporelles et de données tabulaires avec une grande précision.

La puissance de ces modèles repose sur des ressources de calcul considérables, notamment l'utilisation de GPU, et sur l'accès à des ensembles de données massifs. Par exemple, des modèles tels que ChatGPT-4 ont été entraînés sur l'intégralité du contenu disponible sur Internet.

Dans ce chapitre, nous allons examiner en détail les différentes architectures de ces modèles génératifs, ainsi que les techniques et pratiques associées à leur fonctionnement et à leur mise en œuvre. Nous aborderons les principes de base, les innovations récentes, et les applications potentielles de ces technologies révolutionnaires.

2.2 Les Auto-Encodeurs Variationnels (VAE)

2.2.1 Introduction

Les auto-encodeurs variationnels (VAE) sont des modèles génératifs puissants, reconnus pour leur capacité à apprendre une représentation compacte et structurée des données.

Cette approche a révolutionné le domaine de l'apprentissage non supervisé et des modèles génératifs. Les VAE appartiennent à la classe des modèles génératifs probabilistes, combinant les principes des autoencodeurs et des modèles de variational Bayes pour générer des données nouvelles et similaires à celles d'un jeu de données d'entraînement. Depuis leur introduction par Kingma et Welling en 2013 [KINGMA et al. 2012], les VAE ont connu un grand succès dans diverses applications, allant de la génération d'images à la synthèse de texte.

2.2.2 Fonctionnement

Les VAE sont composés de trois éléments principaux : l'encodeur, le décodeur et l'espace latent. Chacune de ces composantes joue un rôle crucial dans le fonctionnement global du modèle.

L'Encodeur : L'encodeur est responsable de la transformation des données d'entrée \mathbf{x} en une distribution dans l'espace latent. Plus précisément, il mappe les données d'entrée à une distribution gaussienne paramétrée par une moyenne μ et une variance σ^2 . Cette distribution est souvent représentée comme :

$$q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \mu(\mathbf{x}), \sigma^2(\mathbf{x})) \quad (2.1)$$

où \mathbf{z} est la variable latente que l'encodeur cherche à estimer.

Le Décodeur : Le décodeur prend un échantillon \mathbf{z} de la distribution gaussienne dans l'espace latent et génère une reconstruction $\hat{\mathbf{x}}$ des données d'entrée. Il modélise la distribution des données d'entrée conditionnellement à \mathbf{z} comme suit :

$$p(\mathbf{x} \mid \mathbf{z}) \quad (2.2)$$

Typiquement, le décodeur est un réseau neuronal qui produit les paramètres de cette distribution, souvent supposée gaussienne ou Bernoulli selon la nature des données.

L'Espace latent : est la représentation comprimée et structurée des données d'entrée. Il est généralement conçu comme un espace continu, où chaque point de l'espace latent correspond à une instance possible de données générées. L'idée est que cet espace latent capture les caractéristiques essentielles des données d'entrée de manière à permettre la génération de nouvelles instances en échantillonnant de cet espace.

L'objectif principal d'un VAE est d'optimiser une fonction de coût qui combine deux termes principaux : la divergence de Kullback-Leibler (KL) [JOHNSON et al. 2001] et la vraisemblance de reconstruction.

Divergence de Kullback-Leibler (KL) : Ce terme mesure la différence entre la distribution latente approximée $q(z|x)$ et la distribution prior $p(z)$. La divergence KL est donnée par :

$$D_{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})) \quad (2.3)$$

où $q(z|x)$ est la distribution gaussienne paramétrée par l'encodeur, et $p(z)$ est généralement choisie comme une distribution normale standard.

Vraisemblance de Reconstruction

Ce terme mesure la capacité du modèle à reconstruire les données d'entrée x à partir de la variable latente z . Il est donné par la vraisemblance de x conditionnée par z :

$$\log p(\mathbf{x} | \mathbf{z}) \quad (2.4)$$

Fonction de Perte VAE

La fonction de coût totale d'un VAE, également appelée fonction de perte VAE, combine ces deux termes. Elle peut être formulée comme suit :

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} | \mathbf{z})] + D_{KL}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z})) \quad (2.5)$$

où θ et ϕ sont les paramètres du décodeur et de l'encodeur respectivement. Le premier terme, $-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|z)]$, représente l'erreur de reconstruction, et le second terme, $D_{KL}(q_\phi(z|x) \| p(z))$, régularise la distribution latente pour qu'elle soit proche de la distribution prior.

En optimisant cette fonction de perte, les VAE parviennent à apprendre des représentations latentes qui permettent une reconstruction fidèle des données d'entrée tout en assurant une régularité statistique dans l'espace latent.

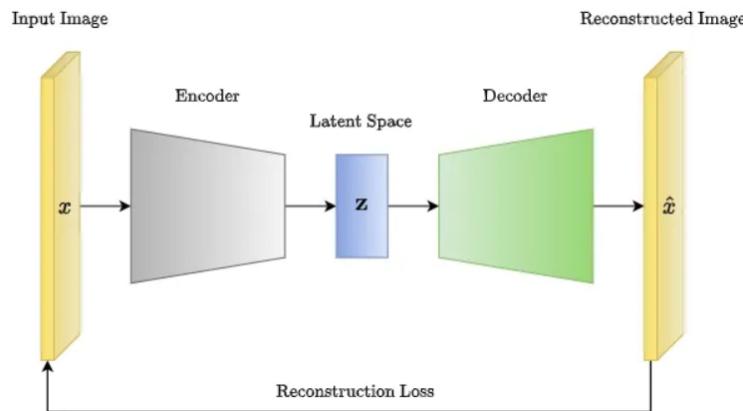


FIG. 2.1 : Le fonctionnement d'un neurone artificiel [KINGMA et al. 2012].

2.2.3 Applications des Auto-Encodeurs Variationnels (VAE)

Les auto-encodeurs variationnels (VAE) ont démontré leur efficacité dans divers domaines grâce à leur capacité à apprendre des représentations latentes significatives. Voici quelques-unes des principales applications des VAE :

Génération d'Images

Les VAE sont largement utilisés pour générer des images réalistes. En apprenant une représentation latente des images d'entraînement, les VAE peuvent échantillonner de cet espace latent pour créer de nouvelles images qui partagent des caractéristiques similaires avec les données d'origine. Cela a des applications potentielles dans la création de contenu, la conception assistée par ordinateur et la synthèse d'images médicales [LUHMAN et al. 2024].

Synthèse de Texte

Les VAE peuvent être appliqués à la génération de texte en apprenant les structures latentes des séquences de texte. Ils permettent de produire des phrases cohérentes et des paragraphes qui imitent le style et le contenu des textes d'entraînement. Cette technique est utile dans des domaines tels que la génération automatique de rapports, l'écriture créative assistée et la traduction automatique[W. WANG et al. 2019].

Compression de Données

Les VAE sont efficaces pour la compression de données, en particulier pour des données de grande dimension comme les images et les vidéos. En apprenant une représentation latente compacte, les VAE peuvent réduire la dimensionnalité des données tout en préservant leurs caractéristiques essentielles. Cela permet une transmission et un stockage plus efficaces des données compressées[YILMAZ et al. 2021].

Détection d'Anomalies

Dans les systèmes de détection d'anomalies, les VAE peuvent être utilisés pour identifier des données aberrantes qui diffèrent significativement des données d'entraînement. En apprenant une distribution latente des données normales, le VAE peut détecter des échantillons qui ne correspondent pas à cette distribution, ce qui est particulièrement utile dans la surveillance de systèmes industriels, la cybersécurité et la détection de fraudes [Z. WANG et al. 2019].

Imputation de Données Manquantes

Les VAE peuvent également être employés pour l'imputation de données manquantes. En apprenant la structure latente des données complètes, les VAE peuvent estimer les valeurs manquantes de manière cohérente avec les données observées. Cette application est cruciale dans des domaines tels que l'analyse de données médicales, les études socio-économiques et les bases de données incomplètes [COLLIER et al. 2020].

En résumé, les VAE sont des outils polyvalents dans le domaine de l'apprentissage machine, avec des applications variées allant de la génération de contenu à la compression de données et à la détection d'anomalies. Leur capacité à apprendre des représentations

latentes significatives permet de traiter efficacement une large gamme de problèmes complexes.

2.3 Réseaux Antagonistes Génératifs (GANs)

2.3.1 Introduction aux Réseaux Antagonistes Génératifs

Les réseaux antagonistes génératifs, ou Generative Adversarial Networks (GANs) en anglais, sont une classe de modèles génératifs introduite par Ian Goodfellow et ses collègues en 2014 [GOODFELLOW et al. 2014]. Les GANs se composent de deux réseaux de neurones concurrents : un générateur et un discriminateur. Le générateur cherche à produire des échantillons réalistes à partir d'un bruit aléatoire, tandis que le discriminateur tente de distinguer les échantillons réels des échantillons générés. Cette compétition incite les deux réseaux à s'améliorer simultanément, aboutissant à la génération de données de haute qualité.

2.3.2 Fonctionnement des GANs

Les GANs sont des modèles très performants et sont adaptés à plusieurs architectures variées. Cependant, ils se composent principalement de deux réseaux de neurones principaux :

Générateur

Le réseau générateur prend un vecteur de bruit z tiré d'une distribution uniforme ou normale et produit une donnée synthétique $G(z)$. Mathématiquement, le générateur peut être représenté comme une fonction $G : Z \rightarrow X$, où Z est l'espace du bruit et X est l'espace des données. L'objectif du générateur est de "tromper" le discriminateur en générant des données indiscernables des données réelles.

$$G(\mathbf{z}; \theta_g) : \mathbf{z} \sim p_z(\mathbf{z}) \rightarrow \mathbf{x} = G(\mathbf{z}) \quad (2.6)$$

où θ_g représente les paramètres du générateur, $p_z(z)$ est la distribution du bruit (généralement uniforme ou normale), et x est la donnée synthétique générée.

Le générateur apprend à partir des erreurs du discriminateur, en améliorant constamment la qualité des données générées. En d'autres termes, il essaie de maximiser la probabilité que le discriminateur classifie ses sorties comme étant des données réelles.

Discriminateur

Le réseau discriminateur reçoit soit une donnée réelle x soit une donnée générée $G(z)$. Il sort une probabilité $D(x)$ ou $D(G(z))$ indiquant si l'entrée est réelle ou générée. Mathématiquement, le discriminateur peut être représenté comme une fonction $D : X \rightarrow [0, 1]$, où $D(x)$ représente la probabilité que x soit une donnée réelle.

$$D(\mathbf{x}; \theta_d) : \mathbf{x} \rightarrow [0, 1] \quad (2.7)$$

où θ_d représente les paramètres du discriminateur. Le discriminateur est entraîné pour maximiser la probabilité d'assigner la bonne étiquette aux échantillons réels et générés. Il essaie de minimiser la probabilité d'être trompé par les fausses données produites par le générateur.

Le discriminateur est, en essence, un classificateur binaire qui essaie de distinguer entre les données authentiques et celles générées.

Fonction de Perte

Les fonctions de perte pour le générateur et le discriminateur sont définies comme suit :

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z})))] \quad (2.8)$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_z} [\log D(G(\mathbf{z}))] \quad (2.9)$$

Où :

\mathcal{L}_D : fonction de perte du discriminateur

\mathcal{L}_G : fonction de perte du générateur

x : donnée réelle tirée de la distribution p_{data}

z : vecteur de bruit tiré de la distribution p_z

$D(x)$: probabilité estimée par le discriminateur que x soit une donnée réelle

$G(z)$: donnée synthétique générée à partir du bruit z

L'objectif du générateur est de minimiser \mathcal{L}_G , tandis que le discriminateur cherche à minimiser \mathcal{L}_D . Plus formellement, l'objectif est de résoudre le problème de minimax suivant :

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z})))] \quad (2.10)$$

Entraînement des GANs

L'entraînement des réseaux antagonistes génératifs (GANs) implique une série d'étapes répétitives où le générateur et le discriminateur sont mis à jour tour à tour pour améliorer leurs performances respectives. Le processus se déroule comme suit :

L'entraînement des GANs se fait par les étapes suivantes :

1. Tirer un échantillon de bruit z de la distribution p_z .
2. Générer une donnée synthétique $G(z)$ à partir du générateur.

3. Tirer un échantillon de données réelles x de la distribution de données p_{data} .
4. Mettre à jour les poids du discriminateur D en minimisant la fonction de perte \mathcal{L}_D .
5. Tirer un nouvel échantillon de bruit z .
6. Mettre à jour les poids du générateur G en minimisant la fonction de perte \mathcal{L}_G .
7. Répéter les étapes ci-dessus pour un nombre prédéfini d’itérations.

En suivant ces étapes, les GANs sont entraînés pour générer des données synthétiques réalistes qui peuvent être utilisées dans diverses applications, telles que la génération d’images, la super-résolution d’images et la synthèse de texte.

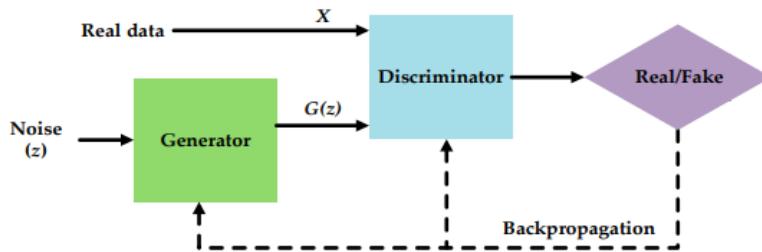


FIG. 2.2 : Un modèle de réseau générateur adversaire (GAN) [Jie FENG et al. 2020].

2.3.3 Applications des GANs

Les GANs ont une variété d’applications dans différents domaines :

Génération d’Images

Les GANs sont largement utilisés pour générer des images réalistes. Ils ont été appliqués avec succès dans la création d’images de haute qualité, la génération de visages humains, et même la création artistique. Par exemple, les GAN peuvent générer des images de personnes qui n’existent pas en apprenant les caractéristiques des visages humains à partir de vastes bases de données d’images.

Super-Résolution d’Images

Les GANs sont également utilisés pour la super-résolution d’images, c’est-à-dire augmenter la résolution d’une image basse résolution. Le modèle SRGAN (Super-Resolution GAN) est un exemple où les GANs sont utilisés pour produire des images de haute résolution à partir d’images de basse résolution, améliorant ainsi les détails visuels et la clarté.

Synthèse de Texte et Traduction Automatique

Dans le traitement du langage naturel (NLP), les GANs sont appliqués à la génération de texte et à la traduction automatique. Les modèles basés sur les GAN peuvent générer des phrases cohérentes et contextuellement appropriées, imitant le style et le contenu des textes d'entraînement. Cela est particulièrement utile pour les applications comme les chatbots, la création de contenu textuel, et les systèmes de traduction.

StyleGAN : This Person Does Not Exist

StyleGAN, ou Style-Based Generator Architecture for Generative Adversarial Networks, est un modèle génératif avancé développé par les chercheurs de NVIDIA[KARRAS et al. 2019]. Il représente une avancée significative dans la génération d'images de haute qualité et photoréalistes, y compris les visages de personnes qui n'existent pas. StyleGAN a diverses applications, y compris dans l'industrie du divertissement, la réalité virtuelle et la recherche académique.

This Person Does Not Exist : Un exemple frappant de l'application de StyleGAN est le site web "This Person Does Not Exist" [P. WANG 2019] . Ce site utilise un modèle StyleGAN pour générer de manière aléatoire des visages de personnes qui n'existent pas à chaque rechargement de la page.



FIG. 2.3 : Quelques images générées par le modèle StyleGAN sur le site 'This Person Does Not Exist'[P. WANG 2019].

En résumé, les GANs sont des outils puissants et polyvalents dans le domaine de l'apprentissage machine, avec des applications qui vont de la génération d'images à la traduction automatique. Leur capacité à apprendre et à générer des données réalistes ouvre de nombreuses perspectives dans divers domaines.

2.4 Diffusion models

2.4.1 Introduction

Les modèles de diffusion, également connus sous le nom de diffusion models, représentent une avancée significative dans le domaine des modèles génératifs au sein de l'apprentissage automatique. Émergents de manière proéminente ces dernières années, ces modèles ont attiré l'attention pour leur remarquable capacité à générer des données synthétiques de haute qualité, les positionnant comme une alternative prometteuse aux techniques plus établies telles que les Réseaux Antagonistes Génératifs (GANs) et les Autoencodeurs Variationnels (VAEs) [DHARIWAL et al. 2021].

La fondation conceptuelle des modèles de diffusion s'inspire des processus de diffusion physique, où les particules migrent des régions de haute concentration vers des zones de moindre concentration [HO et al. 2020]. Ces modèles sont également basés sur les chaînes de Markov comme illustré dans la figure (2.4), un concept clé en probabilité qui décrit une série de transitions d'état où chaque état dépend uniquement de l'état précédent. Dans le contexte des données, ce principe est exploité pour affiner progressivement les données bruitées en représentations réalistes [YANG SONG 2022].

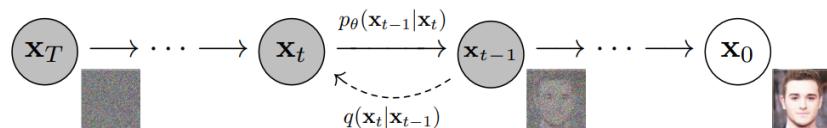


FIG. 2.4 : Chaîne de Markov du processus de diffusion [HO et al. 2020].

2.4.2 Fonctionnement des Modèles de Diffusion

Les modèles de diffusion utilisent un processus en deux étapes cruciales : le bruitage (noising, également appelé Forward Process) et le débruitage (denoising, ou Reverse Process). Ces étapes sont essentielles pour transformer des données réelles en une distribution de bruit et inversement. Le Forward Process consiste à ajouter progressivement du bruit aux données réelles, les transformant ainsi en une séquence de données de plus en plus bruitées. Ensuite, le Reverse Process inverse ce processus en éliminant le bruit étape par étape, recréant des données réalistes à partir du bruit.

Pour effectuer le débruitage, un réseau de neurones est utilisé. Ce réseau de neurones est entraîné à détecter et à éliminer le bruit à chaque étape du Reverse Process. Il apprend à prédire l'état précédent des données à partir de l'état actuel bruité, permettant ainsi de reconstruire progressivement les données originales. Cette méthode en deux phases, combinée à l'utilisation d'un réseau de neurones pour le débruitage, permet de modéliser et de générer des données synthétiques de haute qualité de manière efficace et contrôlée.

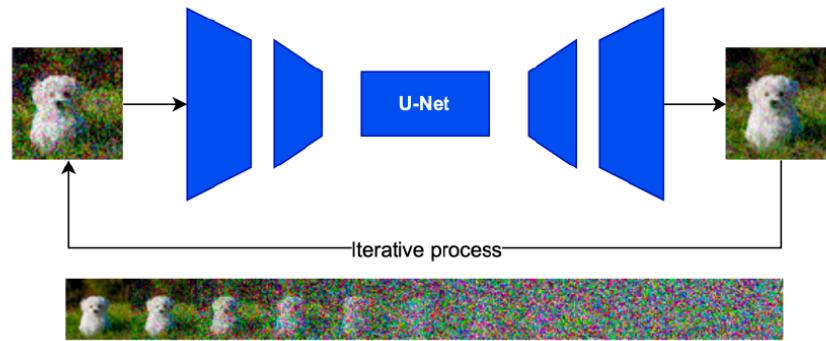


FIG. 2.5 : Le processus d’entraînement d’un UNet pour prédire le bruit ajouté à l’image. [NICHOL et al. 2021].

Les modèles de diffusion utilisent un processus en deux étapes : le bruitage (noising, également appelé *Forward Process*) et le débruitage (denoising, ou *Reverse Process*). Ces étapes sont essentielles pour transformer des données réelles en une distribution de bruit et inversement.

Forward Process (Bruitage)

L’étape de bruitage ajoute progressivement du bruit gaussien aux données réelles. Ce processus est modélisé comme une chaîne de Markov où chaque état x_t dépend uniquement de l’état précédent x_{t-1} .

La transition conditionnelle pour le bruitage est donnée par :

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (2.11)$$

où :

x_t : état des données à l’étape t

β_t : taux de bruitage (un hyperparamètre ou une fonction du temps)

\mathcal{N} : distribution normale gaussienne

\mathbf{I} : matrice identité

La séquence complète des T étapes de bruitage est décrite par :

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \quad (2.12)$$

Explication du Forward Process :

- À chaque étape t , une petite quantité de bruit gaussien est ajoutée à x_{t-1} , produisant x_t .
- La moyenne de la distribution est $\sqrt{1 - \beta_t} x_{t-1}$, indiquant que la contribution des données d’origine diminue progressivement.
- La variance est β_t , contrôlant la quantité de bruit ajouté.

Reverse Process (Débruitage)

L'étape de débruitage consiste à inverser le processus de bruitage pour transformer le bruit gaussien en données réalistes. Cela se fait en utilisant un modèle génératif pour approximer la distribution inverse.

La transition conditionnelle pour le débruitage est donnée par :

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t)) \quad (2.13)$$

où :

p_{θ} : distribution paramétrée par les poids du modèle θ

μ_{θ} : moyenne apprise par le modèle

Σ_{θ} : covariance apprise par le modèle

Explication du Reverse Process :

- Le modèle est entraîné pour prédire x_{t-1} à partir de x_t .
- La moyenne $\mu_{\theta}(x_t, t)$ est une fonction paramétrée par θ , dépendant de x_t et du temps t .
- La covariance $\Sigma_{\theta}(x_t, t)$ représente l'incertitude du modèle.

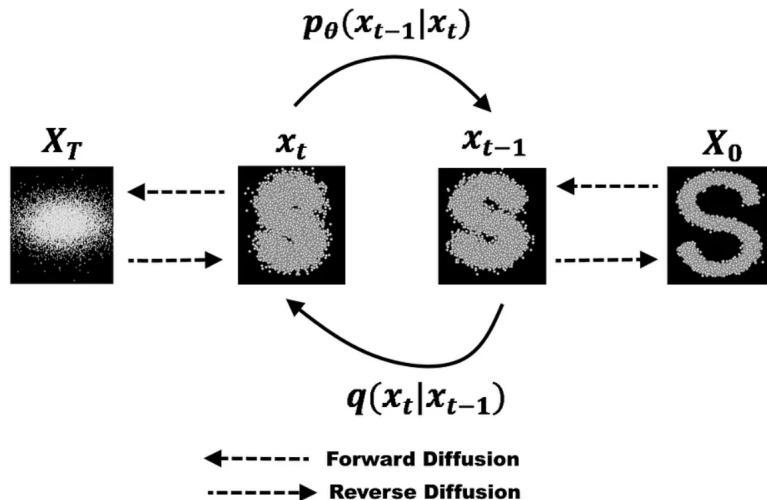


FIG. 2.6 : Illustration des Processus de Diffusion : Forward Process et Reverse Process.

Objectif d'Entraînement : L'objectif est de minimiser la divergence de Kullback-Leibler (KL) entre la distribution réelle q et la distribution générée par le modèle p_{θ} :

$$\mathcal{L} = \mathbb{E}_q \left[\sum_{t=1}^T D_{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \right] \quad (2.14)$$

2.4.3 Applications des Modèles de Diffusion

Les modèles de diffusion ont trouvé des applications dans divers domaines grâce à leur capacité à générer des données synthétiques de haute qualité. Voici quelques-unes des applications les plus remarquables :

- **Génération d'images** : Les modèles de diffusion sont largement utilisés pour la génération d'images réalistes à partir de bruit aléatoire. Ils permettent de créer des images de haute qualité qui peuvent être utilisées dans le domaine de l'art, du design, et même dans la production de contenu multimédia.
- **Synthèse vocale** : Ces modèles sont également appliqués dans la génération de voix synthétiques, offrant des améliorations significatives dans la clarté et la naturalité de la voix produite, utilisée dans les assistants vocaux et les technologies de lecture automatique.
- **Création de vidéos** : En transformant des séquences de bruit en vidéos cohérentes, les modèles de diffusion peuvent générer des animations et des vidéos synthétiques, ouvrant de nouvelles possibilités dans le divertissement et la publicité.
- **Traitements et amélioration d'images** : Les modèles de diffusion peuvent être utilisés pour restaurer des images bruitées, améliorer la résolution des images, et effectuer des transformations d'image avancées telles que la colorisation et le changement de style.
- **Séries temporelles** : On peut aussi utiliser les modèles de diffusion pour d'autres types de données comme les séries temporelles. Ils peuvent aider à la prévision, à la détection d'anomalies et à la génération de nouvelles séries temporelles synthétiques [YANG et al. 2023].

Parmi les applications ayant connu un succès notable, on trouve :

- **Stable Diffusion** : Un modèle de diffusion qui a gagné en popularité pour sa capacité à générer des images d'une qualité exceptionnelle et son utilisation dans divers projets artistiques et commerciaux. Stable Diffusion est particulièrement apprécié pour sa flexibilité et sa robustesse, permettant aux artistes et aux créateurs de générer des œuvres d'art numériques impressionnantes avec un niveau de détail et de réalisme sans précédent. En outre, il est utilisé dans des industries telles que le design graphique, la publicité et même la production cinématographique, démontrant sa vaste applicabilité commerciale.
- **DALL-E de OpenAI** : Ce modèle est capable de créer des images à partir de descriptions textuelles, révolutionnant ainsi la manière dont nous concevons et interagissons avec les images générées par l'intelligence artificielle. DALL-E peut interpréter des descriptions textuelles complexes et générer des images correspondantes, ce qui ouvre de nouvelles possibilités en matière de création de contenu visuel. Par exemple, il peut créer des images de concepts ou d'objets inexistants basés uniquement sur des descriptions verbales, ce qui est extrêmement utile pour le prototypage, l'illustration de concepts et l'innovation créative. DALL-E a été salué pour

sa capacité à comprendre et à visualiser des idées abstraites, rendant la technologie accessible et utile pour les designers, les artistes et les innovateurs.



FIG. 2.7 : Exemple d'images générée par DALL-E2. [RAMESH et al. 2022].

Ces applications démontrent la polyvalence et l'efficacité des modèles de diffusion, faisant d'eux des outils indispensables dans l'arsenal de l'intelligence artificielle moderne.

2.4.4 Conclusion

En conclusion, les modèles de diffusion représentent une avancée significative dans le domaine des modèles génératifs au sein de l'apprentissage automatique. En s'appuyant sur un processus en deux étapes, le Forward Process et le Reverse Process, ces modèles parviennent à transformer des données réelles en une distribution de bruit et à les reconvertis en données réalistes de haute qualité. L'utilisation d'un réseau de neurones pour le débruitage améliore encore l'efficacité et la précision du processus. Grâce à leur capacité à générer des données synthétiques fiables, les modèles de diffusion se positionnent comme une alternative prometteuse et robuste aux techniques traditionnelles telles que les Réseaux Antagonistes Génératifs et les Autoencodeurs Variationnels. Leur évolution continue et leur adoption croissante témoignent de leur potentiel immense et de leur pertinence dans divers domaines d'application de l'intelligence artificielle.

Chapitre 3

Maintenance prédictive des moteurs électriques

3.1 Introduction

La maintenance prédictive est devenue une approche incontournable pour la gestion efficace des équipements industriels, en particulier les moteurs électriques. Avec l'avènement de l'Industrie 4.0 et l'intégration croissante de l'Internet des Objets (IoT) et de l'intelligence artificielle (IA) dans les processus industriels, la maintenance prédictive permet d'anticiper les défaillances potentielles avant qu'elles ne se produisent, assurant ainsi une continuité opérationnelle optimale et une réduction significative des coûts de maintenance [MROZEK et al. 2023].

Les moteurs électriques, étant au cœur de nombreuses applications industrielles, représentent des composants critiques dont la fiabilité et l'efficacité doivent être constamment surveillées. La maintenance prédictive pour ces moteurs repose sur l'analyse en temps réel de diverses données opérationnelles telles que les vibrations, les températures, les courants électriques et les niveaux de bruit. Grâce à des capteurs avancés et à des algorithmes sophistiqués, il est possible de détecter des anomalies subtiles et d'identifier les signes précurseurs de défaillances mécaniques ou électriques [BENTIVOGLI et al. 2023].

L'adoption de la maintenance prédictive apporte plusieurs avantages majeurs. Elle permet non seulement d'améliorer la durée de vie des moteurs électriques, mais aussi de planifier les interventions de maintenance de manière plus stratégique, évitant ainsi les arrêts imprévus et minimisant les perturbations de la production. De plus, cette approche contribue à une gestion plus durable des ressources en réduisant les déchets et en optimisant l'utilisation de l'énergie.

Dans ce chapitre, nous explorerons en détail les principes fondamentaux de la maintenance prédictive appliquée aux moteurs électriques. Nous examinerons les technologies et outils utilisés pour la collecte et l'analyse des données, ainsi que les différentes méthodes d'interprétation des signaux pour prévoir les défaillances. Enfin, nous discuterons des défis et des perspectives d'avenir de cette approche, mettant en lumière son rôle crucial dans la transformation numérique de l'industrie.

3.2 Types de Maintenance

3.2.1 Maintenance Corrective

La maintenance corrective est réalisée après la détection d'une panne ou d'une défaillance. Son objectif est de remettre l'équipement en état de fonctionner en corrigeant le défaut[SEZDI 2019].

La maintenance corrective intervient généralement après l'échec de l'équipement, lorsque celui-ci cesse de fonctionner comme prévu. Elle est souvent utilisée dans les industries où le coût de l'arrêt non planifié est moins critique ou où les équipements ne peuvent pas être surveillés en continu. Par exemple, dans certaines installations de production manufacturière, les interventions correctives peuvent être tolérées si les pièces de rechange sont facilement disponibles et si les réparations peuvent être effectuées rapidement. Cette méthode est également courante dans les petites entreprises où les ressources pour une maintenance plus proactive peuvent être limitées.

3.2.2 Maintenance Préventive

La maintenance préventive est effectuée régulièrement selon un calendrier prédéfini ou en fonction du temps ou du nombre de cycles d'utilisation, indépendamment de l'état de l'équipement. Elle vise à réduire la probabilité de défaillance [SEZDI 2019].

La maintenance préventive repose sur des inspections et des services réguliers pour éviter les pannes inattendues. Cette méthode est largement adoptée dans les industries telles que l'aérospatiale, l'automobile et les centrales électriques où la fiabilité et la sécurité sont critiques. Par exemple, dans l'industrie aéronautique, les avions subissent des inspections régulières et des entretiens programmés pour garantir la sécurité des vols et prévenir les pannes en vol. De même, dans les centrales électriques, des programmes de maintenance préventive stricts sont mis en place pour assurer un fonctionnement continu et sûr des générateurs et des turbines.

3.2.3 Maintenance Prédictive

La maintenance prédictive repose sur la surveillance en temps réel et l'analyse des données de l'équipement pour anticiper les pannes avant qu'elles ne surviennent. Elle utilise des techniques telles que la vibration, l'analyse d'huile, l'imagerie thermique, et les données de performance [BENTIVOGLI et al. 2023 ; SANGEETHA et al. 2017].

La maintenance prédictive vise à identifier les signes avant-coureurs de défaillances potentielles en utilisant des technologies avancées. Les capteurs IoT, les analyses de données et les algorithmes d'apprentissage automatique sont employés pour surveiller les conditions des équipements en temps réel. Cette approche est particulièrement utile dans les industries telles que le pétrole et le gaz, la production d'énergie et la fabrication de haute précision. Par exemple, dans les centrales éoliennes, les systèmes de maintenance prédictive surveillent les vibrations et les températures des composants critiques pour prévenir les pannes coûteuses et prolonger la durée de vie des équipements. Dans l'industrie ma-

nufacturière, elle permet de maintenir la qualité des produits et d'optimiser les cycles de production en réduisant les temps d'arrêt imprévus.

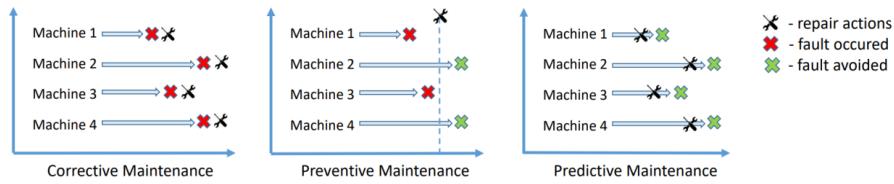


FIG. 3.1 : Stratégies de maintenance montrant les différentes étapes du processus de réparation avant et après la défaillance potentielle de l'équipement. [MROZEK et al. 2023].

Type de Maintenance	Avantages	Inconvénients
Corrective	Simple à planifier, nécessite moins de surveillance continue	Arrêts imprévus, coûts élevés de réparation d'urgence, et perturbations importantes de la production
Préventive	Réduit les risques de pannes imprévues, améliore la fiabilité de l'équipement	Coûts supplémentaires en raison de l'entretien d'équipements encore en bon état, nécessite une planification rigoureuse
Préditive	Permet de planifier les interventions de manière optimale, minimise les arrêts non planifiés, prolonge la durée de vie des équipements, et optimise les coûts	Nécessite des investissements initiaux en capteurs et en technologie de surveillance, ainsi que des compétences analytiques spécialisées

TAB. 3.1 : Comparaison entre les Types de Maintenance

3.3 Maintenance prédictive pour les moteurs électriques

La maintenance prédictive des moteurs électriques est une approche proactive qui vise à anticiper les pannes et à optimiser la performance des équipements. Contrairement à la maintenance réactive, qui intervient après la survenue d'un problème, ou à la maintenance préventive, qui suit un calendrier fixe, la maintenance prédictive s'appuie sur la surveillance continue de l'état des machines et l'analyse de données en temps réel.

Cette stratégie utilise des technologies avancées telles que l'Internet des objets (IoT), l'intelligence artificielle (IA) et l'analyse de données pour collecter et interpréter des informations sur les vibrations, la température, le courant et d'autres paramètres opérationnels

des moteurs électriques. Grâce à ces données, il est possible de détecter les signes avant-coureurs de défaillances potentielles, comme l'usure des roulements, les déséquilibres de rotor ou les problèmes d'isolation.

L'objectif principal de la maintenance prédictive est d'améliorer la fiabilité et la durée de vie des moteurs électriques tout en réduisant les coûts de maintenance et les interruptions de service. En anticipant les problèmes avant qu'ils ne deviennent critiques, les entreprises peuvent planifier les interventions de maintenance de manière plus efficace, minimisant ainsi les temps d'arrêt imprévus et les coûts associés.

En somme, la maintenance prédictive représente une avancée significative dans la gestion des actifs industriels, offrant une approche plus intelligente et efficiente pour la maintenance des moteurs électriques.

3.3.1 Moteurs électriques

Les moteurs électriques sont des dispositifs essentiels qui convertissent l'énergie électrique en énergie mécanique, jouant un rôle crucial dans de nombreux aspects de la vie quotidienne et industrielle. Utilisés dans divers secteurs tels que la fabrication, le transport et les applications domestiques, les moteurs électriques permettent un contrôle précis, indispensable pour les applications industrielles nécessitant une régulation rigoureuse de la vitesse et du couple.

Parmi les différents types de moteurs électriques, les moteurs asynchrones, ou moteurs à induction, occupent une place prépondérante. Inventés par Nikola Tesla, ces moteurs sont largement utilisés en raison de leur simplicité, de leur robustesse et de leur coût relativement bas. Un moteur asynchrone fonctionne sur le principe de l'induction électromagnétique : lorsqu'un courant alternatif traverse les bobines du stator, un champ magnétique tournant est créé, induisant un courant dans le rotor. Ce courant induit génère à son tour un champ magnétique qui interagit avec celui du stator, produisant un couple qui fait tourner le rotor.

Ces moteurs sont utilisés dans une vaste gamme d'industries et d'applications. Dans l'industrie manufacturière, ils entraînent des convoyeurs, des broyeurs et des compresseurs. Dans les systèmes de traction, ils sont essentiels pour les trains et les véhicules électriques. De plus, dans les services publics, ils sont utilisés pour les pompes, les ventilateurs et les systèmes de ventilation et de climatisation. Les moteurs asynchrones se distinguent par leur fiabilité, leur capacité à fonctionner dans des conditions difficiles et leur maintenance relativement facile, ce qui en fait un choix privilégié pour de nombreuses applications industrielles et domestiques.

Moteurs Asynchrones

Il existe deux types de moteurs asynchrones : monophasés et triphasés. Dans cette section, nous aborderons spécifiquement les moteurs triphasés. Dans la figure (3.2), les principaux composants d'un moteur asynchrone triphasé, également appelé machine à induction, sont présentés. Voici une description de chaque composant :

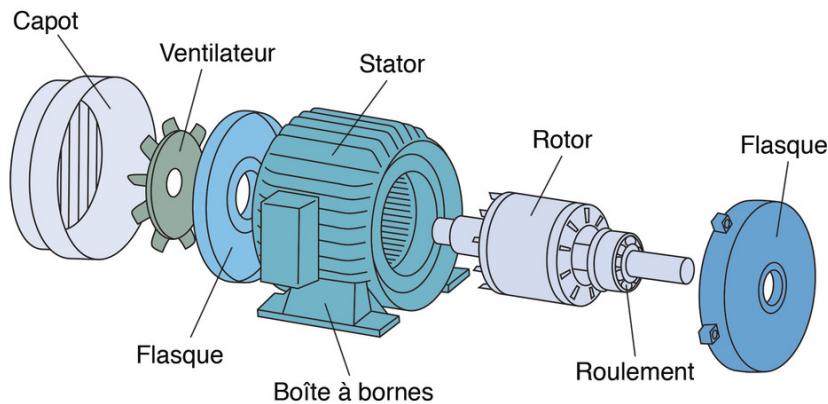


FIG. 3.2 : Principaux Composants d'un moteur asynchrone triphasé (Machine à Induction).

- **Capot** : Il s'agit de la couverture externe du moteur qui protège les composants internes de la poussière, de l'humidité et d'autres contaminants.
- **Ventilateur** : Situé à côté du capot, le ventilateur assure la ventilation du moteur en refroidissant les composants internes pour éviter la surchauffe.
- **Flasque** : Les flasques sont des plaques situées aux extrémités du moteur, qui maintiennent les autres composants en place et assurent l'alignement correct de l'axe du rotor.
- **Stator** : C'est la partie fixe du moteur. Il est constitué de bobines de fil de cuivre qui génèrent un champ magnétique rotatif lorsque le courant triphasé y circule.
- **Boîte à bornes** : Située sur le stator, elle abrite les connexions électriques du moteur, permettant de connecter le moteur à l'alimentation électrique.
- **Rotor** : C'est la partie rotative du moteur qui tourne sous l'effet du champ magnétique généré par le stator. Il est connecté à l'axe et transfère l'énergie mécanique.
- **Roulement** : Les roulements soutiennent le rotor et permettent son mouvement rotatif en minimisant la friction.

Défauts des Moteurs Asynchrones

Les moteurs asynchrones, également appelés machines à induction, peuvent présenter divers défauts mécaniques et électriques. Ces défauts peuvent affecter les performances et la durée de vie du moteur. Voici une description des principaux défauts mécaniques et électriques :

Défauts Mécaniques

Défaut d'excentricité de l'entrefer : L'excentricité de l'entrefer se produit lorsque l'espace entre le stator et le rotor n'est pas uniforme. Cela peut être causé par un désalignement, une usure inégale des roulements ou des défauts de fabrication. Ce défaut peut entraîner des vibrations excessives, des bruits anormaux et une usure prématurée des composants du moteur.

Analyse des dommages aux roulements : Les roulements sont essentiels pour le bon fonctionnement du rotor. Les dommages aux roulements peuvent être causés par une lubrification insuffisante, des contaminants, une surcharge ou une mauvaise installation. Les symptômes de dommages aux roulements incluent des vibrations élevées, des bruits de grincement ou de cliquetis, et une augmentation de la température de fonctionnement.

Défauts Électriques

Court-circuit entre spires : Un court-circuit entre spires se produit lorsque l'isolation entre les enroulements d'une bobine du stator est endommagée, permettant au courant de circuler directement entre les spires. Ce défaut peut entraîner une surchauffe localisée, une diminution de l'efficacité du moteur et, dans les cas graves, une défaillance complète du moteur.

Barre de rotor cassée : Les barres de rotor sont des composants clés du rotor de la cage d'écureuil dans un moteur asynchrone. Une barre de rotor cassée peut se produire en raison de contraintes mécaniques répétées, de surcharges ou de défauts de fabrication. Ce défaut peut provoquer un déséquilibre du rotor, des vibrations excessives et une diminution des performances du moteur.

3.4 Traitement de Signal Pour les moteurs électriques

Un signal est une représentation physique ou une mesure de certaines caractéristiques d'un système, généralement en fonction du temps. Dans le contexte des moteurs électriques, les signaux peuvent inclure des courants, des tensions, des vitesses de rotation, des températures, des vibrations, et bien d'autres paramètres. Ces signaux peuvent être capturés par des capteurs installés sur ou à proximité du moteur.

L'analyse des signaux de courant et de tension de sortie est cruciale pour la maintenance prédictive des moteurs électriques. Des techniques telles que l'analyse des harmoniques, l'analyse du spectre de fréquence et la surveillance des formes d'onde permettent de détecter les anomalies et de diagnostiquer les défauts avant qu'ils ne causent des pannes majeures. Par exemple, un déséquilibre dans les courants de phase peut indiquer un défaut mécanique comme une excentricité de l'entrefer, tandis qu'une augmentation des harmoniques peut signaler un court-circuit entre spires.

Signaux Triphasés dans les Moteurs Électriques

Les moteurs électriques triphasés utilisent des signaux électriques triphasés pour fonctionner efficacement. Ces signaux, qui consistent en trois courants alternatifs déphasés de 120 degrés chacun, sont essentiels pour la création d'un champ magnétique rotatif dans le stator, qui à son tour entraîne la rotation du rotor.

Signaux d'entrée

Les signaux d'entrée d'un moteur électrique triphasé sont constitués de courants et de tensions triphasés. Chaque phase est décalée de 120 degrés par rapport aux autres, permettant une distribution équilibrée de la puissance et réduisant les vibrations et les pertes. Les courants et tensions triphasés d'entrée doivent être équilibrés et stables pour assurer un fonctionnement optimal du moteur. Des déséquilibres ou des variations peuvent indiquer des problèmes dans l'alimentation ou des défauts internes.

Signaux de sortie

Les signaux de sortie d'un moteur électrique, principalement les courants et les tensions mesurés aux bornes du moteur, peuvent fournir des informations précieuses sur l'état de santé du moteur. Un moteur en bon état génère des signaux de sortie avec un certain motif ou "pattern" caractéristique. Ce motif est souvent analysé en termes de formes d'onde, d'harmoniques et de spectres de fréquence.

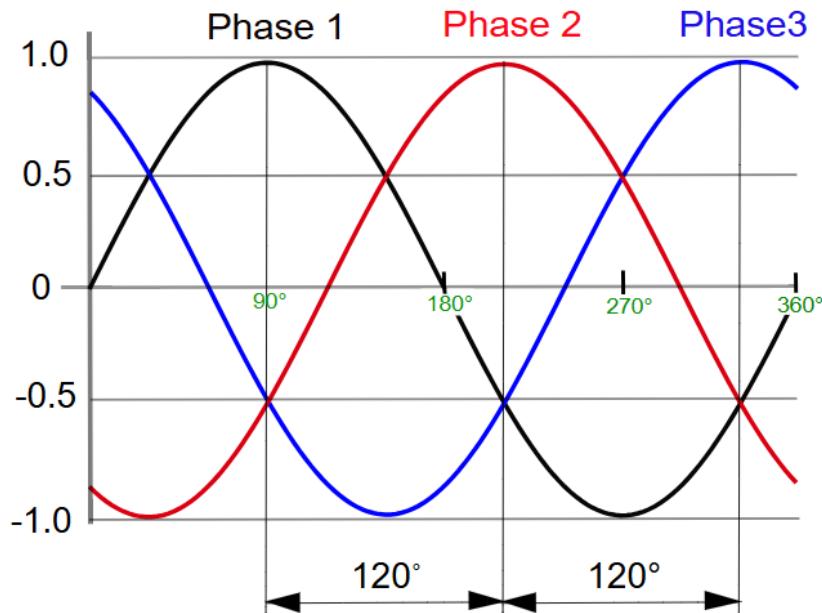


FIG. 3.3 : Principaux Composants d'un moteur asynchrone triphasé (Machine à Induction).

Motor Current Signature Analysis (MCSA)

L'analyse des signaux de courant et de tension de sortie est cruciale pour la maintenance prédictive des moteurs électriques. Une technique particulièrement efficace pour cette analyse est la Motor Current Signature Analysis (MCSA).

La MCSA est une méthode de surveillance et de diagnostic basée sur l'analyse des courants du moteur. En mesurant et en analysant les signatures de courant, il est possible de détecter des défauts internes et externes du moteur [BONET-JARA et al. 2023]. La MCSA peut identifier des problèmes tels que :

- **Défauts mécaniques** : Excentricité de l'entrefer, désalignement, déséquilibre.
- **Défauts électriques** : Court-circuit entre spires, barres de rotor cassées, défauts d'isolation.
- **Problèmes d'alimentation** : Déséquilibre de phase, distorsions harmoniques.

La MCSA utilise des techniques d'analyse de spectre de fréquence pour identifier les composantes spécifiques du signal de courant qui sont associées à différents types de défauts. Par exemple, un déséquilibre du rotor peut générer des fréquences spécifiques qui apparaissent dans le spectre de courant. En détectant ces fréquences, il est possible de diagnostiquer le défaut avant qu'il ne cause une panne majeure.

Avantages de la MCSA

L'utilisation de la MCSA pour la surveillance des moteurs électriques présente plusieurs avantages :

- **Détection précoce des défauts** : Identification des problèmes avant qu'ils ne deviennent critiques.
- **Maintenance proactive** : Planification des interventions de maintenance basées sur l'état réel du moteur.
- **Réduction des coûts** : Minimisation des temps d'arrêt imprévus et des réparations coûteuses.
- **Amélioration de la fiabilité** : Augmentation de la durée de vie des moteurs et amélioration de leur performance globale.

la surveillance continue des signaux triphasés et l'utilisation de techniques comme la MCSA sont essentielles pour assurer le bon fonctionnement et la longévité des moteurs électriques. L'analyse proactive des courants et des tensions permet de détecter les problèmes potentiels à un stade précoce, facilitant ainsi une intervention rapide et réduisant les coûts de maintenance.

3.5 Données : Séries Temporelles

Dans de nombreux contextes, les données de séries temporelles peuvent être considérées comme des signaux, en particulier lorsque les points de données représentent des mesures prises à intervalles de temps réguliers. Par exemple, les relevés de température quotidienne constituent une série temporelle qui peut être analysée comme un signal pour identifier les schémas, les tendances et les anomalies [BROPHY et al. 2023]. Il existe deux types de séries temporelles : discrètes et continues.

Les séries temporelles discrètes, caractérisées par des points de données espacés à des intervalles réguliers, sont couramment utilisées dans de nombreuses disciplines pour faciliter l'analyse statistique et la prévision. Elles permettent une compréhension claire des tendances à long terme et des variations saisonnières, et sont particulièrement utiles dans des domaines tels que la climatologie, l'économie et les sciences sociales [BROPHY et al. 2023].

À l'opposé, les séries temporelles continues, où les données sont collectées à des intervalles infinitésimaux, offrent une représentation plus détaillée et précise des phénomènes étudiés. Cette approche est cruciale dans des domaines tels que le traitement du signal, la physique et l'ingénierie, où une résolution temporelle fine est nécessaire pour capturer les dynamiques rapides et les variations subtiles des systèmes [BROPHY et al. 2023].

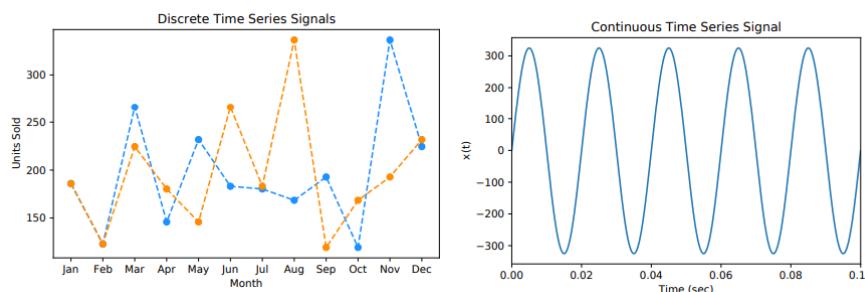


FIG. 3.4 : Exemples de tracés de séries temporelles discrètes (à gauche) et continues (à droite) [BROPHY et al. 2023].

Catégories de Séries Temporelles

Il existe deux catégories de séries temporelles : univariée et multivariée [FAWAZ et al. 2019b].

Définition 1 : Série temporelle univariée

Une série temporelle univariée $X = [x_1, x_2, \dots, x_T]$ est un ensemble ordonné de valeurs réelles. La longueur de X est égale au nombre de valeurs réelles T .

$$X = [x_1, x_2, \dots, x_T] \quad (3.1)$$

où $x_i \in \mathbb{R}$ et $i = 1, 2, \dots, T$.

Par exemple, une série temporelle univariée de température prise toutes les heures

pendant une journée pourrait être représentée par : $X = [15.5, 16.0, 16.5, \dots, 20.0]$, où $T = 24$.

Définition 2 : Série temporelle multivariée

Une série temporelle multivariée de dimension M , notée $X = [X_1, X_2, \dots, X_M]$, se compose de M séries temporelles univariées différentes, chaque X_i étant un vecteur de \mathbb{R}^T .

$$X = [X_1, X_2, \dots, X_M] \quad (3.2)$$

où $X_i = [x_{i1}, x_{i2}, \dots, x_{iT}]$, avec $x_{ij} \in \mathbb{R}$ pour $i = 1, 2, \dots, M$ et $j = 1, 2, \dots, T$.

Par exemple, une série temporelle multivariée de dimension $M = 2$, comprenant la température et l'humidité relevées toutes les heures pendant une journée, pourrait être représentée par : $X = [X_1, X_2]$, où X_1 représente la température et X_2 représente l'humidité, chacun avec $T = 24$.

3.6 Conclusion

La maintenance prédictive des moteurs électriques représente une avancée significative dans la gestion des actifs industriels, offrant une approche proactive qui vise à anticiper les pannes et à optimiser la performance des équipements. Contrairement à la maintenance réactive, qui intervient après la survenue d'un problème, et à la maintenance préventive, qui suit un calendrier fixe, la maintenance prédictive repose sur la surveillance continue de l'état des machines et l'analyse de données en temps réel.

Cette stratégie utilise des technologies avancées telles que l'Internet des objets (IoT), l'intelligence artificielle (IA) et l'analyse de données pour collecter et interpréter des informations critiques sur les moteurs électriques, permettant ainsi de détecter les signes avant-coureurs de défaillances potentielles. L'objectif principal de cette approche est d'améliorer la fiabilité et la durée de vie des moteurs électriques tout en réduisant les coûts de maintenance et les interruptions de service.

En s'appuyant sur des techniques comme la Motor Current Signature Analysis (MCSA), la maintenance prédictive permet une détection précoce des défauts, une planification proactive des interventions de maintenance, et une optimisation des coûts. Cela se traduit par une minimisation des temps d'arrêt imprévus et une amélioration globale de la performance des moteurs.

Enfin, l'analyse des séries temporelles, qu'elles soient discrètes ou continues, joue un rôle crucial dans ce processus en fournissant des données détaillées et précises pour l'analyse des signaux et l'identification des anomalies. La distinction entre séries temporelles univariées et multivariées enrichit encore cette approche, permettant une analyse plus complète et intégrée des différentes variables opérationnelles.

Ainsi, la maintenance prédictive se positionne comme une composante essentielle de la gestion moderne des moteurs électriques, offrant des avantages significatifs en termes de fiabilité, de coût et de performance.

partie II

Contribution

Chapitre 4

Organisme d'accueil

4.1 Introduction

Dans ce chapitre, nous présenterons l'organisme qui nous a accueillis pour la réalisation de notre projet de fin d'études. Nous commencerons par une brève introduction de l'entreprise, en mettant en évidence son domaine d'activité et son positionnement sur le marché. Ensuite, nous détaillerons les différents services proposés par l'organisme, ainsi que son organigramme organisationnel. Cette présentation nous permettra de mieux comprendre l'environnement dans lequel notre projet s'est déroulé et les contraintes spécifiques auxquelles nous avons été confrontés.

4.2 Présentation de l'organisme d'accueil

STIE, ou Schneider Toshiba Inverter Europe SAS, est un département relié à Schneider Electric. Ce département offre une gamme complète d'équipements électroniques industriels. Il propose des inverseurs polyvalents, des variateurs de vitesse industriels, des composants de contrôle, et des produits de distribution électrique. STIE dispose également de laboratoires équipés de divers matériels, tels que des moteurs électriques et leurs dérivés, pour répondre aux besoins de ses clients à travers le monde. STIE est principalement basé à Pacy-sur-Eure.

Schneider Electric est un leader mondial de la technologie industrielle avec une expertise de référence dans l'électrification, l'automatisation, et la digitalisation des industries intelligentes, des infrastructures résilientes, des centres de données durables, des bâtiments intelligents, et des maisons intuitives. L'entreprise mène la transformation numérique de la gestion de l'énergie et des automatismes dans les secteurs résidentiel, des bâtiments, des centres de données, des infrastructures et des industries. Présente dans plus de 115 pays, Schneider Electric est le leader incontesté de la gestion électrique, couvrant la moyenne tension, la basse tension, l'énergie sécurisée, et les systèmes d'automatismes. La société fournit des solutions d'efficacité intégrées qui associent gestion de l'énergie, automatismes, et logiciels. L'écosystème qu'elle a construit lui permet de collaborer sur sa plateforme ouverte avec une large communauté de partenaires, d'intégrateurs, et de dé-

veloppeurs pour offrir à ses clients à la fois contrôle et efficacité opérationnelle en temps réel. Son siège social est situé à Rueil-Malmaison, et la répartition géographique de son chiffre d'affaires est la suivante :

- France : 5,8%
- Europe de l'Ouest : 18,5%
- États-Unis : 27,9%
- Amérique du Nord : 4,2%
- Chine : 15,1%
- Asie-Pacifique : 15,2%
- Autres : 13,3%

4.3 Services

Schneider Electric propose une gamme étendue de produits et services innovants, notamment :

- **Produits d'automatisation et de contrôle** : Solutions complètes pour l'automatisation des processus et la gestion des systèmes de contrôle industriel.
- **Produits et systèmes basse tension** : Équipements pour la distribution électrique et la protection des installations basse tension.
- **Énergie solaire et stockage** : Solutions pour la production d'énergie solaire et le stockage d'énergie.
- **Distribution moyenne tension et automatisation du réseau** : Systèmes pour la distribution d'énergie moyenne tension et l'automatisation des réseaux électriques.
- **Énergie critique, refroidissement et racks** : Solutions pour la gestion de l'énergie critique, le refroidissement des infrastructures et les racks de serveur.

4.4 L'organigramme de Schneider Electric

L'organigramme de Schneider Electric représente la structure de l'entreprise et les relations hiérarchiques entre ses membres. Voici une description détaillée de la structure organisationnelle de Schneider Electric :

Direction Générale

À la tête de l'entreprise se trouve Peter Herweck, PDG. Il est responsable de la direction générale et de la prise de décisions stratégiques.

Équipe de Management

L'équipe de direction de Schneider Electric comprend plusieurs membres clés. Peter Herweck occupe le poste de PDG, apportant une vision stratégique globale à l'entreprise. Barbara Frie est responsable de la Proposition de Valeur Employé, tandis que Deepu Chawla occupe le poste de Vice-Président Senior. Chaque membre apporte une expertise unique et une vision stratégique à l'entreprise.

Département des Ressources Humaines

Le département des ressources humaines est supervisé par Perrine Colson, Partenaire Commerciale en Ressources Humaines. Elle est responsable de la gestion du personnel, du recrutement et de l'administration du personnel..

Département de R&D

Le département de Recherche et Développement (STIE) est dirigé par Allan Pierre, Directeur de la R&D globale. Il est responsable de l'innovation, de la conception, de l'évolution et de la maintenance des Variateurs de Vitesse et Démarreurs Progressifs chez Schneider Electric. Allan Pierre gère une équipe de plus de 300 professionnels, assurant l'alignement avec les objectifs stratégiques et les exigences des clients.

Équipe de Drive R&D

L'équipe de Drive R&D se concentre sur la recherche et le développement dans le domaine des drives, la partie qui contrôle les moteurs électriques. Cette équipe est dirigée par Al Kassem Jebia, avec des membres clés tels que Nicolas Henwood et Ali Tfaily avec qui je travaille, cette équipe contribue à l'avancement des technologies et des solutions innovantes pour les moteurs électriques.

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'organisme d'accueil qui nous a permis de réaliser notre projet de fin d'études. Nous avons fourni une description détaillée de l'entreprise, de ses activités principales et de son positionnement sur le marché. Nous avons également mis en évidence les différents services proposés par l'organisme, ainsi que son organigramme organisationnel.

Cette présentation nous a permis de mieux comprendre l'environnement dans lequel notre projet s'est déroulé et les contraintes spécifiques auxquelles nous avons été confrontés.

Chapitre 5

Conception

5.1 Introduction

Dans ce chapitre, nous aborderons la problématique centrale à laquelle ce projet tente de répondre ainsi que la solution proposée pour la résoudre. La problématique identifiée concerne la nécessité de disposer d'un jeu de données plus riche et plus diversifié, capable de couvrir une large gamme de moteurs électriques disponibles sur le marché. Cette diversité est essentielle pour développer des modèles de maintenance prédictive robustes et généralisables.

Pour pallier le manque de données réelles disponibles pour tous les types de moteurs, nous proposons l'utilisation de modèles génératifs. Ces modèles permettent de créer artificiellement des données supplémentaires, tout en assurant que celles-ci reflètent fidèlement les caractéristiques des données réelles. Nous présenterons en détail les modèles génératifs utilisés, ainsi que leurs architectures.

De plus, nous discuterons des types de données que nous cherchons à générer, en précisant l'architecture de données mise en place pour l'entraînement des modèles génératifs. Enfin, nous présenterons les méthodes d'évaluation employées pour mesurer la qualité des données générées, en s'assurant qu'elles sont suffisamment représentatives des données réelles et utiles pour les applications de maintenance prédictive.

5.2 Problématique et besoins

Dans le cadre de mon stage chez Schneider Electric, l'objectif final est de développer une solution de maintenance prédictive pour les moteurs électriques asynchrones. La maintenance prédictive a pour but de détecter les défauts potentiels avant que le moteur ne tombe en panne, en s'appuyant sur une approche basée sur le machine learning. Cette méthode nécessite l'utilisation d'un modèle de réseaux de neurones, qui doit être entraîné sur une grande quantité de données afin d'être performant.

Cependant, une problématique majeure se pose : il existe une multitude de marques et de modèles de moteurs électriques, chacun ayant des caractéristiques différentes. Pour que la maintenance prédictive soit efficace, il est essentiel de développer une solution qui soit

généralisable à tous les types de moteurs utilisés par les clients de Schneider. Toutefois, il est pratiquement impossible de collecter des données exhaustives sur tous les moteurs disponibles sur le marché, ce qui complique considérablement l'entraînement du modèle de machine learning.

Chez Schneider Electric, une des solutions techniques envisagées repose sur l'utilisation d'un dispositif appelé "drive" (illustré à la figure 5.1). Ce drive est connecté au moteur électrique et permet de contrôler des paramètres essentiels tels que le couple et la vitesse. De plus, il est capable de lire les signaux entrants et sortants du moteur asynchrone triphasé.

Le drive de Schneider est équipé d'un microcontrôleur, et l'idée est de déployer un modèle de classification directement sur ce microcontrôleur afin de détecter les défauts ainsi que les types de pannes dans un moteur asynchrone triphasé. Ce modèle, embarqué sur le microcontrôleur, devra être capable de fonctionner efficacement en temps réel pour identifier les défaillances potentielles et alerter les opérateurs avant qu'une panne majeure ne survienne.



FIG. 5.1 : Schneider Electric ATV930D30N4 AC Speed Drive, 3 HP.

Objectifs du stage :

- Génération de données d'un moteur en bon état (healthy)
- Génération de données d'un moteur en mauvais état (non healthy) ayant déjà subi un défaut
- Utilisation de modèles génératifs pour la détection des défauts

La finalité de notre projet est de développer un modèle génératif capable d'être entraîné sur les données réelles provenant de moteurs électriques. Une fois entraîné, ce modèle pourra générer des données qui, bien qu'aléatoires, seront similaires aux données d'origine. L'objectif est de garantir que ces nouvelles données restent fidèles aux caractéristiques des données réelles, tout en étant suffisamment distinctes pour éviter toute redondance exacte. Ce processus permet de simuler des scénarios variés et d'enrichir le jeu de données avec des exemples réalistes, sans pour autant reproduire à l'identique les enregistrements initiaux.

5.3 Notions

- **Couple et Vitesse**

Le couple et la vitesse sont deux paramètres clés qui caractérisent le fonctionnement d'un moteur électrique.

- **Couple (T)** : Le couple (Torque en anglais) est la force rotative produite par le moteur, exprimée en Newton-mètre (Nm). Il représente la capacité du moteur à produire une rotation et est directement lié à la charge que le moteur peut entraîner. Le couple est un indicateur crucial de la performance d'un moteur, surtout dans des conditions de charge variable.
- **Vitesse (S)** : La vitesse (Speed en anglais), généralement mesurée en tours par minute (tr/min), désigne la rapidité avec laquelle l'arbre du moteur tourne. Elle est souvent liée à la fréquence du courant alternatif dans les moteurs asynchrones, mais peut être contrôlée à l'aide de variateurs de vitesse. La relation entre le couple et la vitesse est un point central dans la caractérisation du point de fonctionnement d'un moteur.

Le point de fonctionnement : d'un moteur est donc défini par une combinaison spécifique de couple et de vitesse (*Torque, Speed*), et il évolue en fonction des conditions de charge et des paramètres d'alimentation du moteur. Un suivi précis de ces deux paramètres est essentiel pour la maintenance et le diagnostic des moteurs.

- **Glissement**

Le glissement (Slip en anglais) est généralement exprimé en pourcentage. Il représente le décalage relatif entre la vitesse de rotation du champ magnétique du stator, appelée vitesse synchrone et la vitesse réelle du rotor, et constitue une caractéristique clé du fonctionnement des moteurs asynchrones.

La formule du glissement (s) est donnée par :

$$s = \frac{N_s - N_r}{N_s} \cdot 100 \quad (5.1)$$

où :

- N_s est la vitesse synchrone, exprimée en tours par minute (tr/min),
- N_r est la vitesse réelle du rotor, exprimée en tours par minute (tr/min).

- **Courant et Tension**

Le courant et la tension sont deux paramètres électriques fondamentaux pour le fonctionnement des moteurs électriques.

- **Courant (I)** : Le courant électrique, mesuré en ampères (A), est le flux de charges électriques à travers les enroulements du moteur. Une mesure précise du courant est cruciale pour détecter les anomalies, telles que les surcharges ou les déséquilibres, qui peuvent endommager le moteur.
- **Tension (U)** : La tension, mesurée en volts (V), est la différence de potentiel qui pousse le courant à travers le moteur. Une tension correcte est nécessaire pour assurer que le moteur fonctionne à sa capacité optimale. Des fluctuations de tension peuvent affecter le couple produit et la vitesse de rotation.

5.4 Données utilisées

Dans ce projet, les données exploitées proviennent des drives de Schneider Electric, qui jouent un rôle crucial dans le contrôle et la surveillance des moteurs électriques asynchrones. Ces drives sont capables de réguler finement les paramètres essentiels du moteur, tels que le couple et la vitesse, le contrôle précis du couple (T) et de la vitesse (S) est essentiel pour assurer une performance optimale et éviter les défaillances.

Les variables critiques collectées par les drives de Schneider Electric comprennent principalement les tensions triphasées (V_1, V_2, V_3) et les courants triphasés (I_1, I_2, I_3) aux bornes du moteur. Ces variables sont fondamentales pour la caractérisation du fonctionnement électrique du moteur, car elles reflètent directement les conditions de charge et les phénomènes dynamiques internes, leurs variations peuvent signaler des anomalies telles que des déséquilibres, des harmoniques ou des surcharges.

Méthodologie de Collecte des Données

Avant d'initier la collecte de données, il est impératif de définir avec précision le point de fonctionnement du moteur, c'est-à-dire la combinaison spécifique de couple et de vitesse (T, S). Chaque point de fonctionnement est déterminé par une configuration précise de ces deux paramètres, et la collecte des variables électriques mentionnées plus haut est réalisée pour chaque combinaison de T et S sélectionnée. Cette méthode permet d'établir un profil complet du moteur sous différentes conditions de charge et de vitesse.

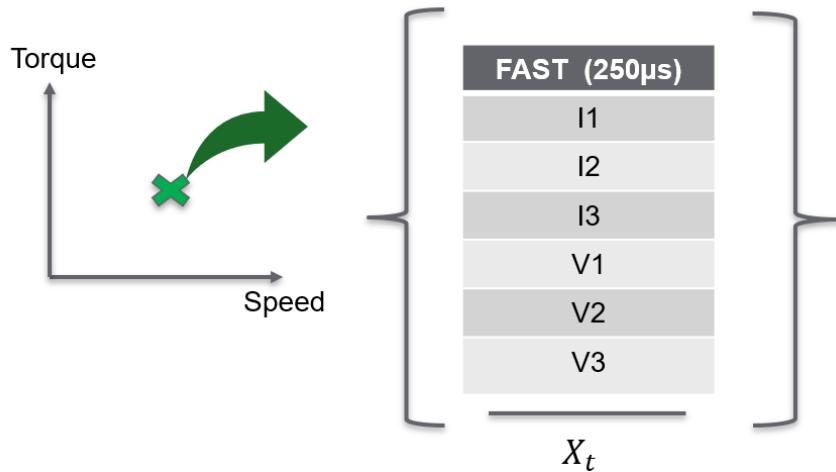


FIG. 5.2 : Organization de data [YOUN et al. 2019].

La collecte des données s'effectue sur une durée de 20 secondes, avec un échantillonnage à haute fréquence, toutes les 250 (μs). Ce taux d'échantillonnage élevé permet de capturer les variations rapides et les dynamiques transitoires du moteur, fournissant ainsi un aperçu détaillé de son comportement électrique. Au terme de cette période de collecte, un tableau de 80 000 points de données est créé, chaque point étant enregistré de manière chronologique pour former une série temporelle.

Par exemple, la série temporelle illustrée dans la figure correspond au point de fonctionnement (Couple = 0%, Vitesse = 15%). Cette série est composée de 80 000 points, chaque point ayant été collecté à un intervalle de 250 (μs).

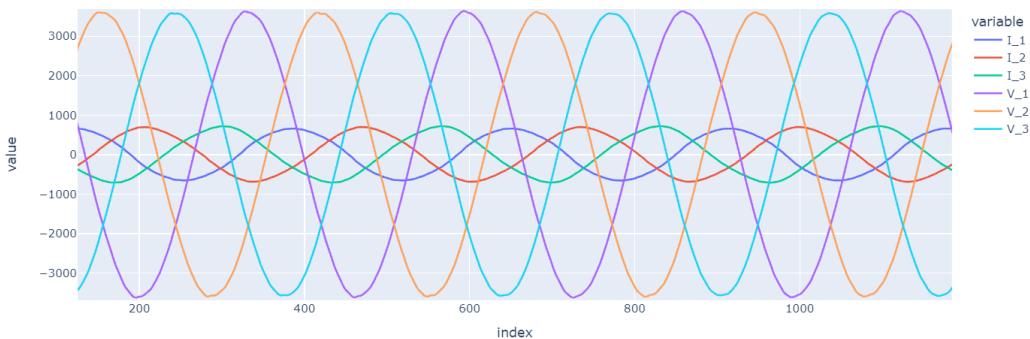


FIG. 5.3 : Organization de data .

Pour enrichir notre jeu de données, nous ajoutons des colonnes supplémentaires représentant la vitesse, le couple, ainsi que le glissement. Ces nouvelles variables permettent de mieux caractériser le comportement du moteur sous différentes conditions de fonctionnement.

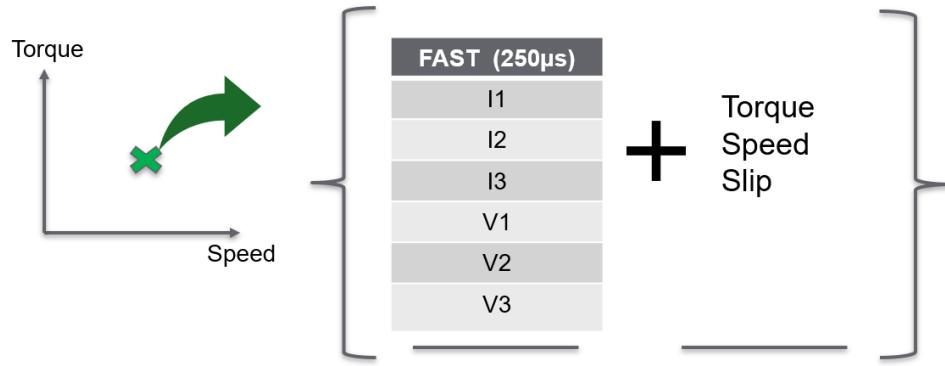


FIG. 5.4 : Organisation de data.

Segmentation des Données

Un autre défi majeur dans la génération de données est lié à la taille considérable des séries temporelles. Par exemple, un seul point de fonctionnement est représenté par une série temporelle composée de 80 000 points de données. Cette quantité massive de données pose des contraintes significatives en termes de capacité de calcul et de stockage. De plus, pour les modèles de réseaux de neurones récurrents ou autres modèles séquentiels, il est difficile de maintenir les dépendances temporelles sur une séquence aussi longue. En effet, un modèle qui traite un point à la position 70 000 pourrait avoir du mal à conserver une mémoire efficace des informations importantes présentes à la position 10, ce qui peut compromettre la qualité des prédictions.

Pour rendre le processus plus gérable et optimiser l'apprentissage des modèles, nous avons décidé de segmenter les séries temporelles en sous-séquences plus courtes. Dans notre cas, nous avons choisi de segmenter les séries temporelles en blocs de taille définie par un paramètre `seq_len`, qui se situe dans l'intervalle [1000, 2000]. Cette segmentation permet non seulement de réduire la charge computationnelle, mais aussi de faciliter l'entraînement des modèles en leur permettant de mieux capturer les relations temporelles à l'intérieur de ces sous-séquences plus courtes. Ainsi, chaque segment conserve une portion significative de la dynamique du moteur tout en rendant l'apprentissage plus efficace et plus pratique.

Normalisation des Données

Les valeurs de courant et de tension mesurées dans les moteurs électriques sont souvent très élevées, ce qui peut poser des problèmes lors de l'entraînement des modèles de machine learning, en particulier pour les réseaux de neurones. Afin de rendre ces valeurs plus compatibles avec les algorithmes d'apprentissage, il est essentiel de les normaliser.

La normalisation consiste à transformer les valeurs de courant et de tension pour les ramener dans une échelle plus réduite et uniforme. Cela présente plusieurs avantages. Premièrement, elle permet de prévenir la domination des valeurs de tension, qui sont généralement supérieures aux valeurs de courant. En effet, si les valeurs de tension ne sont pas normalisées, elles peuvent influencer de manière disproportionnée les décisions du modèle, au détriment des valeurs de courant. En normalisant ces deux ensembles

de données dans un même intervalle, nous assurons un traitement équitable de toutes les variables par le modèle, évitant ainsi les biais induits par des échelles de grandeur différentes.

Deuxièmement, la normalisation facilite l'entraînement des réseaux de neurones en accélérant leur convergence vers un minimum global ou local. Les réseaux de neurones sont particulièrement sensibles aux échelles des données d'entrée. Lorsque les entrées varient sur des plages de valeurs très différentes, les gradients utilisés pour ajuster les poids des connexions peuvent devenir instables, ralentissant ainsi le processus d'apprentissage et rendant plus difficile la minimisation de la fonction de coût. En normalisant les données, on obtient des gradients plus homogènes, ce qui améliore la stabilité et la rapidité de convergence de l'algorithme d'optimisation.

Pour cela, nous avons mis en œuvre deux méthodes de normalisation : la normalisation Min-Max et la normalisation nominale, cette dernière étant couramment utilisée par l'équipe de Schneider Electric.

MinMax

La normalisation Min-Max est une technique de prétraitement des données qui permet de redimensionner les valeurs d'une variable afin qu'elles se situent dans un intervalle spécifique, généralement entre 0 et 1. La normalisation Min-Max est définie par la formule suivante :

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (5.2)$$

où :

x : la valeur originale de la variable

x_{\min} : la valeur minimale de la variable dans l'ensemble de données

x_{\max} : la valeur maximale de la variable dans l'ensemble de données

x' : la valeur normalisée

Cette transformation ramène toutes les valeurs des courants et des tensions dans l'intervalle [0, 1], comme illustré dans la figure (5.5).

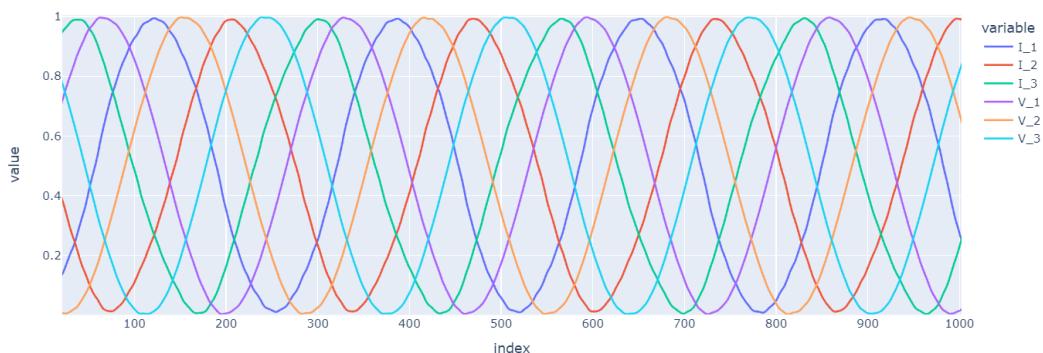


FIG. 5.5 : Organization de data [YOON et al. 2019].

Normalisation Nominale

En utilisant la normalisation nominale, telle que proposée par l'équipe de Schneider Electric, l'algorithme se présente comme suit :

Algorithm 1 : Normalization Nominale

Input : Valeurs des courants et tensions

Output : Valeurs normalisées des courants et tensions

1 Initialisation :

$$2 \quad V_{\text{nom}} \leftarrow 400$$

$$3 \quad V_{\text{max}} \leftarrow 921$$

$$4 \quad I_{\text{nom}} \leftarrow 10.2$$

$$5 \quad I_{\text{max}} \leftarrow 35.59$$

6 Normalisation :

$$7 \quad I_1 \leftarrow I_1 \frac{I_{\text{max}}}{2^{12} I_{\text{nom}}}$$

$$8 \quad I_2 \leftarrow I_2 \frac{I_{\text{max}}}{2^{12} I_{\text{nom}}}$$

$$9 \quad I_3 \leftarrow I_3 \frac{I_{\text{max}}}{2^{12} I_{\text{nom}}}$$

$$10 \quad V_1 \leftarrow V_1 \frac{V_{\text{max}}}{2^{12} V_{\text{nom}}}$$

$$11 \quad V_2 \leftarrow V_2 \frac{V_{\text{max}}}{2^{12} V_{\text{nom}}}$$

$$12 \quad V_3 \leftarrow V_3 \frac{V_{\text{max}}}{2^{12} V_{\text{nom}}}$$

13 return Valeurs normalisées des courants I_1, I_2, I_3 et des tensions V_1, V_2, V_3

Dans la figure suivante, nous observons les courants et tensions normalisés avec cette technique :

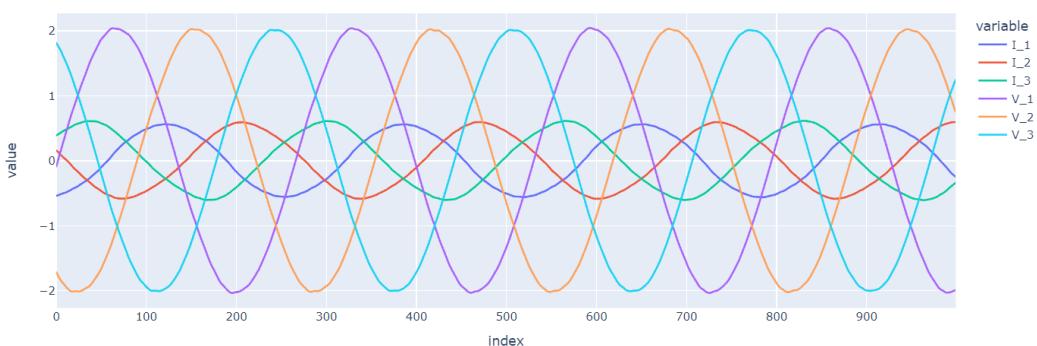


FIG. 5.6 : Organization de data [YOON et al. 2019].

5.5 Solution proposée

5.5.1 TimeGAN

Dans notre approche pour générer des données sous forme de séries temporelles correspondant à celles des moteurs électriques, nous nous sommes inspirés d'un modèle génératif nommé Time-series GAN (TimeGAN), présenté dans l'article de [YOON et al. 2019]. Dans ce qui suit, nous présenterons en détail l'architecture de notre modèle génératif, en exposant les différents composants ainsi que les fondements théoriques sur lesquels repose notre modèle.

Architecure de TimeGAN

Compte tenu de la nature séquentielle des données utilisées, qui prennent la forme de séries temporelles, l'architecture des réseaux de neurones adoptée repose sur des réseaux de neurones récurrents (RNN). Afin d'améliorer les performances du modèle, une architecture basée sur des réseaux LSTM (Long Short-Term Memory) a été intégrée, en complément de réseaux entièrement connectés (fully connected layers). La figure (5.7) présente l'architecture globale du modèle génératif proposé.

Le modèle TimeGAN se structure autour de quatre réseaux de neurones principaux : une fonction d'embedding (encodeur), une fonction de recovery (décodeur), un générateur de séquences et un discriminateur. Ces quatre réseaux interagissent au sein d'un espace latent une représentations réduites des données.

Réseaux d'Embedding et de Recovery

Les fonctions d'embedding et de recovery établissent les liens entre l'espace des caractéristiques et l'espace latent, permettant au modèle d'apprendre les dynamiques temporelles des données sous une forme réduite. Le réseau d'embedding convertit les données de séries temporelles en une représentation latente. Le réseau de recovery cherche ensuite à reconstruire la série temporelle originale à partir de cette représentation réduite. La fonction de perte vise à minimiser l'écart entre la série temporelle initiale et celle reconstruite à partir de l'espace latent.

Réseaux de Générateur et de Discriminateur

On commence par le réseau générateur, qui prend en entrée un bruit (bruit gaussien) et tente de générer des vecteurs dans l'espace latent, en s'assurant que ces vecteurs se rapprochent de ceux encodés par le réseau d'embedding. Ensuite, le discriminateur intervient en prenant en entrée les vecteurs générés par le générateur et vérifie leur similitude avec les vecteurs originaux encodés par le réseau de recovery. Le discriminateur guide et ajuste ainsi le générateur, de sorte que la fonction de perte incite le générateur à produire des vecteurs de plus en plus similaires aux vecteurs originaux.

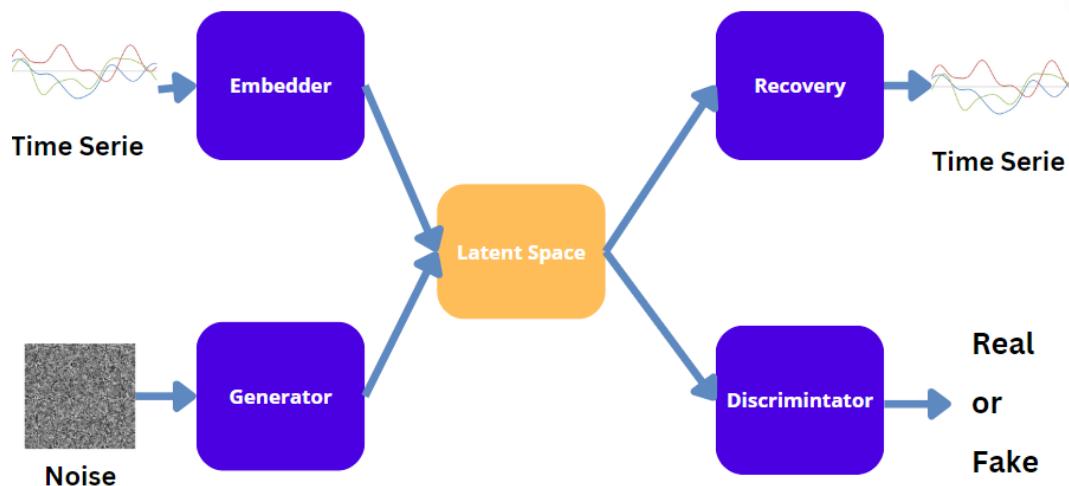


FIG. 5.7 : Architecure de TimeGAN[YOON et al. 2019].

Génération de données

Une fois que l'entraînement du modèle est terminé et que le modèle a convergé, la génération de données devient simple. Il suffit de fournir un bruit au réseau générateur, qui va alors produire des vecteurs dans l'espace latent. Ces vecteurs sont ensuite décodés en séries temporelles par le réseau de recovery.

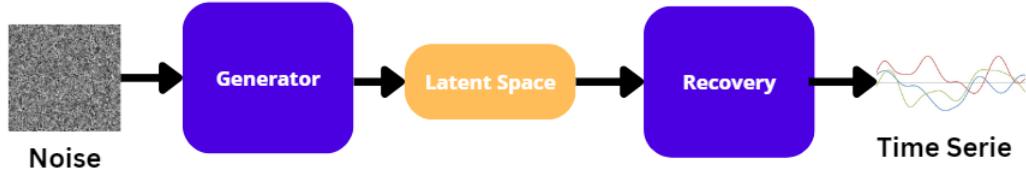


FIG. 5.8 : Architecture de TimeGAN[YOON et al. 2019].

5.6 Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes suivies pour concevoir notre méthode de génération de séries temporelles. Nous avons détaillé l'architecture globale de notre modèle, qui se compose de quatre réseaux de neurones : le génératrice, le discriminateur, le réseau de récupération et le réseau d'embedding. Nous avons également montré comment ces réseaux génèrent les données et comment ils fonctionnent ensemble dans un espace latent, utilisé pour la compression des données.

Nous avons expliqué l'organisation des jeux de données et l'importance de la segmentation des séries temporelles. De plus, nous avons discuté des techniques de normalisation, que ce soit avec la méthode MinMax ou avec la technique proposée par STIE.

Dans le chapitre suivant, nous présenterons nos expérimentations sur notre méthode pour générer des données pour des moteurs asynchrones.

Chapitre 6

Réalisation et tests

6.1 Introduction

Après avoir exposé les notions de base et les détails de notre solution pour la génération de données, ainsi que les types de séries temporelles de moteurs électriques avec lesquelles nous travaillons, ce chapitre présente les langages de programmation, les bibliothèques et les frameworks sélectionnés pour mettre en œuvre notre approche. Nous y décrivons également l'environnement technique dans lequel notre solution a été conçue, et nous évaluons la performance de notre solution ainsi que la qualité des données générées par notre modèle génératif.

6.2 Technologies utilisées

6.2.1 Python

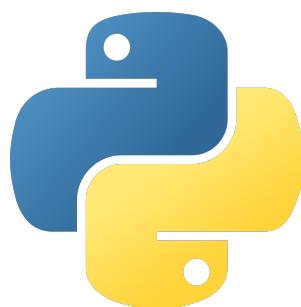


FIG. 6.1 : Python.

!

Python joue un rôle important dans le domaine de l'intelligence artificielle (IA) grâce à sa polyvalence, sa simplicité et sa richesse en bibliothèques spécialisées. En tant que langage de programmation, Python est devenu le premier choix pour de nombreux chercheurs, ingénieurs et développeurs travaillant dans le domaine de l'IA. D'une part, Python

est connu pour sa syntaxe claire et lisible, ce qui permet aux nouveaux arrivants dans le domaine de l'IA de se familiariser rapidement avec les concepts de base. D'autre part, Python offre plusieurs bibliothèques et frameworks spécialisés dans l'IA, tels que TensorFlow, Keras, PyTorch et Scikit-learn. Ces bibliothèques sont souvent utilisées pour le développement de réseaux de neurones, d'algorithmes d'apprentissage automatique et d'autres techniques d'IA. Finalement, Python est un langage polyvalent qui permet aux développeurs de créer et tester différentes approches rapidement.

6.3 Bibliothèque utilisées

Python offre une riche collection de bibliothèques spécialisées dans plusieurs domaines, en particulier l'IA. Ces bibliothèques offrent des outils et des frameworks puissants pour développer des modèles d'apprentissage automatique, des réseaux de neurones, et bien plus encore. Dans cette section, nous allons explorer les bibliothèques que nous avons utilisé pour implémenter et tester notre méthode.

6.3.1 NumPy



FIG. 6.2 : NumPy.

NumPy¹ est une bibliothèque open source qui offre de nombreuses fonctionnalités pour le calcul numérique en Python. Elle offre un support puissant pour la manipulation de tableaux multidimensionnels, ainsi que pour l'exécution de calculs mathématiques complexes sur ces tableaux. Ces tableaux multidimensionnels permettent de stocker et de manipuler efficacement des données numériques sous forme de matrices et de vecteurs. Elle fournit également des fonctions mathématiques de base, des opérations d'algèbre linéaire, des opérations sur les tableaux, des fonctions statistiques et bien plus encore. Elle est largement utilisé en IA pour le traitement et la manipulation de données, la préparation de jeux de données, ainsi que pour la mise en œuvre d'algorithmes d'apprentissage automatique et de réseaux de neurones. La performance élevée de NumPy en calcul numérique en fait un bon choix pour les tâches intensives en termes de calcul.

6.3.2 Pandas

¹acronyme de "Numerical Python"



FIG. 6.3 : Pandas.

Pandas est une bibliothèque essentielle en Python pour la manipulation et l'analyse de données. Elle fournit des structures de données rapides, flexibles et expressives, notamment les DataFrames, qui permettent de gérer et d'analyser des ensembles de données de manière efficace. Pandas est conçu pour rendre la manipulation des données aussi simple que possible, qu'il s'agisse de nettoyage, de transformation, ou d'agrégation de données.

Avec Pandas, il est facile de lire et d'écrire des données à partir de diverses sources, telles que des fichiers CSV, Excel, SQL, ou encore des formats JSON, tout en conservant la structure tabulaire des données. Une fois les données chargées, Pandas permet de les filtrer, les trier, les regrouper, les fusionner, et de réaliser des opérations arithmétiques ou des transformations complexes sur celles-ci.

6.3.3 Plotly



FIG. 6.4 : Plotly.

Plotly est une bibliothèque de visualisation en Python qui permet de créer des graphiques interactifs et statiques de haute qualité. Cette bibliothèque se distingue par sa capacité à générer une large gamme de graphiques, allant des graphiques linéaires et en barres aux graphiques de dispersion, cartes choroplèthes, graphiques en 3D, et bien plus encore. Plotly offre des fonctionnalités interactives avancées, telles que le zoom, le panoramique et l'affichage de détails au survol des données, ce qui facilite l'exploration et l'analyse des données complexes. Plotly est particulièrement utile dans les domaines de la science des données, de l'intelligence artificielle et de l'apprentissage automatique, où il est couramment utilisé pour visualiser les performances des modèles, les distributions de données, les tendances, les matrices de confusion, et les courbes d'apprentissage. Sa capacité à produire des visualisations interactives en fait un outil précieux pour les analyses exploratoires et la présentation des résultats.

6.3.4 Pytorch



FIG. 6.5 : Pytorch.

PyTorch est une bibliothèque open-source d'apprentissage automatique et d'intelligence artificielle en Python, développée principalement par Facebook's AI Research lab (FAIR). Elle repose sur un concept fondamental appelé "tenseur", qui est une structure de données multidimensionnelle similaire aux tableaux NumPy et elle est conçue pour faciliter le développement et la mise en œuvre de modèles de réseaux de neurones profonds. PyTorch est surtout distinguée par sa prise en charge des calculs automatiques de gradients, ce qui signifie qu'il est possible de définir des opérations mathématiques sur les tenseurs et que PyTorch peut automatiquement calculer les gradients de ces opérations qui sont nécessaires pour ajuster les poids dans les réseaux de neurones et ainsi minimiser une fonction de perte. Elle est utilisé pour la conception des modèles de réseaux de neurones profonds, y compris les réseaux de neurones convolutionnels (CNN), les réseaux de neurones récurrents (RNN), les transformateurs, etc.

6.4 Outils utilisés

6.4.1 Google Colab



FIG. 6.6 : Google Colab.

Google Colab (abrégé de Colaboratory) est une plateforme de notebooks interactifs basée sur le cloud, développée par Google. Elle permet aux utilisateurs d'écrire, d'exécuter et de partager du code Python de manière collaborative, sans nécessiter de configuration ou d'installation. Elle propose des notebooks interactifs qui permettent d'insérer des cellules de code exécutable et des cellules de texte et chaque notebook Colab s'exécute dans un environnement virtuel où les utilisateurs peuvent accéder à la puissance de calcul des

processeurs graphiques (GPU) et des unités de traitement tensoriel (TPU) pour accélérer l'entraînement de modèles d'apprentissage automatique. Elle propose également de nombreuses bibliothèques préinstallées, mais les utilisateurs peuvent également installer et utiliser des bibliothèques tierces via des commandes simples.

6.4.2 Google Drive



FIG. 6.7 : Google Drive.

Google Drive est un service de stockage en ligne développé par Google pour stocker, synchroniser et partager des fichiers et des dossiers sur le cloud. Il offre une variété de fonctionnalités telles que le stockage en ligne, la synchronisation multi-appareils, le partage de fichiers, la collaboration en temps réel, etc. De plus, il peut être utilisé avec Google Colab.

6.5 Environnement de développement

Préparation de données

Pour les données utilisées, elles proviennent de moteurs asynchrones réels de Schneider Toshiba Inverter Europe (STIE) à Pacy-sur-Eure. Ces données ont été collectées par les automaticiens de STIE et nous ont été transmises sous forme de fichiers MATLAB. Nous les récupérons, les convertissons en fichiers CSV, puis utilisons la bibliothèque Pandas pour les traiter. Avant d'exploiter les données, nous les normalisons en appliquant soit la normalisation Min-Max, soit la normalisation nominale recommandée par l'équipe de STIE. Ensuite, nous procédons à la segmentation des données, et nous présentons l'algorithme de segmentation utilisé comme suit.

Serveur de Schneider Electric

Pour l'entraînement des modèles, nous utilisons un serveur dédié chez Schneider Electric. Ce serveur est spécialement configuré pour gérer les tâches de calcul intensif requises

par l'entraînement de modèles complexes. Grâce à cette infrastructure, nous pouvons traiter de grandes quantités de données en parallèle, ce qui améliore significativement la vitesse et l'efficacité des processus de modélisation.

Le serveur chez Schneider Electric est optimisé pour les applications d'apprentissage automatique, disposant de GPU performants pour accélérer les calculs. Les données collectées à partir des moteurs asynchrones réels, après avoir été normalisées et segmentées, sont transférées sur ce serveur pour l'entraînement des modèles prédictifs. Cette infrastructure nous permet d'itérer rapidement sur différents modèles, d'ajuster les paramètres et d'obtenir des résultats précis dans des délais raisonnables.

6.6 Tests et résultats

6.6.1 Méthodes D'évaluation

Pour évaluer la qualité des données générées par notre modèle, nous employons les métriques proposées dans [YOUN et al. 2019], qui assurent les aspects clés suivants :

1. **Diversité** :les échantillons générés doivent couvrir de manière adéquate la distribution des données réelles ;
2. **Fidélité** :les échantillons générés doivent être indistinguables des données réelles ;
3. **Utilité** :les échantillons générés doivent être tout aussi utiles que les données réelles pour les tâches prédictives.

Pour assurer ces trois qualités : **diversité**, **fidélité**, et **utilité**, nous utilisons les métriques suivantes :

1. Visualisation

Nous appliquons les analyses t-SNE [MAATEN et al. 2008] et PCA [WOLD et al. 1987] sur les ensembles de données originaux et synthétiques (en comprimant la dimension temporelle). Cela permet de visualiser dans un espace à 2 dimensions dans quelle mesure la distribution des échantillons générés ressemble à celle de l'original, donnant ainsi une évaluation qualitative de la diversité.

2. Score Discriminant

Pour une mesure quantitative de la similarité, nous entraînons un modèle de classification de séries temporelles (en optimisant un LSTM à 2 couches) pour distinguer les séquences provenant des ensembles de données originaux et générés. Tout d'abord, chaque séquence originale est étiquetée *réelle*, et chaque séquence générée est étiquetée *non réelle*. Ensuite, un classificateur RNN standard est entraîné pour distinguer les deux classes dans le cadre d'une tâche supervisée standard. Nous rapportons ensuite l'erreur de classification sur l'ensemble de test réservé (données synthétiques), ce qui donne une évaluation quantitative de la fidélité.

3. Score Prédictif

Pour être utile, les données échantillonnées doivent hériter des caractéristiques prédictives de l'original. En particulier, nous nous attendons à ce que notre méthode excelle dans la capture des distributions conditionnelles au fil du temps. Par conséquent, en utilisant l'ensemble de données synthétiques, nous entraînons un modèle de prédiction de séries temporelles (en optimisant un LSTM à 2 couches) pour prédire les vecteurs temporels du prochain pas sur chaque séquence d'entrée. Ensuite, nous évaluons le modèle entraîné sur l'ensemble de données original. La performance est mesurée en termes d'erreur absolue moyenne (MAE).

6.6.2 Tests et résultats

Cette section finale propose une analyse approfondie des résultats obtenus après avoir testé notre méthode de génération de données de séries temporelles pour les moteurs asynchrones. Les résultats sont évalués en fonction de différents paramètres de notre modèle ainsi que de l'organisation de notre jeu de données. Nous concluons par une synthèse des tests effectués, en mettant en évidence les améliorations potentielles et les ajustements nécessaires pour optimiser notre approche.

Nous configurons notre modèle et notre jeu de données selon les paramètres présentés dans le tableau ci-dessous :

Paramètre	Valeur
Longueur de séquence	1000
Normalisation	Min-Max
Taille de l'espace latent pour le modèle	32
Nombre de couches pour les réseaux de neurones	4

TAB. 6.1 : Configuration du modèle et du jeu de données

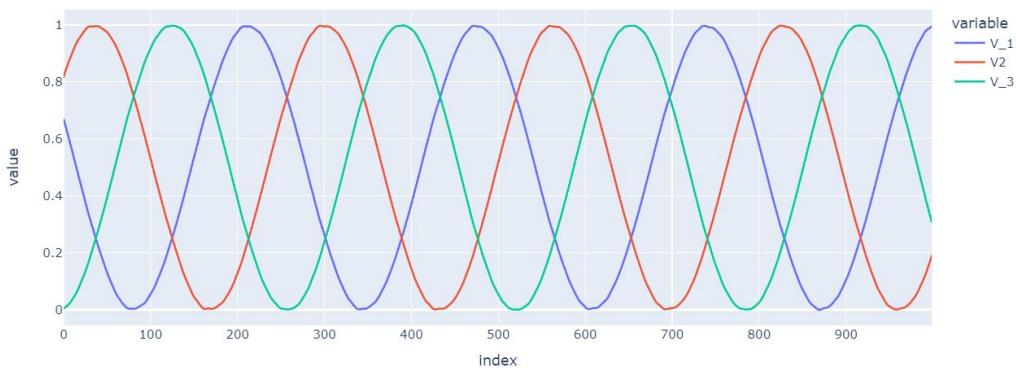


FIG. 6.8 : Seq.

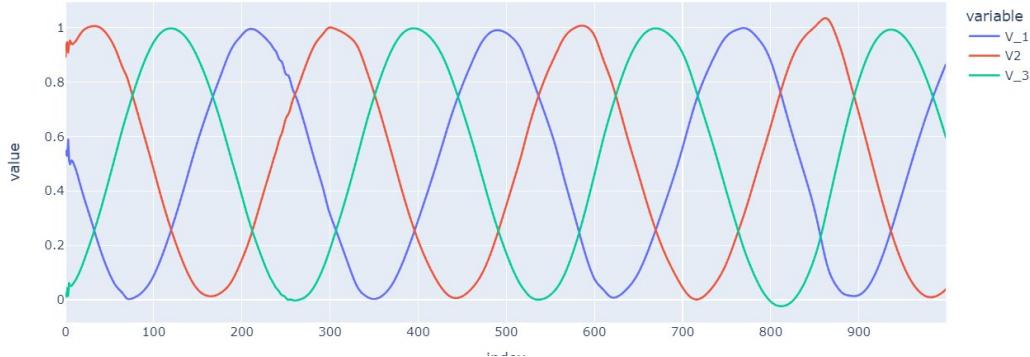


FIG. 6.9 : Seq.

Score Discriminant (Lower the better) : 0.1568

Score Prédictif (Lower the better) : 0.5138

Paramètre	Valeur
Longueur de séquence	2000
Normalisation	Min-Max
Taille de l'espace latent pour le modèle	32
Nombre de couches pour les réseaux de neurones	4

TAB. 6.2 : Configuration du modèle et du jeu de données

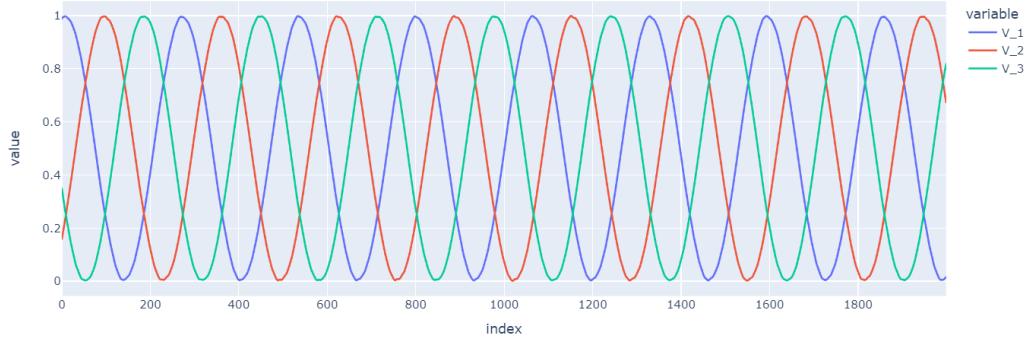


FIG. 6.10 : Seq.

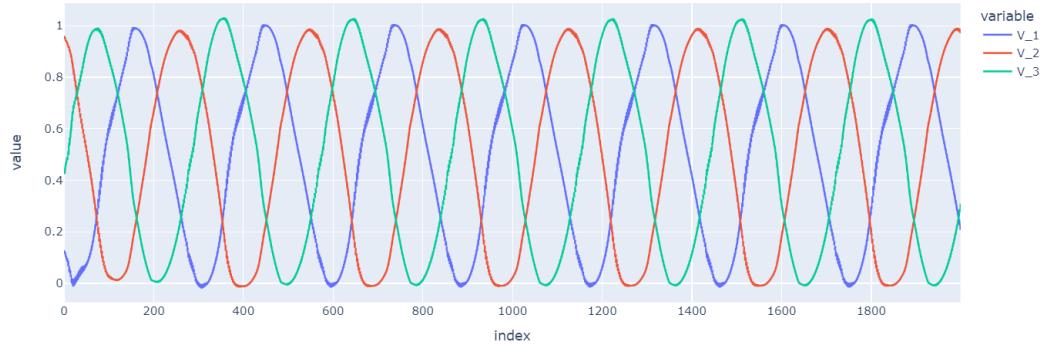


FIG. 6.11 : Seq.

Score Discriminant (Lower the better) : 0.4

Score Prédictif (Lower the better) : 0.3465

6.7 Conclusion

En conclusion de ce chapitre dédié aux tests et aux résultats, nous avons présenté en détail la réalisation technique de notre solution. Cette section a permis de mettre en lumière les outils et technologies essentiels à la mise en œuvre de notre solution pour la génération de données de séries temporelles pour les moteurs électriques.

Les expérimentations menées ont montré l'importance de la segmentation dans la génération des séries temporelles. En particulier, nous avons observé que la longueur de segmentation joue un rôle crucial : lorsque les segments sont plus courts, la génération des séries temporelles avec cette longueur produit des séries de meilleure qualité.

De plus, nous avons utilisé plusieurs métriques, telles que le score discriminatif et le score prédictif, pour évaluer la qualité des données générées. Ces évaluations ont confirmé la pertinence des données synthétiques pour les tâches prédictives.

En conclusion, ce chapitre a fourni une vue d'ensemble complète de la réalisation technique de notre projet et des résultats obtenus. Les informations présentées ici serviront de base pour les discussions et analyses futures, et ouvriront la voie à des améliorations et extensions potentielles de notre solution.

Chapitre 7

Conclusion et perspectives

7.1 Perspectives

7.2 Appréciation personnelle

Bibliographie

- AGGARWAL, Charu C. (2018). *Neural networks and deep learning : A textbook*. Springer.
- ALHARBI, Nouf Fahad et Nabil M. HEWAHI (2021). “Exploring deep neural network capability for intrusion detection using different mobile phones platforms”. In : *International Journal of Computing and Digital Systems* 10.1, p. 1391-1406.
- AMINI, Alexander et al. (2018). “Spatial uncertainty sampling for end-to-end control”. In : *arXiv preprint arXiv :1805.04829*.
- BENTIVOGLI, Andrea et al. (2023). “Enabling Predictive Maintenance on Electric Motors Through a Self-sustainable Wireless Sensor Node”. In : *Applications in Electronics Pervading Industry, Environment and Society*. Sous la dir. de Riccardo BERTA et Alessandro De GLORIA. Springer Nature Switzerland AG, p. 13-29.
- BISHOP, Christopher M. (2016). *Pattern recognition and machine learning*. Springer New York.
- BONET-JARA, Jorge et al. (2023). “Sensorless Speed Estimation for the Diagnosis of Induction Motors via MCSA. Review and Commercial Devices Analysis”. In : * ORCID.
- BROPHY, Eoin et al. (juill. 2023). “Generative Adversarial Networks in Time Series : A Survey and Taxonomy”. In : *A Preprint*.
- COLLIER, Mark, Alfredo NAZABAL et Christopher K.I. WILLIAMS (2020). “VAEs in the Presence of Missing Data”. In : *arXiv preprint arXiv :2006.05301*.
- DHARIWAL, Prafulla et Alex NICHOL (2021). “Diffusion Models Beat GANs on Image Synthesis”. In : *arXiv preprint arXiv :2105.05233*.
- DONG, Peijie et al. (juin 2022). “Prior-guided one-shot neural architecture search”. In : *arXiv.org*.
- FAWAZ, Hassan Ismail et al. (2019a). “A survey on long short-term memory networks for time series prediction”. In : *Physica A : Statistical Mechanics and its Applications* 534, p. 122-315.
- (2019b). “Deep learning for time series classification : a review”. In : *Data Mining and Knowledge Discovery* 33.4, p. 917-963.
- FENG, Jie et al. (2020). “Generative Adversarial Networks based on collaborative learning and attention mechanism for hyperspectral image classification”. In : *Remote Sensing* 12.7, p. 1149.
- FENG, Junxi et al. (2019). “Reconstruction of Porous Media from extremely limited information using conditional generative adversarial networks”. In : *Physical Review E* 100.3.
- GOODFELLOW, Ian, Yoshua BENGIO et Aaron COURVILLE (2016). *Deep Learning*. MIT Press.

Bibliographie

- GOODFELLOW, Ian et al. (2014). "Generative adversarial nets". In : *Advances in neural information processing systems*, p. 2672-2680.
- HE, Kaiming et al. (juin 2016). "Deep Residual Learning for Image Recognition". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- HO, Jonathan, Ajay JAIN et Pieter ABBEEL (2020). "Denoising Diffusion Probabilistic Models". In : *arXiv preprint arXiv :2006.11239*.
- HOCHREITER, Sepp et Jürgen SCHMIDHUBER (1997). "Long short-term memory". In : *Neural computation* 9.8, p. 1735-1780.
- HU, Rong et al. (2022). "A multi-attack intrusion detection model based on mosaic coded convolutional neural network and centralized encoding". In : *PLOS ONE* 17.5.
- JOHNSON, Don H. et Sinan SINANOVIĆ (mars 2001). "Symmetrizing the Kullback-Leibler Distance". In : Available at : <http://www.ece.rice.edu/dhj/paper1.pdf>.
- KARRAS, Tero, Samuli LAINE et Timo AILA (2019). "A style-based generator architecture for generative adversarial networks". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 4401-4410.
- KIMURA, Nobuaki et al. (2019). "Convolutional neural network coupled with a transfer-learning approach for time-series flood predictions". In : *Water* 12.1, p. 96.
- KINGMA, Diederik P. et Max WELLING (2012). "Auto-Encoding Variational Bayes". In.
- KUMARASWAMY, Balachandra (2021). "Neural Networks for Data Classification". In : *Artificial Intelligence in Data Mining*, p. 109-131.
- LUHMAN, Troy et Eric LUHMAN (2024). "High Fidelity Image Synthesis With Deep VAEs In Latent Space". In.
- MAATEN, Laurens van der et Geoffrey HINTON (2008). "Visualizing data using t-SNE". In : *Journal of Machine Learning Research* 9.Nov. Sous la dir. d'Yoshua BENGIO, p. 2579-2605.
- MCCULLOCH, Warren S. et Walter PITTS (1943). "A logical calculus of the ideas immanent in nervous activity". In : *The Bulletin of Mathematical Biophysics* 5.4, p. 115-133.
- MROZEK, Dariusz et al. (2023). "From Corrective to Predictive Maintenance—A Review of Maintenance Approaches for the Power Industry". In : *Sensors* 23.13, p. 5970.
- MUNIASAMY, Anandhavalli et al. (2020). "Deep Learning for Predictive Analytics in Healthcare". In : *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019)*. Sous la dir. d'Aboul Ella HASSANIEN et al. Cham : Springer International Publishing, p. 32-42.
- NICHOL, Alex et Prafulla DHARIWAL (2021). "Improved Denoising Diffusion Probabilistic Models". In : *arXiv preprint arXiv :2102.09672*.
- RAMESH, Aditya et al. (2022). "Hierarchical Text-Conditional Image Generation with CLIP Latents". In : *arXiv preprint arXiv :2204.06125*.
- ROSENBLATT, F. (1958). "The Perceptron : A probabilistic model for information storage and organization in the brain." In : *Psychological Review* 65.6, p. 386-408.
- SANGEETHA, P. et S. HEMAMALINI (2017). "An IoT and Machine Learning-Based Predictive Maintenance System for Electrical Motors". In : *IET Signal Processing* 11.5, p. 604-612.
- SEZDI, Mana (2019). "The Technical Support : Repair, Preventive Maintenance, and Inspection". In : sous la dir. de Sudip PAUL.

Bibliographie

- WANG, Wenlin et al. (2019). “Topic-Guided Variational Autoencoders for Text Generation”. In : *arXiv preprint arXiv :1903.07137*.
- WANG, Zexin et al. (2019). “Revisiting VAE for Unsupervised Time Series Anomaly Detection : A Frequency Perspective”. In : *arXiv preprint arXiv :1906.03821*.
- WIERING, Marco et Martijn van OTTERLO, éd. (2012). *Reinforcement Learning : State-of-the-Art*. Springer.
- WOLD, Svante, Kim H. ESBENSEN et Paul GELADI (1987). “Principal Component Analysis”. In : *Chemometrics and Intelligent Laboratory Systems* 2.1-3, p. 37-52.
- YANG, Yiyuan et al. (2023). “A Survey on Diffusion Models for Time Series and Spatio-Temporal Data”. In : *arXiv preprint arXiv :2301.08548*.
- YANG SONG, Stefano Ermon (2022). “Diffusion Models : A Comprehensive Survey of Methods and Applications”. In : *arXiv preprint arXiv :2209.00796*.
- YILMAZ, M. Akin et al. (2021). “Self-Organized Variational Autoencoders (Self-VAE) for Learned Image Compression”. In : *arXiv preprint arXiv :2104.05794*.
- YOON, Jinsung, Daniel JARRETT et Mihaela van der SCHaar (2019). “Time-series generative adversarial networks”. In : *Advances in Neural Information Processing Systems*. T. 32. NeurIPS.

Webographie

WANG, Phillip (2019). *This Person Does Not Exist*. <https://thispersondoesnotexist.com>. Accessed : 2024-08-03.